Selection Hyper-heuristics for Population-based Meta-heuristics in Continuous Dynamic Environments

by

Stéfan Aloysius Gert van der Stockt

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

December 2019

Publication data:

Stéfan Aloysius Gert van der Stockt. Selection Hyper-heuristics for Population-based Meta-heuristics in Continuous Dynamic Environments. Philosophiae Doctor thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, December 2019.

Electronic, hyper-linked versions of this dissertation are available on-line, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

Selection Hyper-heuristics for Population-based Meta-heuristics in Continuous Dynamic Environments

by

Stéfan Aloysius Gert van der Stockt
E-mail: stefan.vanderstockt@gmail.com

Abstract

Dynamic optimization problems provide a challenge in that optima have to be tracked as the environment changes. The complexity of a dynamic optimization problem is determined by the severity and frequency of changes, as well as the behavior of the values and trajectory of optima. While many efficient algorithms have been developed to solve these types of problems, the choice of the best algorithm is highly dependent on the type of change present in the environment. This thesis analyzes the ability of popular selection operators used in a hyper-heuristic framework to continuously select the most appropriate optimization method over time to solve a DOP better than the individual optimization methods can.

A contradictory situation faced by DOP-focused meta-heuristics is identified from literature: statically tuning meta-heuristic parameters for DOPs is impossible, yet dynamically adapting multiple meta-heuristic parameters in an ad hoc fashion produces poor results. The no free lunch (NFL) theorems for optimization are discussed, along with reasoning from literature as to why the conditions that are required for the NFL theorem to hold are practically impossible to find in real-world continuous-valued optimization problems. Hyper-heuristics are positioned as meta-search methods that can therefor raise performance in practical DOPs.

The *heterogeneous meta-hyper-heuristic* (HMHH) framework was originally devised for static environments. This thesis extends the HMHH framework by establishing the criteria needed to identify appropriate meta-heuristics that enable HMHHs to solve DOPs, introduces *global* and *island* neighborhoods that govern heuristic visibility of the

population of candidate solutions, and introduces different heuristic selection triggering mechanisms beyond time-based triggers. The HMHH framework is extended further to handle a mix of population-based meta-heuristics and single-solution methods under the same population-based paradigm.

A new performance measure for DOPs, namely the relative error distance, or P_r, is proposed that does not assume normally distributed performance data across an algorithm run, is resilient against fitness landscape scale changes, better incorporates performance variance across multiple fitness landscape changes, and allows easier algorithm comparisons using established nonparametric statistical methods. A new measure for heuristic diversity in population-based meta-heuristics, namely $\mathcal{N}(t)$, is introduced that is derived from Shannon's normalized entropy measure. Additional measures are proposed that consider the heuristic reassignment frequency, or δ, and heuristic reassignment volume, or φ, of a multi-population-based hyper-heuristic over the entire course of an algorithm run.

Empirical studies examine the performance and behavioral differences between various hyper-heuristic selection operators. The proposed experimental procedures for all algorithm evaluations, comparisons, and parameter sensitivity analysis rely entirely on nonparametric statistical procedures. Twenty-seven unique environments, based on the holistic classification of Duhain and Engelbrecht, are systematically created using the moving peaks benchmark function (MPB) generator. Parameter values are compliant with the generally accepted scenario 2 settings for the MPB. The results show that these hyper-heuristic approaches can yield higher performance more consistently across different types of environments.

Keywords: hyper-heuristics, dynamic environments, swarm intelligence, evolutionary computation.

Supervisors : Dr. Christopher Cleghorn
Prof. Andries Petrus Engelbrecht
Department : Department of Computer Science
Degree : Philosophiae Doctor

"*Know who knows what you know you don't know.*"

– Adrian Ray

"*You make decisions, take actions, affect the world, receive feedback from the world, incorporate it into yourself, then the updated 'you' makes more decisions, and so forth, 'round and 'round.*"

– Douglas Hofstadter

"*It has always seemed to me extreme presumptuousness on the part of those who want to make human ability the measure of what nature can and knows how to do, since, when one comes down to it, there is not one effect in nature, no matter how small, than even the most speculative minds can fully understand.*"

– Galileo Galilei

Acknowledgments

I would like to thank the following people for their ongoing support and guidance during the completion of this thesis:

- The Lord God almighty, for creating such a rich and complex world for inquisitive minds to explore, and for giving me the ability and opportunity to make an infinitesimally small contribution to mankind's understanding of His creation.
- My wife, Tarina van der Stockt, who stood by my side every day during this entire journey and without whom I would not have been able to simultaneously manage both life and writing this thesis!
- My parents, Charlene and Etienne van der Stockt, who laid the foundations of my entire education and encouraged me to take ever-larger steps in the pursuit of knowledge.
- My sister and brother-in-law, Karen and Gary, who were together on the same journey as Tarina and I from start to end, and were the only ones that truly experienced, endured, and overcame the same challenges as us.
- My managers and friends – Kai Young, Jeff Coveyduc, Rahul Akolkar, and Patrice Scully, who supported my pursuit of my academic dreams in extraordinary ways through the years while I was working on a full-time career as well. I could not have done this without your help.

Contents

List of Figures v

List of Algorithms ix

List of Tables x

1 Introduction 1

1.1 Introduction . . . 1
1.2 Motivation . . . 2
1.3 Objectives . . . 4
1.4 Contributions . . . 5
1.5 Scope . . . 7
1.6 Thesis Outline . . . 9

2 Dynamic Optimization 11

2.1 Introduction . . . 11
2.2 Optimization Problems . . . 12
2.3 Optimization in Dynamic Environments . . . 15
2.3.1 Classifications of Dynamic Optimization Problems . . . 17
2.4 The Moving Peaks Benchmark . . . 20
2.4.1 Extending the Moving Peaks Benchmark . . . 22
2.4.2 Criticism of the Moving Peaks Benchmark . . . 26
2.5 Algorithm Components for Dynamic Environment Optimization . . . 26
2.5.1 Maintaining/introducing Diversity . . . 27
2.5.2 Detecting Changes in the Environment . . . 27

2.5.3 Reacting to Changes . . . 28
2.5.4 Implicit/explicit Memory . . . 29
2.5.5 Multiple Populations . . . 30
2.5.6 Predicting Changes . . . 30
2.5.7 Self-adaptation . . . 31
2.6 Population-based Meta-heuristics for Dynamic Optimization . . . 31
2.6.1 Population-based Meta-heuristics in Evolutionary Computation . . . 32
2.6.2 Population-based Meta-heuristics in Swarm Intelligence . . . 34
2.6.3 Discussion of EC and SI Population-based Approaches . . . 38
2.7 Comparing Algorithm Performance in Dynamic Environments . . . 39
2.7.1 Facets of Performance Measures for Dynamic Environments . . . 40
2.7.2 Optimality-based Measures . . . 44
2.7.3 Behavior-based Performance Measures . . . 57
2.7.4 Statistical Comparison of Algorithm Performance . . . 62
2.8 Summary . . . 68

3 Selection Hyper-heuristics **70**
3.1 Introduction . . . 70
3.2 The Need for Adaptation in Optimization . . . 71
3.2.1 Meta-heuristics Parameter Tuning . . . 72
3.2.2 The No Free Lunch Theorems for Optimization . . . 73
3.3 Overview of Hyper-heuristics . . . 75
3.3.1 Classification of Hyper-heuristics . . . 75
3.3.2 Problem Space versus Heuristic Space . . . 77
3.3.3 Selection Hyper-heuristics for Real-valued Dynamic Environments 79
3.4 Selection Hyper-heuristics for Population-based Algorithms . . . 81
3.4.1 Multi-point Search Hyper-heuristics . . . 81
3.4.2 Neighborhood Topologies for Population-based Hyper-heuristics . . . 86
3.4.3 Heuristic Space Diversity . . . 88
3.4.4 Triggering Heuristic Allocation Changes . . . 89
3.4.5 Managing Heuristic State Information . . . 90
3.4.6 Selection Operators for HMHH . . . 93

3.4.7 Complementary Heuristics . . . 102
3.5 Related Approaches . . . 105
3.6 Summary . . . 107

4 Estimation of Performance Baselines for Control Groups 109
4.1 Introduction . . . 109
4.2 Measuring and Comparing DOP Performance . . . 110
4.3 Experimental Procedure . . . 117
4.3.1 Incorporation of DOP Algorithm Building Blocks . . . 117
4.3.2 Heuristic Pool Algorithms . . . 120
4.3.3 Benchmark Function Generator . . . 129
4.3.4 Performance Measures . . . 130
4.3.5 Control Methods . . . 131
4.3.6 Experimental Method . . . 132
4.4 Experimental Results . . . 135
4.4.1 Research Questions . . . 135
4.4.2 Research Question 1 . . . 136
4.4.3 Research Question 2 . . . 144
4.4.4 Research Question 3 . . . 156
4.5 Summary . . . 167

5 Performance Sensitivity of HMHH Under Varying Parameter Values 169
5.1 Introduction . . . 169
5.2 Experimental Procedure . . . 171
5.2.1 Experimental Method . . . 171
5.2.2 Hyper-heuristic Selection Operator Choices . . . 173
5.3 Experimental Results . . . 175
5.3.1 Research Questions . . . 175
5.3.2 Research Question 1 . . . 176
5.3.3 Research Question 2 . . . 188
5.3.4 Research Question 3 . . . 213
5.4 Summary . . . 222

6 Performance and Behavior Analysis of Selection Hyper-heuristics 224
6.1 Introduction . . . 224
6.2 Experimental Procedure . . . 225
6.2.1 Experimental Method . . . 226
6.3 Experimental Results . . . 227
6.3.1 Research Questions . . . 227
6.3.2 Research Question 1 . . . 228
6.3.3 Research Question 2 . . . 233
6.3.4 Research Question 3 . . . 243
6.3.5 Research Question 4 . . . 253
6.4 Summary . . . 297

7 Conclusions 300
7.1 Summary . . . 300
7.2 Achievement of Objectives . . . 305
7.3 Future Work . . . 306
7.3.1 (Self-)Adaptive Heuristic Pools . . . 306
7.3.2 Heuristic Allocation Behavior versus Performance Improvements . 308
7.3.3 Complementary Heuristics . . . 310
7.3.4 Comparative studies of P_r and existing DOP measures . . . 310

Bibliography 312

A Additional Results – Chapter 6 335

B Acronyms 381

C Symbols 383

D Derived Publications 388

List of Figures

2.1 Example of a unimodal objective function landscape in two dimensions. . 14
2.2 Spatial vs. temporal severity trade-off [53] 19
2.3 Moving peaks benchmark example in two dimensions 20

3.1 Hyper-heuristics classification as presented by Burke *et al.* [26] 76
3.2 Conceptual framework for selection hyper-heuristics 78

4.1 Illustration of the operation of P_r 113
4.2 Example of how P_r in two dimensions 116
4.3 Number of executions per algorithm group 134
4.4 Fitness scale ratios per environment type. 138
4.5 Relationship between P_r and the mean relative error, $P_{RE}(t)$ 143
4.6 P_r values for stand-alone heuristics in abrupt environments 146
4.7 P_r values for stand-alone heuristics in chaotic environments 147
4.8 P_r values for stand-alone heuristics in progressive environments 148
4.9 *ADSC* values of stand-alone heuristics in abrupt environments. 149
4.10 *ADSC* values of stand-alone heuristics in chaotic environments. 150
4.11 *ADSC* values of stand-alone heuristics in progressive environments. . . . 151
4.12 Sankey diagram of wins, draws, and losses for stand-alone heuristics . . . 154
4.13 P_r values for speciated heuristics in abrupt environments 157
4.14 P_r values for speciated heuristics in chaotic environments 158
4.15 P_r values for speciated heuristics in progressive environments 159
4.16 *ADSC* values of speciated heuristics in abrupt environments 160
4.17 *ADSC* values of speciated heuristics in chaotic environments 161
4.18 *ADSC* values of speciated heuristics in progressive environments 162

4.19 Sankey diagram of wins, draws, and losses for speciated heuristics 165

5.1 Focus of statistical tests for research question 1 177
5.2 Mean net wins per hyper-heuristic using the *Global_RT* configuration . . 180
5.3 Mean net wins per hyper-heuristic using the *Global_ST* configuration . . 181
5.4 Mean net wins per hyper-heuristic using the *Global_PT* configuration . . 182
5.5 Mean net wins per hyper-heuristic using the *island_RT* configuration . . 185
5.6 Mean net wins per hyper-heuristic using the *Island_ST* configuration . . 186
5.7 Mean net wins per hyper-heuristic using the *Island_PT* configuration . . 187
5.8 Focus of statistical tests for research question 2 190
5.9 Friedman ranks and net wins per hyper-heuristic across DOPs (1/7) . . . 192
5.10 Friedman ranks and net wins per hyper-heuristic across DOPs (2/7) . . . 193
5.11 Friedman ranks and net wins per hyper-heuristic across DOPs (3/7) . . . 194
5.12 Friedman ranks and net wins per hyper-heuristic across DOPs (4/7) . . . 195
5.13 Friedman ranks and net wins per hyper-heuristic across DOPs (5/7) . . . 196
5.14 Friedman ranks and net wins per hyper-heuristic across DOPs (6/7) . . . 197
5.15 Friedman ranks and net wins per hyper-heuristic across DOPs (7/7) . . . 198
5.16 Friedman ranks and net wins per hyper-heuristic across DOPs 206
5.17 Range of wins per hyper-heuristic across DOPs 209
5.18 Range of wins per hyper-heuristic across DOPs 210
5.19 Focus of the Wilcoxon signed ranks tests for research question 3 215
5.20 Pairwise wins and draws of *island* versus *global*, by hyper-heuristics . . . 216
5.21 Pairwise wins and draws of *island* versus *global*, by trigger 217
5.22 Pairwise wins and draws of *island* versus *global*, by DOP 218
5.23 Heat map of differences in wins between the *island* and *global* topology . 219

6.1 Heat map of win rates of hyper-heuristics versus stand-alone heuristics . 230
6.2 Heat map of draw rates of hyper-heuristics versus stand-alone heuristics . 231
6.3 Heat map of win rates of hyper-heuristics versus speciated heuristics . . . 237
6.4 Heat map of draw rates of hyper-heuristics versus speciated heuristics . . 238
6.5 Heat map of loss rates of hyper-heuristics versus speciated heuristics . . . 239
6.6 Heat map of wins achieved by all hyper-heuristics against each other . . 245
6.7 Wins and losses of hyper-heuristics , averaged across DOPs 246

6.8 Illustrative example of measuring $\mathcal{N}(t)$ and $\upsilon(t)$ . . . 256
6.9 Focus of the HSD and entity reassignment analysis . . . 258
6.10 HSD and entity reassignment rate analysis for **Rand** . . . 261
6.11 HSD and entity reassignment rate analysis for **RLFreq** . . . 267
6.12 Median $\mathcal{N}(t)$ versus δ values for **RLFreq** and RT trigger . . . 268
6.13 $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **RLFreq** for *Global_RT* . . . 269
6.14 $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **RLFreq** for *Island_RT* . . . 271
6.15 $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **RLFreq** for the PT trigger . . . 273
6.16 $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **RLFreq** for *Global_ST* . . . 275
6.17 Median $\mathcal{N}(t)$ versus δ values for **RLFreq** and ST trigger . . . 276
6.18 $\mathcal{N}(t)$ and $\upsilon(t)$ for **Rand**, **HTour2X**, and **HTour5X** for *Global_RT* . . . 278
6.19 HSD and entity reassignment rate analysis for **ETour2** . . . 279
6.20 HSD and entity reassignment rate analysis for **ETour25** . . . 280
6.21 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour2** for *Global_RT* . . . 283
6.22 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour25** for *Global_RT* . . . 284
6.23 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour2** for *Global_ST* . . . 285
6.24 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour25** for *Global_ST* . . . 286
6.25 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour2** for *Island_ST* . . . 287
6.26 $\mathcal{N}(t)$ and $\upsilon(t)$ for **ETour25** for *Island_ST* . . . 288
6.27 HSD and entity reassignment rate analysis for **Comp** . . . 290
6.28 $\mathcal{N}(t)$ and $\upsilon(t)$ for **Comp** for *Global_RT* . . . 292
6.29 $\mathcal{N}(t)$ and $\upsilon(t)$ for **Comp** for *Island_RT* . . . 293
6.30 $\mathcal{N}(t)$ and $\upsilon(t)$ for **Comp** for *Global_PT* . . . 295
6.31 $\mathcal{N}(t)$ and $\upsilon(t)$ for **Comp** for *Island_PT* . . . 296

A.1 HSD and entity reassignment rate analysis for **Perm** . . . 360
A.2 HSD and entity reassignment rate analysis for **Soft** . . . 361
A.3 HSD and entity reassignment rate analysis for **Comp** . . . 362
A.4 HSD and entity reassignment rate analysis for **DProp** . . . 363
A.5 HSD and entity reassignment rate analysis for **RLProp** . . . 364
A.6 HSD and entity reassignment rate analysis for **RLFreq** . . . 365
A.7 HSD and entity reassignment rate analysis for **ETour2** . . . 366

A.8 HSD and entity reassignment rate analysis for **ETour25** 367
A.9 HSD and entity reassignment rate analysis for **NAProp** 368
A.10 HSD and entity reassignment rate analysis for **AProp** 369
A.11 HSD and entity reassignment rate analysis for **NARank** 370
A.12 HSD and entity reassignment rate analysis for **ARank** 371
A.13 HSD and entity reassignment rate analysis for **HTour2X** 372
A.14 HSD and entity reassignment rate analysis for **HTour5X** 373
A.15 HSD and entity reassignment rate analysis for **HTour2M** 374
A.16 HSD and entity reassignment rate analysis for **HTour5M** 375
A.17 HSD and entity reassignment rate analysis for **Freq2** 376
A.18 HSD and entity reassignment rate analysis for **Freq5** 377
A.19 HSD and entity reassignment rate analysis for **RoulX** 378
A.20 HSD and entity reassignment rate analysis for **RoulM** 379
A.21 HSD and entity reassignment rate analysis for **Rand** 380

List of Algorithms

1 General outline of a local search algorithm [60] 15
2 General outline of an evolutionary algorithm [60] 16
3 General outline of a *gbest* particle swarm optimization algorithm [32] . . 17
4 Heterogeneous Meta-Hyper-Heuristic [76] 84
5 Charged particle swarm optimization algorithm. 123
6 Quantum particle swarm optimization algorithm, ... 124
7 Random Immigrant GA heuristic 126
8 Differential Evolution with adaptive scaling factor and crossover rate . . 128
9 Gaussian mutation-based heuristic 129

List of Tables

2.1 MPB Scenario 2 parameter settings 22
2.2 MPB parameters for each environment type 25

4.1 Percent non-normal distributions of $P_{RE}(t)$ values across change periods . 140
4.2 Percent non-normal distributions of P_r values across samples 141
4.3 Average ranks per environment for heuristics using the Friedman test . . 152
4.4 Wins, draws and losses per environment of each heuristic 153
4.5 Average Friedman ranks for speciated heuristics 163
4.6 Wins, draws and losses per environment for each speciated heuristic. . . . 164

5.1 Selection operator algorithms labels and parameters. 174
5.2 Correlation between k and net wins (per trigger) 179
5.3 Wins minus losses per hyper-heuristic and trigger 199
5.4 Correlation between k and ranks, and k and net wins 200
5.5 Friedman test p-values of wins minus losses, per hyper-heuristic 203
5.6 Significant wins per trigger, by hyper-heuristic 205

6.1 Wins, draws and losses of hyper-heuristics against stand-alone heuristics 234
6.2 Wins, draws and losses of hyper-heuristics against stand-alone heuristics 235
6.3 Wins, draws and losses of hyper-heuristics against speciated heuristics . . 241
6.4 Wins, draws and losses of hyper-heuristics against speciated heuristics . . 242

A.1 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *global* topology and RT trigger 336
A.2 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *global* topology and PT trigger 337

A.3 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *global* topology and ST trigger 338
A.4 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *island* topology and RT trigger 339
A.5 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *island* topology and PT trigger 340
A.6 Wins, draws and losses for each hyper-heuristic against the stand-alone heuristics, using the *island* topology and ST trigger 341
A.7 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *global* topology and RT trigger 342
A.8 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *global* topology and PT trigger 343
A.9 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *global* topology and ST trigger 344
A.10 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *island* topology and RT trigger 345
A.11 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *island* topology and PT trigger 346
A.12 Wins, draws and losses for each hyper-heuristic against the speciated heuristics, using the *island* topology and ST trigger 347
A.13 Friedman ranks and p-values per hyper-heuristic using the *global* topology and RT trigger 348
A.14 Wins, draws, and losses per hyper-heuristic using the *global* topology and RT trigger 349
A.15 Friedman ranks and p-values per hyper-heuristic using the *global* topology and PT trigger 350
A.16 Wins, draws, and losses per hyper-heuristic using the *global* topology and PT trigger 351
A.17 Friedman ranks and p-values per hyper-heuristic using the *global* topology and ST trigger 352
A.18 Wins, draws, and losses per hyper-heuristic using the *global* topology and ST trigger 353

A.19 Friedman ranks and p-values per hyper-heuristic using the *island* topology and RT trigger 354
A.20 Wins, draws, and losses per hyper-heuristic using the *island* topology and RT trigger 355
A.21 Friedman ranks and p-values per hyper-heuristic using the *island* topology and PT trigger 356
A.22 Wins, draws, and losses per hyper-heuristic using the *island* topology and PT trigger 357
A.23 Friedman ranks and p-values per hyper-heuristic using the *island* topology and ST trigger 358
A.24 Wins, draws, and losses per hyper-heuristic using the *island* topology and ST trigger 359

Chapter 1

Introduction

"The world as we have created it is a process of our thinking.
It cannot be changed without changing our thinking."
– Albert Einstein

1.1 Introduction

A dynamic optimization problem (DOP) is a special class of problem where optima change as time goes by. The complexity of a DOP is determined by the severity and frequency of changes, as well as the behavior of the values and trajectory of optima. Many efficient algorithms have been developed to solve these types of problems. Recent surveys show that a significant amount of Computational Intelligence (CI) research focuses on the meta-heuristic approaches of *evolutionary computation* (EC) and *swarm intelligence* (SI) to solving DOPs [40][93][119][136].

The surveys echo that different types of algorithms perform better in certain kinds of DOPs than in others, which is in line with what would be expected from the no free lunch theorem for optimization [184]. This presents a challenge to practitioners since it takes time to understand the nature of a given problem, and to identify a suitable algorithm to solve the problem. The wrong algorithm (or meta-heuristic parameter) choice can yield detrimental performance. Ideally, practitioners need an immediate "off the peg"

solution to a DOP instead of spending time developing more tailored approaches.

The fields of *Operations Research*, *Computer Science*, and *Artificial Intelligence* have produced complementary methods called *hyper-heuristics* that adapt the optimization process by choosing which low-level heuristic to apply to a problem over time [24]. A simpler definition is that selection hyper-heuristics are heuristics to choose heuristics [24]. What distinguishes hyper-heuristics from most other control adaptation approaches is the clear separation between solving a *problem* and searching for a suitable *method* (or *heuristic*) to solve a problem. Hyper-heuristics treat both the problem and heuristics as "black boxes" by not having detailed knowledge about the problem space, nor knowing exactly how heuristics solve a problem.

This thesis analyzes the ability of hyper-heuristic approaches to continuously select the most appropriate optimization method for a DOP as time goes by. Empirical studies examine both the performance of hyper-heuristics compared to the individual heuristics, as well as the behavioral differences between various hyper-heuristic approaches to better understand their mode of operation. Section 1.2 motivates the reasons for this research. The exact objectives of the research are provided in section 1.3. Section 1.4 lists the novel contributions made by this study. The scope of the research is outlined in section 1.5. Finally, section 1.6 gives an outline of the thesis as a whole.

1.2 Motivation

Hyper-heuristics continuously change the optimization algorithm as time goes by to better solve the problem under consideration. Motivations for hyper-heuristics include the ability to design a single method to handle *classes* of problems vs. just specific instances of problems, increased accuracy, better matching of optimization algorithms to problems (or problem stages), domain independence of algorithms, and more flexibility in the application of optimization algorithms to problems. Since DOPs change over time, a natural question arises whether hyper-heuristics are well-suited to change the optimization algorithm to better suit the current state of the problem *while the algorithm is solving the problem.*

A number of studies investigate the application of hyper-heuristics to dynamic environments [99][100][143][171][173][175][174]. Most of these studies focus on hyper-heuristics

managing simple heuristic operators (eg. Gaussian mutation operators). While these early studies show promising results, a natural extension to the application of hyper-heuristics to DOPs is to let hyper-heuristics manage a pool of heuristics that are purpose-built for DOPs.

This thesis will highlight a contradictory situation faced by DOP-focused meta-heuristics: static parameter tuning of DOP-specific meta-heuristic parameters is unattainable [46][97][103][**?**], while mixing together multiple self-adaptive parameter control methods in an ad hoc fashion is typically sub-optimal (the so-called *patchwork problem* [97]). A hyper-heuristic framework addresses this conundrum by separating the implementation of the common "building blocks" for DOP approaches discussed in various reviews focused on DOPs [40][93][119][136] into a *heuristic layer* and *hyper-heuristic layer*. The *heuristic layer* is responsible for *maintaining / introducing diversity* in the problem space, *reacting to change* of the environment, and *managing memory/state* in the problem space. The *hyper-heuristic* layer is responsible for managing *multiple populations* of candidate solutions that focus on different aspects of the search, *maintaining a memory* of correlations between heuristic allocation, feedback values, and resulting performance, and *self-adaptation* of search behavior by learning how to best allocate computational resources to the most promising heuristics at time t.

Birattari *et al.* [11] assert that meta-heuristics are, essentially, templates for an operation that only becomes a concrete algorithm once parameter values are selected. Deliberate parameter value choices generally produce very specific *exploration* and *exploitation* behavior in a meta-heuristic [57][58]. A hyper-heuristic framework encapsulates each heuristic as a distinct collection of low-level operators, design decisions, and parameter values. The result is a mix of highly varied modes of operation contained in a pool of heuristics. For example, an SI approach comprising of specific *neighborhood structures*, *electrostatic charges*, *position update* operators, and *velocity update* operators will exhibit completely different behavior than an EC approach with *cross-over*, *mutation*, *replacement*, and *elitism* operators.

This thesis investigates how well various hyper-heuristic selection operators can continually balance computational resources across different population-based meta-heuristics that are specialized to solving DOPs. The goal is to determine if such a hyper-heuristic system can solve a DOP better than the individual meta-heuristics can. The

ultimate purpose is not to find the best algorithm to solve DOPs, nor to compare hyper-heuristics with state-of-the-art DOP algorithms, nor to exhaustively determine the best factors that characterize a good heuristic pool. The aim is to better understand what heuristic allocation behavior leads to improved performance.

1.3 Objectives

The main goal of this thesis is to investigate how well various hyper-heuristic selection operators can continually balance computational resources across different population-based meta-heuristics in order to solve a DOP better than the individual meta-heuristics can. Within this broad objective the following sub-objectives will also be investigated:

- Investigate existing hyper-heuristic algorithms, frameworks, and approaches that aim to solve DOPs, specifically focusing on selection hyper-heuristic and population-based approaches.
- Define suitable parameters for the well-known *moving peaks benchmark* [19] that allow the creation of 27 unique types of DOPs, as defined by the unified DOP classification of Duhain and Engelbrecht [53], and are compliant with scenario 2 of the moving peaks benchmark function generator [19][127].
- Investigate the effectiveness of the *Heterogeneous Meta-hyper-heuristic* (HMHH) [75][76][74], which was originally intended for static environments, to solve DOPs.
- Extend HMHH with the different neighborhood topologies, and subsequently compare and contrast the performance of various hyper-heuristics when using each topology.
- Extend HMHH with multiple types of heuristic change triggers that are used to decide when a new heuristic should be applied to a particular candidate solution, and investigate the effect different types of triggers have on the performance of various hyper-heuristics.
- Propose a new performance measure for DOP-focused algorithms that improves upon existing measures by more faithfully representing the holistic performance of

an algorithm over time.

- Perform a sensitivity analysis on the shared parameter for heuristic change triggers, namely k, which controls the frequency of heuristic changes occurring. Compare and contrast the performance of each hyper-heuristic to determine which combinations of hyper-heuristics and trigger types are more sensitive than others.
- Compare the performance of various well-known hyper-heuristics against each other, as well as against various control groups.
- Investigate the heuristic space diversity and heuristic reassignment rates of various hyper-heuristics to better understand their modes of operation.

The purpose is not to find the best algorithm to solve DOPs, nor to compare hyper-heuristics with state-of-the-art DOP algorithms, nor to exhaustively determine the best factors that characterize a good heuristic pool.

1.4 Contributions

The following novel contributions are made by this thesis:

1. This is the first study to analyze the performance and behavior of a broad range of hyper-heuristic selection operators across the full spectrum of real-valued DOP types. Unique environments are systematically created with parameter values that are compliant with scenario 2 of the moving peaks benchmark function generator [19][127], and based on the holistic classification of Duhain and Engelbrecht [53] (which unites the well-known classifications of Eberhart and Shi [56] and Angeline [4] together with spatial and temporal change severity classes).
2. A contradictory situation faced by DOP-focused meta-heuristics is identified: statically tuning meta-heuristic parameters is impossible in DOPs [46][97][103][**?**], yet dynamically adapting multiple meta-heuristic parameters in an ad hoc fashion using state-of-the-art self-adaptive parameter control methods produces poor results (this has been framed as the so-called *'patchwork problem'* [97]). This thesis shows

how selection hyper-heuristics for population-based meta-heuristics can address this problem practically.

3. The HMHH by Grobler *et al.* [75][76][74] was originally devised for static environments. This thesis extends the HMHH framework by establishing the criteria needed to identify appropriate meta-heuristics to enable HMHH to solve DOPs, introduces *global* and *island* neighborhoods that govern heuristic visibility of the population of candidate solutions, and introduces different heuristic selection triggering mechanisms for HMHH beyond relying only on periodic time-based triggers.

4. The HMHH framework is extended to handle a mix of population-based meta-heuristics and single-solution methods under the same population-based paradigm. This allows single-point search heuristics, commonly used in the hyper-heuristic literature, such as a Gaussian mutation operators to be used in conjunction with complex meta-heuristics from the SI and EC fields.

5. A new performance measure for DOPs, namely the *relative error distance* (or P_r, as calculated using equation (4.1)), is proposed that does not assume normally distributed performance data across an algorithm run, is resilient against fitness landscape scale changes, better incorporates performance variance across multiple fitness landscape changes, and allows easier algorithm comparisons using established nonparametric statistical methods.

6. Metrics capturing heuristic allocation behavior, heuristic space diversity, and allocation stability are used to understand *why* and *how* hyper-heuristic selection operators differ in their behavior. A new measure of heuristic diversity in population-based meta-heuristics, namely $\mathcal{N}(t)$, as calculated using equation (3.2), is introduced that is derived from Shannon's normalized entropy measure [160].

7. Together with the newly proposed heuristic space diversity measure, $\mathcal{N}(t)$ (as calculated using equation (3.2)), guidelines are provided to better understand the heuristic space diversity behavior of a given hyper-heuristic. New measures are proposed that consider the heuristic reassignment frequency, or δ (as calculated using equation (6.2)), and heuristic reassignment volume, or φ (as calculated using

equation (6.3)), of a multi-population-based hyper-heuristic over the entire course of an algorithm run.

8. The experimental procedures for all algorithm evaluations, comparisons, and sensitivity analysis rely entirely on nonparametric statistical methods. The analysis method and results in this thesis constitute the first study to apply nonparametric statistical analysis procedures to analyze the performance of selection hyper-heuristics for populations-based meta-heuristics. This analysis includes using a performance measure that itself does not assume a normal distribution of the underlying telemetry data (i.e., the newly proposed P_r measure).

9. The results show that a complex relationship exists between the heuristic selection logic, the frequency of heuristic changes, the change triggering logic, the HMHH neighborhood topology, and the exact type of DOP being solved. However, every combination of these components yields performance that is superior to using any of the individual DOP-specific heuristics in isolation. A general correlation is also visible between increased performance and more frequent heuristic selection. Lastly, even under a random trigger, more frequent heuristic changes across a portion of the population is frequently the most beneficial approach, implying that simple algorithmic diversity over time is effective at improving performance in DOPs.

1.5 Scope

The scope of this thesis is as follows:

- Dynamic environments with dynamically changing dimensions are not in the scope of this thesis.
- Environment change detection is out of scope to avoid introducing undue bias. All changes are introduced in the same controlled manner and all algorithms are informed of all changes in a uniform fashion.
- No equality or inequality constraints are present in any of the testing benchmark functions. Only standard boundary constraints of the function domain are used.

- Dynamic multi-objective optimization problems (DMOPS) are out of scope of the thesis. Only single-objective problems are considered.

- No truly *self-adaptive* heuristics that employ learning are used. Self-adaptive heuristics could result in heuristics converging (in heuristic space) on similar types of behavior. Many heuristics would exhibit no noticeable difference in behavior, which would impede the ability of the HMHH framework to alter how the search is conducted. Additionally, many HMHH selection operators employ learning and memory mechanisms to learn which heuristic's behavior is appropriate given specific environmental feedback. Self-adaptive heuristics would continually change their behavior over time, causing the hyper-heuristic's knowledge of the workings of each heuristic to become outdated. Future studies should investigate the trade-off and synergies between these two levels of self-adaptation, and be mindful of not exasperating the *patchwork problem* by carefully managing which parameters are adapted, and in what manner.

- While HMHH can technically utilize heuristics that manage sub-populations of entities, this study is limited to heuristics with single populations. Multi-population heuristics would result in a fragmented 'sub-populations of sub-populations' situation, which would increase the number of entities required for HMHH to be effective (particularly in the island neighborhood topology). The use of sub-populations by heuristics would greatly diminish any gains in computational efficiency HMHH would bring. Additionally, heuristic state management is simpler in a single-population heuristic approach when entity reallocation occurs.

- Nguyen *et al.* [136] highlight a number of studies that show how the use of environment change prediction could negatively impact the optimization method. The wrong training data, lack of training data, or the very nature of the DOP could all lead to extremely poor performance by wrongly biasing the search to certain areas. The goal of this study is to investigate a broadly applicable hyper-heuristic approach and not to specialize the method to a subset of DOP types that exhibit predictable behavior. For similar reasons, any explicit memory schemes that track good solutions and the problem space conditions that led to them are out of scope.

- All hyper-heuristics share the same pool of heuristics, because testing the effectiveness of each trigger under varying heuristic pool compositions is out of scope for this thesis, and is left for future studies.

1.6 Thesis Outline

The thesis is structured as follows:

Chapter 2 provides a brief overview of optimization theory and defines DOPs. The main algorithmic components of successful algorithms for DOPs are reviewed. An overview of well-known SI and EC meta-heuristics from literature is given. The moving peaks benchmark function generator is discussed and extended and well-known performance and behavior measures for DOPs from literature are presented. Lastly, a discussion is provided regarding why nonparametric statistical procedures are essential to correctly compare the performance of CI algorithms, along with an overview of well-established and recommended methods and procedures.

Chapter 3 discusses the need for adaptation of algorithms when solving DOPs. An overview is given of the static parameter tuning problem and, its dynamic equivalent, the parameter control problem. A breakdown is provided of the applicability of both tuning paradigms to DOPs, and why hyper-heuristics are an important class of adaptive control methods. The *no free lunch* (NFL) theorems for optimization are discussed, along with reasoning from literature as to why the conditions that are required for the NFL theorem to hold are practically impossible to find in real-world continuous-valued optimization problems. Hyper-heuristics are positioned as meta-search methods that can therefor raise performance above that of any of the individual heuristics alone. A full literature review is provided of hyper-heuristics as a field, along with a classification of hyper-heuristic approaches and an overview of the notions of problem space versus heuristic space. A detailed overview of selection hyper-heuristics for population-based meta-heuristics is given along with overviews of heuristic change trigger types, neighborhood topologies, and how heuristics managed by a hyper-heuristic need to complement each other. Lastly, control adaptation approaches that are related to hyper-heuristics are reviewed.

Chapter 4 establishes the performance of various control methods, which grounds the analysis of all investigated hyper-heuristics against an objective set of baselines. The

control methods alleviate concerns around whether or not any increased performance by hyper-heuristics is due to intelligent selection, or simply due the use of multiple sub-populations, multiple methods, or random heuristic assignments. The approach described serves as the foundation for all experimental work in the rest of the thesis. A new performance measure for DOPs is proposed that more faithfully reports the sustained average performance of an algorithm over time while also better capturing the variance in performance over time.

Chapter 5 explores the effect that various mechanisms to trigger heuristic change combined with different neighborhood topologies have on the performance of selection operators in the HMHH framework. The frequency parameter of each trigger is systematically varied to ascertain the effect, if any, that each trigger has on performance. This process is repeated in 27 different types of DOPs across a wide variety of hyper-heuristics, as detailed in section 5.2.2. The entire procedure is repeated across two HMHH neighborhood topologies and contrasted against each other.

Chapter 6 compares and analyzes the performance of various HMHH selection operators against each other and the control group baseline established in chapter 4. The aims are to determine whether each hyper-heuristic raises performance above that of the individual heuristics or speciated heuristics, to establish whether each hyper-heuristic performs better or worse than the fixed or random selection control methods, and to understand which hyper-heuristics perform better than other hyper-heuristics. The heuristic space diversity and entity reassignment behavior of a variety of exemplary HMHH selection operators are investigated to characterize their behavior and performance.

Chapter 7 concludes the thesis with a summary of findings and proposed areas of future research.

A number of appendices are provided. Appendix A provides additional results relevant to chapter 6. Appendix B provides a list of acronyms used in this thesis, while appendix C contains a list of symbols that are used often. Appendix D provides a list of accepted and under review publications that resulted from this study.

Chapter 2

Dynamic Optimization

"Yesterday is gone. Tomorrow has not yet come. We have only today. Let us begin."

– Mother Theresa

This chapter presents a working definition of a *dynamic optimization problem* (DOP) as used in this thesis, along with an overview of different classifications of DOPs. An outline of the *moving peaks benchmark* function generator as a well-known DOP benchmark function generator from literature is presented, and moving peaks benchmark parameter settings are provided to simulate specific types of DOPs. The challenges faced by optimization algorithms for DOPs are discussed, followed by an overview of well-known SI and EC methods to solve DOPs. Various performance measures to evaluate and compare the performance of optimization algorithms in DOPs are also presented.

2.1 Introduction

DOPs are challenging since the state (or search space) of the optimization problem changes over time. As time passes, optimal solutions may become sub-optimal while previously inferior regions of the search space may suddenly yield the best solutions. The dynamic nature of an ever-changing problem landscape means that most optimization

algorithms focused on solving problems in static environments tend to be ineffective [40][60][93][119][136]. Section 2.2 provides an overview of optimization problems overall and how CI optimization algorithms are designed to solve them. Section 2.3 discusses how DOPs are different than their static counterparts, and provides a classification of various types of dynamic environments.

Researchers use controlled benchmark functions to simulate dynamic environments and to measure the effectiveness of new and existing optimization algorithms. Section 2.4 provides an overview of the most well-known and widely used DOP benchmark function generator called the moving peaks benchmark by Branke [19]. Extensions to the moving peaks benchmark are discussed that enable a fuller spectrum of different dynamic environment types. Criticisms of the moving peaks benchmark are also discussed.

Section 2.5 discusses key strategies (or "*building blocks*" of functionality) that can be found in the majority of successful algorithms developed for solving DOPs [40][93][119][136]. Section 2.6 gives an overview of a number of well-known DOP-focused population-based meta-heuristics from the SI and EC fields.

Section 2.7 outlines a number of performance and behavioral measures that are often used by researchers to characterize how well a DOP-focused algorithm performs and behaves. A discussion on valid and invalid statistical comparison procedures is provided pertaining to the comparison of performance between any given pair of algorithms, comparing one algorithm to many other algorithms, and comparing all algorithms to each other. Finally, section 2.8 concludes the chapter.

2.2 Optimization Problems

Mathematical optimization is the process of finding the combination(s) of input value(s) in a search space of possible input values that yield "optimal" output values of some function f. Optimality is defined relative to an optimization goal such as either maximizing or minimizing the output of f. The *objective function* f coupled with an optimization goal together constitutes an *optimization problem*.

Optimization problems can be classified according to the number and type of objective function input variables, the presence of equality and/or inequality constraints in the search space of input variables, the degree of linearity of the objective function f, the

continuity or discontinuity properties of f, the convexity of f, the number of optima in f and whether they change, and the number of optimization criteria [60]. An optimization problem with an n_x-dimensional real-valued input domain, $\mathbf{x} \in \mathbb{R}^{n_x}$, can be stated as

$$\text{maximize} \quad f(\mathbf{x}), \qquad \text{subject to } \mathbf{x} \in \mathbb{R}^{n_x}$$

The goal of an *optimization algorithm* is to find a *global optimum* point, $\mathbf{x}^* \in \mathbb{R}^{n_x}$, that satisfies the optimality principle,

$$f(\mathbf{x}^*) \geq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^{n_x}$$

for maximization problems. The objective can also be stated as a minimization problem in which case the optimization criterion becomes

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^{n_x}$$

In other words, the point $\mathbf{x}^*$ is that point in the domain $\mathbb{R}^{n_x}$ of f that yields the maximum (or minimum) value of f, where f represents a function to be optimized. The function f is also called the *objective function*. Examples of real-world optimization problems include maximizing profit based on targeted marketing activities, minimizing the risk of an investment portfolio, or minimizing production costs based on flexible product design parameters. Figure 2.1 illustrates the search landscape of a 2-dimensional objective function. Maximizing f entails finding those input values in the two input dimensions that yield the output point corresponding to the top of the cone.

Most real-world optimization problems have function landscapes that are completely unknown. The function f is, for intents and purposes, a "black box" with unknown behavior and characteristics. Practitioners could, potentially, use function sampling techniques to explore f in an attempt to locate the set(s) of input values that yield the most optimal output values. Obtaining sufficient samples this way is not always efficient (i.e. if function evaluation has a high computational or economic cost), and may be ineffective at finding good optima. The sampling process may even be practically intractable if, for example, the domain of f has a high number of dimensions or the function landscape complexity is too high.

Instead of using simpler function sampling techniques, most practitioners rely on *optimization algorithms* to locate global optima. Optimization algorithms search for a

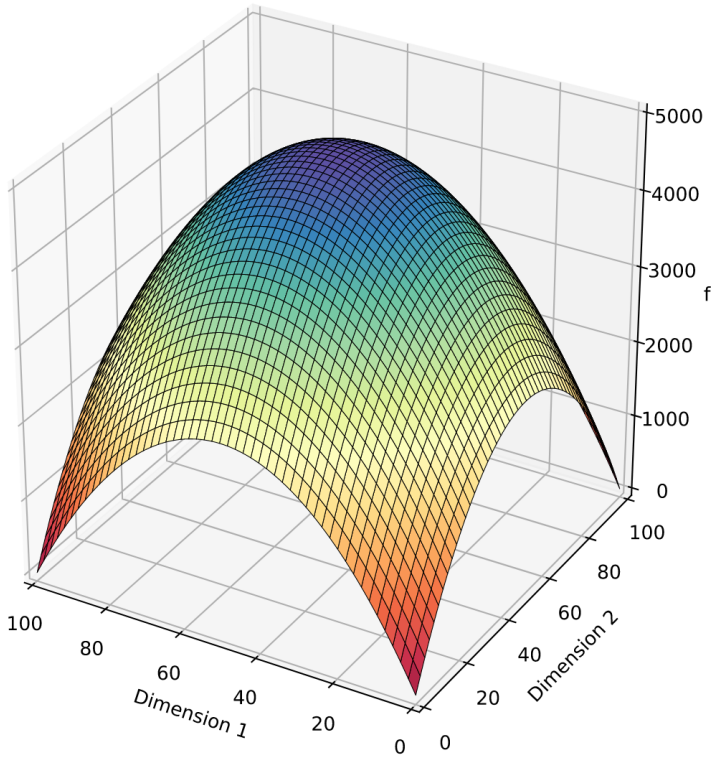


Figure 2.1: Example of a unimodal objective function landscape in two dimensions.

global optimum point $\mathbf{x}^*$ of f by intelligently updating a *candidate solution*, $\mathbf{x} \in \mathbb{R}^{n_x}$, in an iterative manner. Since the function f is unknown, the optimization algorithm has to rely on sampled domain knowledge gathered during the optimization process to intelligently modify $\mathbf{x}$. Over time, the aim of the optimization algorithm is to have $\mathbf{x}$ *converge to* (or approach) a global optimum point $\mathbf{x}^*$.

Optimization algorithms fall into two broad categories, namely *local* search algorithms and *global* search algorithms [60]. Local search algorithms rely only on information that is available in the neighborhood (or physical locality) of the candidate solution, while global search algorithms have the ability to explore the entire function search landscape. A general template of a local search algorithm (taken from Engelbrecht [60]) is outlined in algorithm 1. Different methods of choosing a search direction vector $\mathbf{q}(t)$ and step length scalar $l(t)$ result in different types of local search algorithms.

Global optimization algorithms take a plethora of different forms. The field of CI

outlines approaches that are inspired by natural systems. The sub-fields of CI are SI, EC, *artificial neural networks*, *fuzzy systems*, and *artificial immune systems* [60]. SI algorithms mimic the emergent behavior of populations of animals such as fish, ants, birds, and others to systematically explore the search landscape of an optimization problem. EC approaches draw inspiration from natural evolutionary processes to systematically *evolve* improving solutions to an optimization problem over multiple generations. Algorithm listing 2 shows the general outline of an evolutionary algorithm from the EC field, and algorithm listing 3 shows the general *particle swarm optimization* (PSO) from the SI field (the algorithm outlines are taken from Engelbrecht [60] and Cleghorn and Engelbrecht [32], respectively).

Algorithm 1 General outline of a local search algorithm [60]

Let $t = 0$ count the iterations
Let $\mathbf{x}(0) \in \mathbb{R}^{n_x}$
while a stopping condition is not met **do**
    Evaluate $f(\mathbf{x}(t))$
    Determine a search direction, $\mathbf{q}(t)$
    Determine a step length, $l(t)$
    $\mathbf{x}(t+1) = \mathbf{x}(t) + l(t)\mathbf{q}(t)$
    $t = t + 1$
end while
Return $\mathbf{x}$(t) as the most optimal solution

2.3 Optimization in Dynamic Environments

A DOP is a special class of optimization problem where the search landscape of the objective function f changes as time goes by while the problem is being solved by an optimization algorithm [136]. DOPs present a challenge to optimization algorithms in that the optima of f not only have to be located, but also have to be tracked as the environment changes. The complexity of a DOP is determined by the severity and frequency of landscape changes, as well as the behavior of the values and trajectory of

Algorithm 2 General outline of an evolutionary algorithm [60]

Let $t = 0$ count the generations
Initialize an n_x-dimensional population $\mathcal{C}(0)$ of n_s individuals
while a stopping condition is not met **do**
  Evaluate $f(\mathbf{x}_i(t))$ of each individual i in the population $\mathcal{C}(t)$
  Perform reproduction between parents to create offspring
  Select a new population $\mathcal{C}(t+1)$
  $t = t + 1$
end while
Return the most fit $\mathbf{x}$(t) from $\mathcal{C}(t)$ as the optimal solution

optima. Different types of DOPs show unique patterns in optima trajectory, search space composition (i.e. the presence of a single versus multiple base functions), homogeneity of optima movement, change pervasiveness among optima, the hardness of the search space, and even changes in the number of dimensions of f as well as the presence of dynamically changing constraints [40][53][93][119][136].

In the context of this thesis, a DOP refers to a real-valued optimization problem with static boundary constraints and a fixed number of dimensions, defined as

$$\text{maximize} \quad f(\mathbf{x}, \boldsymbol{\omega}(t)), \quad \mathbf{x} \in \mathbb{R}^{n_x}$$

where f is a time-dependent objective function and $\boldsymbol{\omega}(t) = (\omega_1(t), \ldots, \omega_{n_\omega}(t))$ are time-dependent control parameters of f [60]. Solving $f(\mathbf{x}, \boldsymbol{\omega}(t))$ means finding a global optimum point $\mathbf{x}^* \in \mathbb{R}^{n_x}$ of the search landscape at time t, namely $f(\mathbf{x}^*, \boldsymbol{\omega}(t)) \geq f(\mathbf{x}, \boldsymbol{\omega}(t)), \forall \mathbf{x} \in \mathbb{R}^{n_x}$ (for maximization).

This definition of a DOP is general enough to allow this thesis to focus on DOPs that (as per Nguyen *et al.* [136] and Cruz *et al.* [40]): have a single objective, mixes predictable and unpredictable change patterns, have visible changes, i.e. no detection strategies are needed, are unconstrained, and do not have explicit time-linkage between states that depend on algorithm interactions with the environment.

Algorithm 3 General outline of a *gbest* particle swarm optimization algorithm [32]

Initialize an n_x-dimensional swarm $\mathcal{C}(0)$ of n_s particles
Let f be the optimization function (maximization here)
Let $\mathbf{y}_i$ be the personal best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\hat{\mathbf{y}}_i$ be the neighborhood best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\mathbf{v}_i$ be the velocity vector of particle i (initialized to $\mathbf{0}$)
repeat
  for each particle $i = 1, \ldots, n_s$ **do**
    //Set the personal best position, $\mathbf{y}_i$, of each particle i
    if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
      $\mathbf{y}_i = \mathbf{x}_i$
    end if
    //Set the neighborhood best position, $\hat{\mathbf{y}}_i$, of each particle i
    for each particle j containing particle i in their neighborhood **do**
      if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}}_j)$ **then**
        $\hat{\mathbf{y}}_j = \mathbf{y}_i$
      end if
    end for
  end for
  for each particle $i = 1, \ldots, n_s$ **do**
    Update the velocity of particle i
    Update the position of particle i
  end for
until a stopping condition is met
Return the global best solution as the most optimal solution

2.3.1 Classifications of Dynamic Optimization Problems

Dynamic environments differ in their frequency and severity of changes, patterns in optima trajectory, search space composition (i.e. single or multiple base functions), homogeneity of optima movement, change pervasiveness among optima, and the hardness of the search space [52][53]. Many classification systems have been devised to categorize

DOPs. Eberhart *et al.* [56][84] define three types of dynamic environments based on the direction of change of the optima:

- *Type I* environments, where optima locations change but not their values, i.e. the input variables that constitute the new best optimum of f change but the magnitude of the output value of f at the new optimum remains unchanged.
- *Type II* environments, where optima values change but not their positions, i.e. the input variables of f at the optimum remain unchanged, but the magnitude of the output of f at the optimum changes.
- *Type III* environments where both the optima values and positions change, i.e both the input variables of f at the optimum point as well as the magnitude of the value of f at the optimum change.

Angeline [4] classifies dynamic environments based on the *trajectory* of optima over multiple changes. The three types of dynamic environments include linear, circular, and random environments. Linear trajectories imply that the input values of the optima of f change in a linear relationship to each other over time. Visually, the "peaks" of the search landscape appear to move in a straight line. Similarly, circular and random trajectories imply that the input variables of the optima of f respectively have a circular change pattern or uncorrelated change pattern where the peaks appear to move either circularly or randomly.

Duhain and Engelbrecht [53] consider all dimensions in each of the the classifications of Angeline, Eberhart *et al.*, Weicker [180][181][182], and De Jong [45]. They argue that the degree of spatial and temporal severity of environment changes should be considered together, and propose the following behavioral classes:

- **Static** and **quasi-static** environments have spatial and temporal change severities that are either null or insignificant (relative to the scale of the DOP). The DOP is, essentially, a static optimization problem.
- **Abrupt** environments have large landscape changes that do not occur often. The function landscape tends to be stable for a duration of time, followed by spatial changes that modify the optima locations and/or values in a significant manner.

- **Progressive** environments have very frequent landscape changes, but changes tend to be small. Over time the function landscape shape can differ significantly, but the alterations occur gradually over multiple environment change events.

- **Chaotic** environments have relatively large landscape changes that occur frequently. The function landscape is modified in a significant manner during each environment change event, and these types of changes occur often.

Duhain and Engelbrecht subsequently combine the classifications of Eberhart *et al.* and Angeline with the spatial and temporal change severity behavior classes outlined above into 27 unique DOP types. Figure 2.2 illustrates the relationships between the change severity classes.

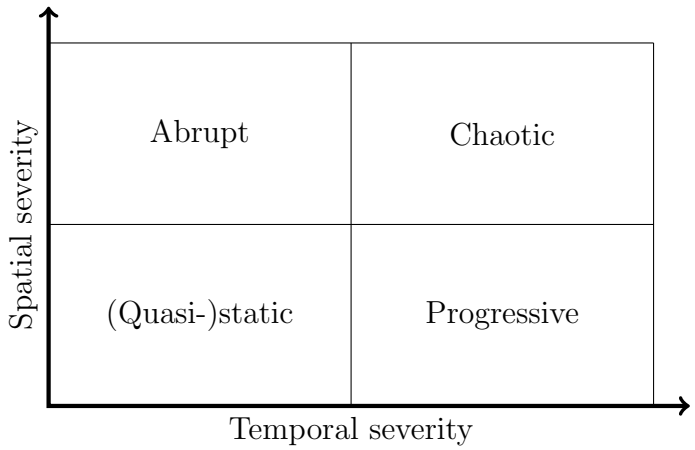


Figure 2.2: Spatial vs. temporal severity trade-off [53]

In this thesis, the 3-tuple notation (X, Y, Z) encodes the 27 environment types of Duhain and Engelbrecht, where $X \in \{\text{A, P, C}\}$ indicates if the environment change is *abrupt*, *progressive* or *chaotic*, $Y \in \{1, 2, 3\}$ indicates Eberhart *et al.*'s classification into Type I, II or III, and $Z \in \{\text{L, C, R}\}$ indicates Angeline's classification into *linear*, *circular* or *random* peak trajectories. Duhain and Engelbrecht [53] provide detailed parameter selection guidelines to define 27 uniquely different types of DOPs using the moving peaks benchmark, which will be discussed next.

2.4 The Moving Peaks Benchmark

The *moving peaks benchmark* (MPB) by Branke [19] is a function maximization problem that consists of an n_x-dimensional search landscape made up of multiple peaks (or optima). Each peak has independent height, width, location and movement dynamics. Over time all peaks move around within the defined boundaries of the search space. The search landscape of the MPB for two dimensions is illustrated in figure 2.3.

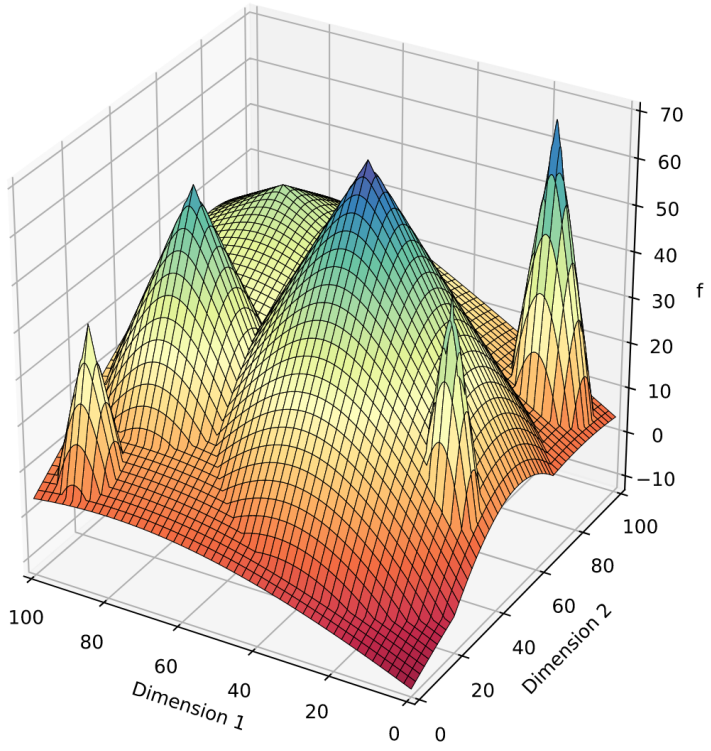


Figure 2.3: Example of the moving peaks benchmark function landscape in two dimensions. There are six peaks of varying height and width.

The MPB benchmark function generator has the following parameters that can be set to yield different types of DOPs:

- Dimension of the problem.
- The number of peaks.
- A peak shape function, such as a cone or a spherical shape.

- The minimum and maximum allowed peak height.
- The minimum and maximum allowed peak width.
- The frequency of changes of peaks (expressed in terms of the number of function evaluations).
- A change severity factor that stipulates the distance that peaks move during each change.
- The correlation factor between previous peak movements and new movements.
- The severity by which peak heights and widths are altered between changes.

The function value of the MPB, f, is the maximum value across all of the individual peak functions f_p, i.e.

$$f(\mathbf{x}, t) = \max\{f_b(\mathbf{x}), \max_{p=1,\dots,n_p}\{f_p(\mathbf{x}, h_p(t), w_p(t), \mathbf{l}_p(t))\}\} \tag{2.1}$$

where f is the MPB function, f_b is the basis function landscape (zero in this thesis), n_p is the number of peaks, and f_p defines each peak p with height h_p, width w_p, and location of the peak top $\mathbf{l}_p \in \mathbb{R}^{n_x}$ at time t. In this study f_p is a cone function defined as

$$f_p(\mathbf{x}, h_p(t), w_p(t), \mathbf{l}_p(t)) = h_p(t) - w_p(t)\sqrt{\sum_{j=1}^{n_x}(l_{p,j}(t) - x_j)^2} \tag{2.2}$$

where n_x is the dimension of $\mathbf{x}$. Peak heights and widths are modified as follows:

$$h_p(t) = h_p(t-1) + h_s\sigma(t)$$

$$w_p(t) = w_p(t-1) + w_s\sigma(t)$$

where $\sigma(t) \sim N(0,1)$, and h_s and w_s are the height severity and width severity respectively. The MPB shift vector $\mathbf{s}_v(t)$ is

$$\mathbf{s}_v(t) = \frac{s}{\|\mathbf{p}_r + \mathbf{s}_v(t-1)\|}((1-\lambda)\mathbf{p}_r + \lambda\mathbf{s}_v(t-1)) \tag{2.3}$$

where $\mathbf{p}_r$ is a random vector normalized to length s (the spatial severity), and λ controls the correlation of peak directions between changes (0.0 indicates no correlation while 1.0 means the peak moves in a straight line).

The MPB is still the most often-used benchmark function generator for real-valued DOPs [40][93][119][127][136]. Moser and Chiong [127] review a wide range of studies that use the MPB to test the performance of various EC, SI, and hybrid approaches. Moser and Chiong note that most of the studies in literature use parameter value ranges that are consistent with the original scenario 2 proposed by Branke due to the appropriateness of the resulting problem difficulty and solvability [21][127]. The typical settings of the MPB scenario 2 are listed in table 2.1

Table 2.1: MPB Scenario 2 parameter settings

Parameter	**Value**
Peaks	10
Dimensions (n_x)	5
Change period	5000
Height severity (h_s)	7.0
Width severity (w_s)	1.0
Change severity (s)	1.0
Peak function	Cone
Peak height (h_p)	$h_p \in [30, 70]$
Peak width (w_p)	$w_p \in [1, 12]$
Correlation (λ)	$\lambda \in 0, 1$
Peaks	10
Function domain	$(0, 100)^{n_x} \subset \mathbb{R}^{n_x}$
Constraints	*unconstrained*

2.4.1 Extending the Moving Peaks Benchmark

The classification of Duhain and Engelbrecht [53] discussed in section 2.3.1 defines 27 distinctly different types of DOPs. Duhain and Engelbrecht provide parameter considerations to carefully craft each environment using the MPB:

- **Type I**: $h_s = 0$ and $s \neq 0$
- **Type II**: $h_s \neq 0$ and $s = 0$

- **Type III**: $h_s \neq 0$ and $s \neq 0$
- **Linear**: $\lambda = 1$ and $s \neq 0$
- **Circular**: $s = 0$ and the function is rotated on its center
- **Random**: $\lambda = 0$ and $s \neq 0$
- **Progressive**: low h_s and w_s relative to the domain, very high rate of function landscape change
- **Abrupt**: high h_s and w_s relative to the domain, low rate of function landscape change
- **Chaotic**: high h_s and w_s relative to the domain, high rate of function landscape change

Combining each of these nine guidelines for each of the three classification components of the 3-tuple notation (X, Y, Z) from section 2.3.1 yields the 27 different types of environments according to the unified classification of Duhain and Engelbrecht [53].

The majority of the resulting environments are relatively straightforward to construct using the original MPB parameters alone. However, a number of the 27 environment types require additional parameters and functionality. Table 2.2 shows the MPB parameter values that yield 27 unique environments using the parameter considerations of Duhain and Engelbrecht, where h_s is the height severity, w_s is the width severity, s is the spatial severity, λ controls the randomness of a peak's trajectory, M_i is the number of iterations before a function landscape change, M_C is the cycle length of a full function landscape rotation (discussed below), and M_ϕ indicates the growing/shrinking behavior pattern of each peak (as discussed below). h_s is zero for Type I environments (where peak heights remain the same), M_C is only defined in circular type I and type III environments, and s is zero for Type II environments (where peaks never move). Table 2.2 shows the resulting parameters shared by each environment, namely peak height h_p, and peak width w_p, function domain, dimensions, constraints, and number of peaks. The additional parameters, M_C and M_ϕ, and how to set them to yield the remaining MPB environments are discussed below.

For Type I and III circular environments, n_x-dimensional function rotation is used to rotate the peaks in a cyclic fashion. The rotation matrix $R_{ab}(\theta)$ rotates axis a, θ degrees towards axis b as follows:

$$R_{ab}(\theta) = \begin{cases} r_{ii} & = 1 \text{ where } i \neq a, i \neq b \\ r_{aa} & = \cos(\theta) \\ r_{bb} & = \cos(\theta) \\ r_{ab} & = -\sin(\theta) \\ r_{ba} & = \sin(\theta) \\ r_{ij} & = 0 \text{ otherwise} \end{cases} \tag{2.4}$$

where r_{ij} are the entries in the proper rotation matrix [2]. A single simple rotation around a hyper-plane is used to ensure that repeated rotations through 2π degrees will again yield the original landscape. The same hyper-plane is used for all rotations during a single simulation run, but a new random hyper-plane is selected for different instances of the problem. Variable rotation cycle lengths are used to control the change severity instead of using the MPB shift vector or spatial severity s. To ensure that peaks generally move the same distance in the search landscape as stipulated by s in other corresponding MPB environments, the cycle length M_C can be set to the appropriate magnitude to mimic the effect of s in the standard MPB landscape. For example, given a function domain of $(0, 100)^{n_x} \subset \mathbb{R}^{n_x}$, a rotation cycle length of $M_C = 314$ ensures that points a distance of $r = 50$ from the domain center $(50, 50)$ only change position by ${}^{2\pi r}/_{314} \approx 1$, which is close to $s = 1$ (i.e. for progressive environments). Similarly, $M_C = 62$ results in ${}^{2\pi r}/_{105} \approx 3$, which is close to $s = 3$ (i.e. for abrupt and chaotic environments). These values for M_C mimic the spatial severity change step sizes for each environment to be similar to what s would have given.

In Type II environments optima maintain their positions in the search space while optima values change. Angeline's [4] *linear*, *circular* and *random* dynamics subsequently need to be expressed in terms of peak height changes. A new MPB parameter ϕ is introduced that indicates how peak height values increase or decrease in linear, random, or circular patterns. The initial direction of peak height changes (up or down) is decided randomly for each simulation run. Linear peak changes start with peak height values at

Table 2.2: MPB parameters yielding each environment type defined by Duhain and Engelbrecht [53].

Env	h_s	w_s	s	λ	M_i	M_C	M_ϕ	Env	h_s	w_s	s	λ	M_i	M_C	M_ϕ
A1L	0	1	3	1	100	–	R	C1L	0	1	3	1	20	–	R
A1C	0	1	0	0	100	105	R	C1C	0	1	0	0	20	105	R
A1R	0	1	3	0	100	–	R	C1R	0	1	3	0	20	–	R
A2L	7	1	0	0	100	–	L	C2L	7	1	0	0	20	–	L
A2C	7	1	0	0	100	–	C	C2C	7	1	0	0	20	–	C
A2R	7	1	0	0	100	–	R	C2R	7	1	0	0	20	–	R
A3L	7	1	3	1	100	–	R	C3L	7	1	3	1	20	–	R
A3C	7	1	0	0	100	105	R	C3C	7	1	0	0	20	105	R
A3R	7	1	3	0	100	–	R	C3R	7	1	3	0	20	–	R
P1L	0	0.05	1	1	20	–	R	P2R	1	0.05	0	0	20	–	R
P1C	0	0.05	0	0	20	314	R	P3L	1	0.05	1	1	20	–	R
P1R	0	0.05	1	0	20	–	R	P3C	1	0.05	0	0	20	314	R
P2L	1	0.05	0	0	20	–	L	P3R	1	0.05	1	0	20	–	R
P2C	1	0.05	0	0	20	–	C	–	–	–	–	–	–	–	–

one extreme (top or bottom) and progress values linearly towards the other extreme of the peak height range h_p. The size of height changes is controlled by h_s. Circular peak changes oscillate peak height values across the peak height range h_p in a circular fashion. Peak changes are reflected around the edge of the h_p range in the cases where continued changes in a given direction would break the bounds set by h_p. In both linear and circular cases the peak lengths depend on each particular environment's height severity parameter h_s. Random peak height changes simply assign random peak height values in the peak height range h_p.

The considerations above together with permissible parameter value ranges of the original scenario 2 settings [19][64][127] yield 27 unique environments that are comparable to the majority of the DOP literature.

2.4.2 Criticism of the Moving Peaks Benchmark

Recent findings by Bond *et al.* [18] show that the MPB is a problematic benchmark function generator. Firstly, the MPB landscape is unrepresentative of real-life problems due to the symmetry of optima. Secondly, the MPB shows a bouncing effect near dimensional boundaries which causes peak shift to be less than the actual shift parameter. Lastly, the MPB control parameters lack the capacity to significantly alter landscape characteristics such as *ruggedness*, *dispersion*, *gradient*, and *searchability*.

This thesis uses the MPB as the main benchmark function generator for all investigations for a number of reasons. Firstly, the symmetrical peaks and the bouncing effect of optima are similar across all environment and algorithm combinations, and do not affect the goals of the investigation. Secondly, The absence of statistically significant differences between the ruggedness, dispersion, gradients and searchability of subsequent search landscapes over time ensures that all algorithms operate on search landscapes of the same complexity. Differences in algorithm performance will subsequently not be due to the search landscape's complexity changing over time. Lastly, Duhain and Engelbrecht [53] present MPB parameter guidance to rigorously yield each of the 27 types of DOP. Redefining the classification of Duhain and Engelbrecht for other benchmark functions is out of scope for this thesis.

2.5 Algorithm Components for Dynamic Environment Optimization

Recent surveys provide comprehensive overviews of meta-heuristics from the SI and EC fields that are specialized to solve DOPs [40][93][119][136]. Jin and Branke [93], Cruz *et al.* [40], and Nguyen *et al.* [136] give comprehensive overviews on *evolutionary dynamic optimization* (EDO) and *swarm intelligence dynamic optimization* (SIDO). Mavrovouniotis *et al.* [119] provide in-depth focus on existing and emerging SIDO approaches.

The surveys in [40][93][119][136] discuss key components (or "building blocks") found in the majority of DOP-specific meta-heuristic methods, namely *introducing/maintaining diversity*, *detection of changes*, *reacting to changes*, *use of explicit/implicit memory*, *multiple populations*, *predicting changes*, and using *self-adaptive mechanisms*. Any optimiza-

tion algorithm focused on solving DOPs needs to incorporate these components in some shape or form. The rest of this section discusses the functionality that each component provides.

2.5.1 Maintaining/introducing Diversity

Diversity management is critical for tracking the optimum as well as detecting new optima. If the spatial severity of a change to the search landscape is large (potentially causing new optima to be located outside of the radius of the population), many optimization algorithms will be unable to react to the changed environment. Higher diversity improves the exploration ability of the population of candidate solutions and allows the new optima to be located [60].

Diversity can either be *introduced* after an environment change or be *maintained* throughout the search process [93]. Methods that introduce diversity perform well in environments that have frequent small/medium changes, while methods that maintain diversity are good for environments with multiple Type II/III heterogeneous peaks, but are ineffective when changes are small [136]. Diversity maintaining methods have been shown to perform well in environments with infrequent changes, as well as in DOPs with severe spatial changes [136]. However, these methods tend to be slower because of the extra computational overhead to maintain diversity [136][93].

Introducing diversity requires less computational overhead than maintaining diversity, but requires that correct perturbation sizes be established (which may be problem and domain dependent). In some cases it is possible that too little diversity introduced too late in the process may not be enough to counteract the effects of an algorithm that has converged. Care should be taken not to disrupt the operation of an algorithm through replacing too many candidate solutions that the algorithm relies on to work, or replacing the best individuals in the population (such as the global best position in a PSO).

2.5.2 Detecting Changes in the Environment

Detecting change is important for many methods, such as methods that rely on any form of *memory* (i.e. the personal best position in PSO), or methods that are designed to

converge and then diversify again once environment changes occur. Nguyen *et al.* [136] outline how detection of change can happen in three main ways, namely using sentinels as detectors that are re-evaluated to determine if a change has occurred, detecting change based on algorithm behavior, and that environment changes are simply made known to the algorithm.

Detecting changes through dedicated detectors is the most commonly found strategy [136]. Such detectors can form part of the population, or be maintained separately from the main population. Using detectors can have varying results based on the exact optimization problem at hand. Using too few detectors may result in environment changes never being discovered, especially in cases where only certain sub-regions of the fitness landscape changes. Using too many detectors results in unnecessary function evaluation overhead.

Detecting changes based on algorithm behavior involves looking for certain tell-tale signs of the algorithm encountering a landscape that is unexpectedly different than the landscape of the immediate past. Example mechanisms include monitoring the change in the average of the best solutions over a window of iterations, or monitoring the relationship between the diversity of fitness values and the rate of change detection, among many others [136]. Ultimately, an advantage of most approaches that detect changes by considering algorithm behavioral changes is that no extra function evaluations are used. The down-side, however, is that these approaches tend to have high false positive and high false negative rates causing inaccurate change detection events to be triggered. Many approaches are algorithm specific as well, making them unusable across different approaches.

Change detection is out of scope for the purpose of this thesis, and all environment changes are simply made known to the algorithms. This avoids propagating any environment change detection errors through to the algorithms being investigated, allowing a more fair comparison between the methods.

2.5.3 Reacting to Changes

Reacting to environment changes is a critical function of any algorithm solving a DOP. Once the environment changes, an algorithm might be in a state that makes it hard or

even impossible to track new optima. Examples of such situations are when all particles in a PSO or all individuals in a EA have converged on a single point, or when particle memories still refer to personal and global best positions that do not accurately reflect the true state of the objective function anymore.

Reactions to change can be tightly linked to *detecting change* in which case the algorithm explicitly performs actions to cope with the changes in the fitness landscape. Actions may include using a refreshing strategy to update outdated memory, triggering hyper-mutation of individuals to increase diversity, or any other mechanisms set out by the inventors of the heuristic.

Reactions to change can also simply be pervasive in that no explicit action needs to be taken when changes occur. Algorithms that employ this strategy essentially aim to always be in a position to react to a changing landscape. Examples of such strategies include mechanisms such as random immigrants, Brownian individuals, or charged particles that maintain high diversity throughout the run [60].

2.5.4 Implicit/explicit Memory

Memory-based approaches consist of *implicit* memory approaches, such as keeping state in the solution representation of an EA, or *explicit* memory, such as keeping a list of good solutions and the cyclic conditions that lead to them [136]. Nguyen *et al.* [136] review a number of bespoke EA approaches that exploit diploid and multiploid genomes as implicit memory. Nguyen *et al.* also review explicit memory approaches and how such approaches need to provide strategies to

- decide if direct memory of previous good solutions is useful,
- decide if an associative memory approach is required that tracks environment and algorithm-specific conditions over time (i.e. environment state transition probabilities, population distribution statistics, probabilities of good solutions in different parts of the search landscape, among other approaches),
- stipulate how memory is updated or refreshed,
- state when memory is to be updated, and

- decide how to use any memory as part of the algorithm execution.

Memory-based approaches may be good in situations where cyclical behavior occurs in the landscape change, yet many memory schemes tend to suffer from outdated and obsolete information after environment changes [40][93][119][136].

2.5.5 Multiple Populations

The use of multiple populations is a core component to the many CI methods that focus on DOPs. Nguyen *et al.* [136], Cruz *et al.* [40], and Jin and Branke [93] confirm that the use of a parallel search strategy is one of the most successful approaches to solve DOPs. The general idea is that each sub-population handles a different task. Tasks identified by Nguyen *et al.* [136] that are addressable by multiple sub-populations include handling different parts of the search space, finding and tracking different peaks, managing exploration and exploitation activities separately, and separating the search process from keeping track of feasible regions (when constraints are used). Depending on the exact tasks handled by sub-populations it may be important to ensure that the sub-populations do not overlap. The optimal division of the population into sub-populations may also depend on t as well.

2.5.6 Predicting Changes

Prediction of change, similar to the use of memory, aims to learn and to exploit patterns in the dynamic environment by trying to predict future environment changes. The types of prediction that have been applied by various studies range from predicting *when* changes would occur and predicting *where* the location of optima will be after changes occur [136]. Nguyen *et al.* [136] highlight a number of studies that show that the use of prediction could negatively impact the optimization method. Specifically, if the distribution from which the training data is drawn is not sufficiently representative of the distribution encountered in practice, then the predictive models are likely to fail.

In this thesis, the conditions after environment changes are not predicted in order not to bias algorithm results due to inaccurate predictions. Environment changes are simply made known to the algorithm after which all algorithms have an equal opportunity to adapt to the new environment.

2.5.7 Self-adaptation

Self-adaptive and self-reconfiguring methods are identified by Nguyen *et al.* [136] as an important focus area for current DOP research efforts. Self-adaptation is loosely related to change prediction [136], considering that self-adaptation can be seen as a result of predicting how the algorithm could be improved based on historical data. Examples of current state of the art methods to solve DOPs that use self-adaptive parameters and multiple populations include DynDE [122], jDE [22] and CDE [52].

Self-reconfiguring methods and adaptive meta-heuristics are a core focus of this thesis. Chapter 3 elaborates in detail how hyper-heuristics can be used to increase the effectiveness of meta-heuristics compared to running those meta-heuristics in isolation.

2.6 Population-based Meta-heuristics for Dynamic Optimization

Modern meta-heuristics for DOPs combine the building blocks listed in section 2.5 to balance the *exploration* of an ever-changing search landscape against the *exploitation* of the search landscape at time t. A number of recent state-of-the-art EA and SI algorithm examples are reviewed in Nguyen *et al.* [136], Jordehi [95], Das *et al.* [42], and Mavrovouniotis *et al.* [119]. In the discussion in this thesis, the term "*entity*" represents an individual candidate solution that forms part of the overall population of candidate solutions. Entities also track any associated (algorithm-specific) state[1] about the candidate solution such as, for example, particle velocities or global best positions.

The following sections present a number of well-known examples from the EA and SI fields.

[1]Entities may be implemented in different ways at the discretion of the implementer. Common techniques to represent entities include objects in object oriented programming, functions in functional programming, lookup tables, among other methods.

2.6.1 Population-based Meta-heuristics in Evolutionary Computation

The field of EC is inspired by the process of natural evolution of organisms in an environment, and early work dates back to the 1950's [60]. Over time parents reproduce to create offspring, of which only the fitter offspring reproduce and/or survive to the next generation. This process is repeated over multiple generations of reproduction and survival. By the end of the process the individuals in the population are typically very good solutions to the optimization problem being solved.

Much work has been done to create EC algorithms that solve DOPs. Nguyen *et al.* [136] review the state-of-the-art across the EDO spectrum. The reviews of Das *et al.* [42][43] focus on the advances made in applying a particular EC algorithm called *differential evolution* (DE) by Storn and Price [166] and how DE has been adapted to focus on DOPs.

Classic DE has been shown to perform poorly in DOPs, since once DE converges it becomes impossible for the algorithm to detect new or moved optima [43]. The classic DE algorithm has been extended with diversity management strategies to cope better with changing environments. Well-known examples of such strategies are entity reinitialization where entities are simply recreated randomly in the search space after environment changes, quantum individuals that are randomly reinitialized within a hypersphere centered around the best individual in the population, Brownian individuals that take steps of random size and direction from the position of the best individual, using multiple populations of individuals with exclusion rules that prevent all individuals from converging to the same location, and entropic measures that simply add random noise to individuals [42][43][60].

Many studies investigate DE with various combinations of diversity management strategies. *DynDE* by Mendes and Mohais [122] is a DE variant that combines Brownian individuals with exclusion criteria to prevent populations converging to the same optima. Du Plessis and Engelbrecht [51][52] present *competing differential evolution* (CDE) which extends *DynDE* by adopting a multi-population approach called *competitive population evaluation* to locate optima faster using competition for function evaluations, *reinitialization midpoint check* to ensure populations remain on different peaks throughout the

run, and a third approach that combines both of the previous additions.

DDEBQ by Das *et al.* [41] uses multiple populations and combines Brownian individuals and adaptive quantum individuals in conjunction with DE individuals to maintain diversity. The approach applies a neighborhood-based mutation strategy to avoid premature convergence, an exclusion rule to space out the populations across the search space, and an aging mechanism that stops stagnation at local optima.

Hui and Suganthan [87] propose a self-adaptive niching DE for DOPs with a multi-trajectory local search component that exploits different niches. Mukherjee *et al.* [129] present a multi-population based clustering DE for DOPs that uses crowding in dynamic niching to maintain overall diversity, and a crowding based archive. *Population-based incremental learning* (PBIL) [120] by Mavrovouniotis and Yang uses a combination of population-based evolutionary algorithms and incremental learning mechanisms together with a random immigrants approach to learn and exploit a probability vector of promising solutions.

Many self-adaptive DE variants have also been developed. While these studies do not directly target DOPs, the findings are pivotal for any strategy that incorporates DE as part of the overall solution. *Self-adaptive differential evolution* (SaDE) by Qin and Suganthan [150] automatically adapts its learning strategy and parameters during the search. *jDE* [22] by Brest *et al.* encodes the DE control parameters into the individual to allow self-adaption. *Adaptive multi-population framework* (AMP) [104] adaptively adjusts the number of populations based on feedback about the number of peaks in the search landscape. Segura *et al.* [158] take a controversial approach to DE control parameter adaptation schemes that is based on feedback from the environment. The study shows how state-of-the-art trial vector generation methods are ineffective in situations where high diversity is needed to maintain good exploration. Drawing the value for the DE scaling factor from a Cauchy or Gaussian distribution yields superior results.

The *random immigrants genetic algorithm* (RIGA) by Grefenstette [72] is a modification to the standard GA to allow the algorithm to cope better in DOPs. Every generation a subset of the population is replaced by randomly generated individuals according to a *replacement rate*. To allow RIGA to be applied to real-valued problems, *arithmetic crossover* (as described by Michalewicz [124]) can be used instead of uniform

crossover. Arithmetic crossover requires that two parent entities $\mathbf{x}_1$ and $\mathbf{x}_2$ create an offspring entity $\mathbf{o}$ as follows: $\mathbf{o} = (1 - \lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2$, where $\lambda \in [0, 1]$. Grefenstette [72] proposes a replacement rate of 0.1 and a very low mutation rate of 0.001 to prevent too much perturbation of solutions. Both Cruz *et al.* [40] and Nguyen *et al.* [136] list RIGA as a good example of a DOP-specific method that maintains diversity throughout the optimization run.

Multi-phase multi-individual extremal optimization (MMEO) [128] by Moser and Hendtlass is another EC approach that is aimed at solving the moving peak benchmark function as well as possible. In MMEO multiple solutions are repeatedly mutated using a power-law distribution that attempts to exclude bad solutions rather than to find good solutions.

2.6.2 Population-based Meta-heuristics in Swarm Intelligence

The field of SI mimics and exploits the collective behavior found in groups of animals such as schools of fish, flocks of birds, colonies of ants, swarms of bees, and many other metaphors from the animal kingdom [59]. Arguably the most well-known and widely used SI algorithm (for continuous optimization) is the family of PSO algorithms. A common theme across SI approaches is the idea of a swarm of simpler elements which are guided by a combination of their own experience (a cognitive component) and the experience of their neighbors (a social component). An example of such simpler elements are *particles* in a PSO.

Similar to EC approaches, SI algorithms developed for static environments may find it difficult to adapt to ever-changing search landscapes in DOPs. Once the swarm converges to a point in the search space at time t the algorithm may no longer be able to locate and exploit any new optima after environment changes. Many diversity enhancement techniques have been introduced to PSO to address this weakness. One of the earliest techniques by Hu and Eberhart [85] proposes reinitializing part or all of the swarm when changes occur. Other examples include *quantum particles* that have a different position update equation from other particles in the swarm. The positions of quantum particles are randomly reinitialized within a hypersphere centered around the swarm's global best particle. Other examples include *charged particles* that repel each other based on the

distance between them, *random inertia weights* for particles, and simply reinitializing part or all of the swarm after the environment changes, among many other approaches [59].

Jordehi [95] and Mavrovouniotis *et al.* [119] review a number of PSO variants that are well-suited to solve DOPs. Some well-known state-of-the-art approaches adopt one or more diversity enhancement techniques and are discussed in the paragraphs below.

Adapting the neighborhood topology of a PSO is a common mechanism to enable PSO to cope with dynamic environments. Janson and Middendorf [91][92] propose partitioned hierarchical PSO (PH-PSO) which uses a dynamic neighborhood strategy to cope with DOPs. PH-PSO splits the swarm into multiple sub-swarms after an environment change, and each sub-swarm independently searches for the optimum. Sub-swarms are reunited with the overall neighborhood hierarchy over time. An adaptive version of PH-PSO adapts the number of iterations before sub-swarms are reunited with the hierarchy.

Using a specialized static topology can also help to preserve diversity in the swarm. Fine-grained PSO [112] by Li and Dam uses a Von Neumann topology where particles are connected to only four neighbors, thus resulting in slower information propagation across the population. Zheng and Liu [189] use two sub-swarms where one swarm uses a global neighborhood topology to focus on finding the optima and the second swarm uses a locally connected topology that preserves diversity. Over time each swarm exchanges information about the search space.

A well-known PSO approach to cope with changing environments is to rely on repulsion or reinitialization of particles. *Atomic particle swarm optimization* (APSO) and *charged particle swarm optimization* (CPSO) by Blackwell and Bentley [13][14] rely on *charged* particles that repel each other in a Coulomb force-like manner based on the magnitude of the distance between them. The resulting 'suspension of charged particles' ensures high diversity throughout the search process.

Coulomb-like repulsion occurs between pairs of charged particles in the swarm when the Euclidean distance between the particles is less than the *perception limit*. No acceleration occurs if the distance between particles is greater than the perception limit. Particles that are closer to each other than the *core limit* have their acceleration capped to prevent extremely large acceleration terms. An *acceleration* term is computed for

each of the n_s particles as follows

$$\mathbf{a}_i(t) = \sum_{l=1, l\neq i}^{n_s} \mathbf{a}_{il}(t) \tag{2.5}$$

where the repulsion between particles i and j is defined as

$$\mathbf{a}_{il} = \begin{cases} \frac{Q_i Q_l (\mathbf{x}_i(t) - \mathbf{x}_l(t))}{\|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|^3} & \text{if } R_c \leq \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\| \leq R_p \\ \frac{Q_i Q_l (\mathbf{x}_i(t) - \mathbf{x}_l(t))}{R_c^2 \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|} & \text{if } \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\| < R_c \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where Q_i and Q_l are the charges of particles i and l respectively, $\|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|$ is the Euclidean distance between particles, and R_c and R_p are respectively the core limit and perception limit of the particle. The classic PSO velocity equation is modified to include the acceleration term as follows

$$v_{ij}(t+1) = w v_{ij}(t) + c_1 r_{1j}(t)[v_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] + a_{ij}(t) \tag{2.7}$$

where $v_{ij}(t)$ and $x_{ij}(t)$ are, respectively, the velocity and position of particle i in dimension j at time t, c_1 and c_2 are constants used to scale the influence of the cognitive and social acceleration components of the particle, and $r_{1j}(t) \sim U(0,1)$ and $r_{2j}(t) \sim U(0,1)$ are random variables drawn from a uniform random distribution.

The difference between APSO and CPSO is that all particles in CPSO are charged, whereas only 50% of particles are charged in APSO. *Charged* particles have a charge $Q > 0$ while *neutral* particles have $Q = 0$. CPSO has a stronger focus on exploration and maintains a higher solution diversity than APSO, resulting in CPSO working well in environments with severe spatial changes [14]. APSO balances exploration and exploitation better than CPSO, and works well in environments with high temporal change severity [12]. The parameters R_c, R_p, and Q are domain dependent. Blackwell and Bentley [14] use a swarm of 20 particles and the values $Q = 16$, $R_c = 1$, and $R_p = \sqrt{3}x_{\max}$, where $x_{\max}$ is the maximum extent of the domain. Blackwell and Bentley [14] clamp particle velocities to the range $[-v_{\max}, v_{\max}]$ to avoid sudden massive acceleration coefficients from derailing the search.

Quantum particle swarm optimization (QPSO) by Blackwell and Branke [15][16] is a computationally simplified model inspired by APSO and CPSO. The swarm consists of

a mix of charged and neutral particles. Instead of using Coulomb-like repulsion between charged particles, the charged particles are simply re-initialized randomly inside an n_x-dimensional sphere B_n of radius r_{cloud} centered around the *gbest* particle at time t, namely $\hat{\mathbf{y}}(t)$. The PSO position update becomes

$$\mathbf{x}(t+1) = \begin{cases} \mathbf{x}(t) + \mathbf{v}(t) & \text{if } Q_i = 0 \\ B_n(r_{cloud},\ \hat{\mathbf{y}}(t)) & \text{if } Q_i \neq 0 \end{cases} \tag{2.8}$$

where $\mathbf{v}(t)$ is the classic PSO velocity update equation [60]. Blackwell *et al.* [15] comment that CPSO and APSO can be difficult to control, since the spatial extent of the charged swarm depends on the Euclidean distance between charged particles. The $B_n(r_{cloud})$ operator of QPSO, in contrast to APSO and CPSO, yields sustained and controlled diversification behavior. In addition to depending on the magnitude of r_{cloud}, the quantum position update function B_n can also utilize different types of probability distributions that govern where a quantum particle will be reinitialized. Harrison *et al.* [79] investigate ten different probability distributions ranging from uniform, Gaussian, Cauchy, beta, Exponential, Weibull, and various triangular distributions. Their investigation concludes that different types of environment dynamics results in different distributions being the better choice. The study also found that smaller r_{cloud} values tend to perform better, which opens the possibility that the role of quantum particles may not be purely exploratory but that they might actually help in exploitation as well.

Many multi-population PSO variants rely on splitting the swarm into sub-swarms that independently explore the search space. Many approaches opt for each sub-swarm to employ different search logic. Examples include multi-swarm quantum/charged PSO [16][17], clustering PSO [106][188], child and parent PSO [96], fast multi-swarm PSO [105], forking PSO [178], triggered memory-based PSO [111], collaborative PSO [115], dynamic niching PSO [138], and unified PSO [145] among many others.

Speciation PSO (SPSO) [110][111][144] dynamically splits the swarm into multiple *species* that operate independently. Each species is defined by a species seed that is the best-fit particle in the species, and a species radius, r_s. All particles that fall within the radius r_s from the species seed is considered a part of that species. Species are subtly different than sub-swarms since, firstly, particle membership in a species is defined by the radius r_s and are adjusted every iteration while sub-swarm particle allocations are

fixed, and, secondly, the number of species is adjusted dynamically while the number of sub-swarms is fixed. A recent variant by Luo *et al.* [116] combines a speciation approach with memory to cope better in dynamic environments.

2.6.3 Discussion of EC and SI Population-based Approaches

The EA and SI meta-heuristics discussed above all employ one or more of the building blocks discussed in section 2.5. As discussed in the thesis scope section in chapter 1, the aim of this study is to explore how well selection hyper-heuristics can continuously select the most appropriate heuristic to employ at time t. Specifically, the building blocks discussed in section 2.5 are incorporated as follows:

1. The *heuristic layer* is responsible for *maintaining/introducing diversity* in the problem space, *reacting to change* of the environment, and *managing memory/state* in the problem space. Each heuristic is encapsulated as a distinct collection of low-level operators, design decisions, and parameter values. The result is a mix of highly varied modes of operation. For example, an SI approach comprising of specific *neighborhood structures*, *electrostatic charges*, *position update* operators, and *velocity update* operators will exhibit completely different behavior than an EC approach with *crossover*, *mutation*, *replacement*, and *elitism* operators. Deliberate parameter values generally produce very specific *exploration* and *exploitation* behavior in each meta-heuristic [57][58].

2. The *hyper-heuristic* layer is responsible for managing *multiple populations* of candidate solutions that focus on different aspects of the search, *maintaining a memory* of correlations between heuristic allocation, feedback values, and resulting performance, and *self-adaptation* of search behavior by learning how to best allocate computational resources to the most promising heuristics at time t. Different selection hyper-heuristics implement these building blocks in different ways.

Detection of change is out of scope in this study to avoid bias caused by any variance in change detection strategies. Generally, the pool of heuristics would manage change detection in practical applications.

To avoid any bias the pool of heuristics in this study does not contain any *self-adaptive* heuristics, heuristics that employ multiple *sub-populations*, or heuristics that utilize *explicit memory* of predictable environment behavior. Inclusion of these types of heuristics could result in the heuristics "counteracting" the hyper-heuristic in certain unforeseen circumstances. Specifically, a self-adaptive heuristic could (loosely speaking) try to "undo" the adaptive actions imposed on by the hyper-heuristic. Such interactions between the heuristic and hyper-heuristic layers are of great importance and provide an important avenue of future research. Chapter 3 elaborates more on this dynamic. The scope of this thesis does not include management of this type of complexity.

These reasons exclude many state-of-the-art self-adaptive and/or multi-population methods discussed above from the study. However, the overall aim of studying the performance and behavior of different selection hyper-heuristic mechanisms is not compromised. Generally, however, self-adaptive meta-heuristics can and probably should be utilized as part of any practical selection hyper-heuristic approach.

2.7 Comparing Algorithm Performance in Dynamic Environments

Comparing the performance and behavior of multiple algorithms solving a DOP is a daunting task. Dynamic environments continually change over time resulting in ever-changing optimum values and/or optima locations, varying ranges of possible fitness values, different numbers of optima appearing and disappearing across environment change periods, and many other changing facets. As a result, performance measures aimed at static environments typically fail to correctly quantify the effectiveness of algorithms in DOPs [40][60][136]. The dynamic properties of DOPs necessitate dedicated methods to gauge the effectiveness of algorithms that aim to solve a DOP.

Moreover, section 2.5 highlighted a number of "building blocks" and strategies that specialized algorithms may employ to cope with the constant changes in the problem. In addition to measuring the performance of an algorithm, researchers ideally want to be able to directly measure the behavior of various algorithms under different conditions present in a DOP over time. Beyond measures of algorithm performance and behavior

lies the challenge of correctly comparing and correlating all measured results to determine which algorithms (if any) are superior. This is no easy task since a fair apples-to-apples comparison in DOPs is difficult to conduct correctly for the reasons outlined above.

This section discusses the facets that constitute good performance measures for DOPs in sub-section 2.7.1. Sub-sections 2.7.2 and 2.7.3, respectively, review various well-known optimality and behavior measures for DOPs from literature. Lastly, an overview of the statistical challenges and viable methods for comparing algorithms performance on DOPs is provided in 2.7.4.

2.7.1 Facets of Performance Measures for Dynamic Environments

Performance measures, firstly, quantify how effective an optimization algorithm is at solving a DOP but, secondly, also allows for the comparison of the performance of various algorithms against each other. Morrison [125] stipulates that any good performance measure for dynamic environments should

- have intuitive meaning,
- allow for statistical significance testing, and
- have sufficient exposure to environment change dynamics.

Most performance measures for static environments tend to focus on the error near the end of the optimization search process [60]. The criteria of Morrison imply that performance measures developed for DOPs should be able to measure the effectiveness of the algorithm across the entire search process. Specifically, a measure should not only express how good the algorithm is at *finding* the optimum solution, but also measure how well the algorithm can *track* optima as the search landscape changes.

Nguyen *et al.* [136] differentiate between *optimality*-based and *behavior*-based performance measures. *Optimality*-based measures focus on evaluating the quality of the solutions found by the optimization algorithm, while *behavior*-based measures determine if the optimization algorithm shows any useful behaviors. Both types of measures together give a more complete picture of the solution quality of a given optimization algorithm, as well as *why* and *how* the algorithm behaves as it does.

Performance measures for DOPs should exhibit positive qualities that are deemed desirable by researchers. Similarly, performance measures for DOPs should not suffer from any undesirable effects or qualities that could lead to misleading interpretations. In many cases there is a trade-off between desirable and undesirable qualities for DOP-based performance measures. Examples include:

- **Protection against ill-defined values**: Measures with unprotected edge cases are generally hard to work with. Examples include measures with equations that contain fractions which could yield zero-valued denominators, measures that rely on search landscape gradients which are subsequently applied to discontinuous problems, measures that use the square-root function that may operate on negative values, or measures with multiplicative terms that could take on zero values (where such zero values would yield a nonsensical or ill-defined measurement value). Care should be taken to ensure that such measures are never exposed to any conditions that could yield undefined results. In many problems it may be hard or even impossible to foresee if such non-permissible situations could occur. er

- **Unified treatment of maximization and minimization problems**: Measures that require no modification to treat function maximization and minimization problems uniformly are more desirable than measures that require alteration or redefinition to support either modality. Measures that rely on *error* values tend to support both optimization goals while measures that rely on *fitness* values tend to require equation modifications that use the reciprocals of fractals and/or the negation of terms and inequality relationships. Such modifications could inadvertently result in the measure being implemented or interpreted incorrectly.

  An example of this situation is the *sampled relative error* of Li *et al.* [108] that relies on a relative error $r(t) = f(\mathbf{x}^*(t))/f(\mathbf{x}_{best}(t))$ for minimization problems and $r(t) = f(\mathbf{x}_{best}(t))/f(\mathbf{x}^*(t))$ for maximization problems where $f(\mathbf{x}_{best}(t))$ is the fitness value of the best found solution and $f(\mathbf{x}^*(t))$ is the value of the optimum at time t. Note how $r(t)$ increases hyperbolically for minimization problems and linearly for maximization problems as $f(\mathbf{x}_{best}(t))$ tends to $f(\mathbf{x}^*(t))$ [52], which makes interpretation of the measure different for maximization versus minimization

problems. The measure is also an example of a method that produces ill-defined results if either $f(\mathbf{x}^*(t))$ or $f(\mathbf{x}_{best}(t))$ are zero.

Measures that report aggregations of *error* values tend to be more intuitive to understand since any observer inherently knows the best possible performance is an error of zero. Measures that rely on aggregations of *fitness* values are less intuitive, since the observer has no clear reference of what the optimal fitness value should be, especially since the optimum fitness value tends to change with search landscape changes. In both cases, however, the optima must be known.

- **Inherent knowledge about the problem**: Generally, any measure for DOPs has to consider a trade-off between *knowledge* about the search landscape and *generalization* to previously unseen types of DOPs. Knowledge about a problem includes, for example, awareness of when landscape changes occur, the number and locations of optima at every point in time, the range of minimum and maximum possible fitness values, and domain boundaries.

  Access to specific knowledge such as the location of the optima directly determines whether or not certain measures can be used at all, i.e. measures that rely on error values cannot be used without knowledge of the location of the optimum. Any measure that exploits domain knowledge is able to capture the performance of an algorithm more succinctly and accurately at the expense of the fact that the measure is only applicable in DOPs where such knowledge is available. This may make the measure unusable in many real-world problems.

- **Reliance on algorithm details**: Measures that are dependent on discrete algorithm iterations tend to make it difficult to compare SI or EA algorithms that use different population sizes. An algorithm with a small population typically has fewer function evaluations per iteration than an algorithm with a larger population. For algorithms with very large populations it may happen that environment changes occur *during* an iteration. This, in turn, may cause false measurements to occur, or make commensurate comparisons across algorithms with different population sizes impossible. Many modern algorithms also have varying population sizes across different iterations. In situations where the number of function evaluations matter

(i.e. where there may perhaps be a cost associated with each function call) it may be preferable to rely on measures that are not influenced by algorithm population sizes.

- **Subjective parameters**: Measures that rely on arbitrary parameter values such as, for example, iteration window sizes, threshold values, and discrete categorization strategies tend to be subjective. The choice of parameter values greatly influences the outcome of the measure. Each parameter in effect becomes a degree of freedom that requires further analysis, which creates unnecessary complexity.

- **Frame of reference**: Measures can be either *absolute* measures or *relative* measures. Absolute measures are grounded against a fixed point of reference. For example, observers know that zero is the best possible value when considering error values, and that higher error values indicate worse performance. Relative measures on the other hand express performance as measured against a subjective point of reference. For example, expressing the performance of an algorithm as a percentage of the improvement in fitness over time is a relative measure. With a relative frame of reference, algorithms with sub-optimal solutions that show relatively large gains in fitness may unfairly be regarded as better than algorithms with superb solutions that show little improvement over the same time period.

  Absolute measures tend to be more intuitive to understand than relative measures, however, the trade-off is that absolute measures require a deeper understanding of the details of a problem. Absolute measures can exploit domain knowledge about a problem (such as the value and location of the optimum at time t), and reveal objective truths about the search process that relative measures simply cannot capture. Consider, for example, a relative measure that tracks the ratio of the best and worst fitness in the population of candidate solutions at time t (which is completely subjective and dependent on the state of the population, not the problem state). An absolute measure would immediately address the inadequacy of the relative measure. On the other hand, relative measures can be deployed in a wider range of situations where domain knowledge may be unavailable (and where it is impossible to use many absolute measures).

- **Measurement focus level**: A good measure for DOPs should refrain from penalizing an algorithm for behavior that is deemed desirable. Consider, for example, a population-based algorithm that uses its population of candidate solutions to, respectively, explore and exploit the search landscape. A measure that focuses on the performance of *every* candidate solution of the population at *every* iteration during the search process will unfairly penalize the algorithm for exploring the search landscape with a subset of the population. On the other hand, a measure that only reports the best solution found by the algorithm at specific time periods of importance (i.e. immediately before or after a landscape change) presents a more focused and fair view of performance.

- **Outliers**: Measures that are strongly influenced by outlier values are sensitive to creating false perceptions of performance. Examples include reporting only the highest/lowest error or fitness values, reporting the greatest/smallest extents of distance in the search landscape, or reporting the average value of a measure in situations where many values are clearly considered outliers versus the majority of the data.

  Measures that are susceptible to outliers tend to skew the perceptions of observers away from the true state of algorithm performance. For example, a diversity measure that relies on the greatest physical extent between any two candidate solutions in the population is highly susceptible to falsely reporting a greater diversity than what is actually present. Algorithms with outliers are given too much sway at the expense of algorithms that truly have better overall diversity, but with smaller maximum extents.

A good measure of optimality balances the facets discussed above to yield a holistic view of algorithm performance.

2.7.2 Optimality-based Measures

Optimality-based measures capture how well an optimization algorithm solves a given optimization problem. Optimality measures can broadly be split into measures that report the difference between the best found solution and the global optimum value,

measures that report only the best found solution, and measures that rely on the distance between the optimum location and the population of candidate solutions.

Measures in the DOP literature are often expressed in language from fields such as SI, EA, or others. In this thesis the terms "population" refers to the collection of candidate solutions that are managed by the optimization algorithm (i.e. *swarm*), "candidate solution" refers to a member in the algorithm's population (i.e. *individual* or *particle*), "iteration" refers to a single application of the algorithm's logic (i.e. *generation*), and "fitness" refers to the value returned by the function to be optimized (i.e. *function evaluation*).

A review of the literature found the following optimality measures below to be key examples for use in DOPs:

1. *Best-of-generation fitness* [6], $P_{\text{BOG}}(t)$, is a series of fitness values over time of the best candidate solution in each iteration $t \in \{1, \ldots, T\}$, defined as

$$P_{\text{BOG}}(t) = F(\mathbf{x}^t_{\text{best}}, t) \tag{2.9}$$

where F is the objective function and $\mathbf{x}^t_{\text{best}}$ is the best candidate solution vector in the population at iteration t. The $P_{\text{BOG}}(t)$ measure is simple to visualize as a graph of fitness values over time and is intuitive to understand. $P_{\text{BOG}}(t)$ does not require knowledge about environment changes nor is knowledge about the optimum needed. Outliers are (relatively) easy to spot visually and the presence of an outlier does not affect further values of $P_{\text{BOG}}(t)$.

However, since $P_{\text{BOG}}(t)$ is unaware of landscape changes, the measure does not provide a way to compare algorithm performance across a full range of DOP dynamics [125]. $P_{\text{BOG}}(t)$ is not normalized, making it hard to compare performance values across DOP landscape changes. Care must be taken when comparing algorithms of different population sizes: since $P_{\text{BOG}}(t)$ is sampled every iteration, algorithms with larger populations get more fitness evaluations and have a natural advantage that $P_{\text{BOG}}(t)$ is unable to express [52]. $P_{\text{BOG}}(t)$ only measures the best candidate solution per iteration, and does not yield any insight as to how well the rest of the population is performing. Lastly, the measure makes it hard to compare the performance of multiple algorithms (visually or statistically) because performance results are not scalar values [125].

2. *Collective mean fitness* [125], P_{CMF}, records the mean fitness of the best solution over the entire experiment, defined as

$$P_{CMF} = \frac{1}{T}\sum_{t=1}^{T} F(\mathbf{x}_{\text{best}}^{t}, t) \tag{2.10}$$

where T is the number of iterations. The measure is also known as the *mean best-of-generation fitness*. P_{CMF} improves upon P_{BOG} since the output of P_{CMF} is a single numerical value which makes it easier to statistically compare the performance of algorithms. The measure does not require direct knowledge of the optimum nor does it require knowledge about environment change boundaries. Generally, P_{CMF} is intuitive to understand and the measure cannot produce ill-defined values. Performance is considered across the entire time horizon of t and not only at certain periods.

However, P_{CMF} is not normalized across different landscape changes (which might each have different fitness value ranges) and can cause misleading results. This makes it hard for observers to judge performance of an algorithm since they have no objective basis to compare the value of P_{CMF} against. To make matters worse, P_{CMF} does not treat maximization and minimization problems in the same way, making interpretation even harder. Similarly, when averaging P_{CMF} across multiple samples, the measure will be affected by the fact that different problem instances have different fitness scale ranges in each change period, causing misleading results. Different problem types can also not be compared using this measure. Like P_{BOG}, P_{CMF} relies on the algorithm's definition of an *iteration*, and algorithms with larger populations have a natural advantage that P_{CMF} will not highlight [52]. Outliers will affect P_{CMF} if they occur often enough and at large enough magnitudes, but generally P_{CMF} is relatively protected from outliers since the average is taken over T iteration (where T is usually a large value such as 1000).

3. *Collective mean error* [125], P_{CME}, is related to P_{CMF} but uses the objective function error value E instead of the fitness value F, and is defined as

$$P_{CME} = \frac{1}{T}\sum_{t=1}^{T} E(\mathbf{x}_{\text{best}}^{t}, t) \tag{2.11}$$

P_{CME} improves on P_{CMF} by providing zero as an objective point of reference. The optimum algorithm will always have a value of $P_{CME} = 0$, which is more intuitive for observers to understand. C_{CME} generally does not produce ill-defined values. Unlike P_{CMF}, the P_{CME} measure allows uniform treatment of maximization and minimization problems. P_{CME} produces a single numerical value that is easier to compare than P_{BOG}. Like P_{CMF}, performance is considered across the entire time horizon of t and not only at certain periods.

However, P_{CME} does require knowledge about the optimum to compute the error. Similar to P_{CMF}, the P_{CME} measure does not allow reliable comparison of performance across environment change boundaries, across samples, or across different problem types since the error values are not normalized. Like P_{CMF}, the P_{CME} measure takes an implicit salient parameter, namely that the measure relies on the algorithm's definition of an *iteration*. This results in algorithms with larger populations having a natural advantage that P_{CME} will not expose. Outliers will have an effect on P_{CME} if they occur often enough and at large enough magnitudes but, like P_{CMF}, P_{CME} is generally relatively protected from outliers when the number of iterations T is a sufficiently large value.

4. *On-line performance* [20][44], P_{OF}, is the average of all fitness function evaluation values, defined as

$$P_{OF} = \frac{1}{n_{fe}} \sum_{e=1}^{n_{fe}} F(\mathbf{x}^e, e) \tag{2.12}$$

where n_{fe} is the total number of fitness function evaluations and $\mathbf{x}^e$ is the candidate solution being evaluated at function evaluation e. P_{OF} does not require knowledge of environment changes nor knowledge of the location of the optimum. However, P_{OF} gives little insight into the best performance of an algorithm, since the fitness of all other candidate solutions in the population are also included in the measure. This penalizes algorithms that explore sub-optimal areas of the search space, i.e. algorithms that explore heavily just after landscape changes (that finally yield improved final solutions before the next landscape change) will, incorrectly, appear to have a worse P_{OF} score than more mediocre algorithms that do not explore as widely.

P_{OF} does not treat maximization and minimization problems in the same way, making measured values hard to interpret and compare across problems. P_{OF} is sensitive to outliers, since any significantly deviating values that occur often enough will bias the result. This is especially true for algorithms that heavily explore inferior regions of the search space. Similar to P_{CMF} and P_{CME}, P_{OF} does not allow reliable comparison of performance across environment change boundaries, across samples, or across different problem types since the fitness values are not normalized. Unlike P_{CMF} and P_{CME}, P_{OF} does not take any implicit parameters such as the algorithm's notion of *iterations*. Instead, values are taken after every fitness evaluation. P_{OF} can not produce ill-defined values.

5. *Modified off-line error* [20][21], P_{MOE}, is the running average of the best error over the iterations evaluated since the previous environment change occurred, defined as

$$P_{MOE} = \frac{1}{n_{fe}} \sum_{e=1}^{n_{fe}} E(\mathbf{x}^e_{\text{best}}, e) \tag{2.13}$$

where $\mathbf{x}^e_{\text{best}}$ is the best candidate solution found since the last environment change. P_{MOE} is always greater or equal to zero, with $P_{MOE} = 0$ indicating perfect performance. The measurement level is at every function evaluation which ensures that algorithms that consistently show good performance are highlighted (in contrast to measures that take readings at the end of a environment change periods). The faster an algorithm finds a good solution after an environment change, the lower the value of the measure. As such P_{MOE} is intuitive to understand and reason about.

P_{MOE} is resilient to outliers, since only the error of the best candidate solution is considered. The focus on only the best candidate solution means that algorithms that explore sub-optimal regions of the search landscape, with a subset of the population, are not punished unduly. P_{MOE} provides unified treatment of maximization and minimization problems. Lastly, P_{MOE} is highly resilient to ill-defined values since the measure does not rely on any fractions or multiplicative terms, nor any salient algorithm parameters (such as the notion of iterations or generations).

However, P_{MOE} requires that the time of environment changes be known as well as knowledge of the optimum location. A drawback of P_{MOE} measuring performance at every function evaluation is that algorithms that rely on synchronous updates[2] will be unable to update $\mathbf{x}^e_{\text{best}}$ until the algorithm iteration is complete. Care should be taken when comparing synchronous and asynchronous algorithms using P_{MOE}. P_{MOE} does not normalize error values which means that results cannot be compared across different change periods in the same problem instance, across multiple problem instances, or across problem types since the magnitude of the error depends on the magnitude of the fitness in each specific change period of each specific problem instance. Direct comparisons or aggregations of P_{MOE} (such as deviation, mean, minimum, maximum, or median values over change periods, samples, or problem types) will subsequently create false impressions of performance. Nevertheless, P_{MOE} is one of the most commonly reported performance metrics in DOP literature [136].

6. *Modified off-line performance* [20][21], P_{MOF}, is similar to P_{MOE} but uses the optimization function value F instead of the error E, and is defined as

$$P_{MOF}(t) = \frac{1}{n_{fe}} \sum_{e=1}^{n_{fe}} F(\mathbf{x}^e_{\text{best}}, e) \tag{2.14}$$

P_{MOF} retains mostly all the characteristics, advantages and disadvantages of P_{MOE} except that P_{MOF} does not require knowledge of the optimum. This hampers the intuitive understanding of P_{MOF} since the observer has no clear view of what good or bad performance values are. P_{MOF} values may also be negative. P_{MOF} thus loses the ability to treat maximization and minimization problems in a uniform manner. However, P_{MOF} has the ability to be used in problems where it is impossible to know what the optimum is.

7. *Average best error before change* [172], P_{ABEBC}, is the average of the error of the best solution across all time instants immediately preceding an environment

[2] Algorithms that perform synchronous updates evaluate and update their entire population at once, as opposed to asynchronous updates where algorithms evaluate and update each candidate solution one at a time, after which any fitness changes are immediately visible.

change, defined as

$$P_{ABEBC} = \frac{1}{n_c} \sum_{c=1}^{n_c} E(\mathbf{x}_{\text{best}}^{t_c}, t_c) \tag{2.15}$$

where n_c is the number of environment changes, t_c is the time step just before the c-th environment change occurs, and $\mathbf{x}_{\text{best}}^{t_c}$ is the best candidate solution at time t_c. P_{ABEBC} is useful in situations where the final algorithm output of each change period is important, irrespective of how well the algorithm performs during the rest of the change period. Since the focus is on time step t_c and only on the best solution found in the entire population, the measure is relatively resilient to outliers caused by algorithm fluctuations (such as a flurry of diversity increasing behavior immediately after environment changes). P_{ABEBC} allows the performance of different algorithms to be compared more easily since the use of error values allows maximization and minimization problems to be handled the same way.

However, care should be taken when interpreting results. P_{ABEBC} gives no indication about how well any algorithm performed during the entire change period, since point-in-time measurements of algorithm performance is taken at time $t = t_c, \forall c = 1, \ldots, n_c$ instead of the entire change period. P_{ABEBC} is not normalized, biasing the measure towards change periods where the magnitude of the fitness landscape is large. The measure also requires knowledge of the optimum. Lastly, P_{ABEBC} gives no indication as to the volatility of the algorithm being measured, since the average over n_c change periods gives no indication as to the *deviation* in the best error values immediately before changes.

8. *Lowest/highest best error before change* [108], P_{LBEBC} and P_{HBEBC}, are respectively the average lowest and highest error of the best candidate solution just before an environment change, measured across all environment change periods, defined as

$$P_{LBEBC} = \min_{c=1,\ldots,n_c} E(\mathbf{x}_{\text{best}}^{t_c}, t_c) \tag{2.16}$$

and

$$P_{HBEBC} = \max_{c=1,\ldots,n_c} E(\mathbf{x}_{\text{best}}^{t_c}, t_c) \tag{2.17}$$

P_{LBEBC} and P_{HBEBC} are counterparts to P_{ABEBC}. Very little can be learned from P_{LBEBC} or P_{HBEBC} due to the sensitivity of these measure to outlier values.

Any algorithm that achieves a very low or high error at some point during the optimization process will seem more effective or ineffective than another algorithm that consistently achieves low or high errors respectively.

9. *Optimization accuracy* [61][181], $P_{RE}(t)$, is also called the *relative error* and returns a score in the range $[0, 1]$, defined as

$$P_{RE}(t) = \frac{F(\mathbf{x}^t_{\text{best}}, t) - \text{Min}_F(t)}{\text{Max}_F(t) - \text{Min}_F(t)} \tag{2.18}$$

where $\mathbf{x}^t_{\text{best}}$ is the best entity in the population at iteration t, and $\text{Max}_F(t)$ and $\text{Min}_F(t)$ respectively are the best and worst possible fitness values in the problem space at time t. P_{RE} values close to 1.0 indicate good performance while values near 0 indicate poor performance. $P_{RE}(t)$ is less biased towards fitness magnitude changes between different change periods of different problem instances. P_{RE} values can more readily be compared between maximization and minimization problems since it is easy to ensure that $P_{RE} = 1$ always indicates the best possible performance. Since P_{RE} focuses on $\mathbf{x}^t_{\text{best}}$, the measure is relatively shielded against outlier values, and does not penalize any algorithm for exploratory behavior.

However, $P_{RE}(t)$ requires knowledge of both the *best* and *worst* possible fitness values in the search landscape for each instant of time t. $P_{RE}(t)$ is undefined in situations where the search space is a plateau at any time t, since the denominator would become zero. Care should be taken to guard against ill-defined results.

10. *Window accuracy* [181], $P_{WA}(t)$, is the ratio of the difference between the best and worst fitness of an iteration compared to the difference between the best and worst fitness found by the algorithm within a window of iterations, defined as

$$P_{WA}(t) = \frac{F(\mathbf{x}^t_{\text{best}}, t) - W_{\text{worst}}(t)}{W_{\text{best}}(t) - W_{\text{worst}}(t)} \tag{2.19}$$

with

$$W_{\text{best}}(t) = \max_{t'=t-\omega+1,\ldots,t} \left\{ \max_{i=1,\ldots,P} \{F(\mathbf{x}^{t'}_i, t')\} \right\}$$

$$W_{\text{worst}}(t) = \min_{t'=t-\omega+1,\ldots,t} \left\{ \min_{i=1,\ldots,P} \{F(\mathbf{x}^{t'}_i, t')\} \right\}$$

where ω is the number of iterations in the window, and $w_{\text{best}}(t)$ and $W_{\text{worst}}(t)$ respectively are the best and worst fitness values found by the algorithm in a running window of ω iterations (assuming a maximization problem). $P_{WA}(t)$ does not require knowledge of the optimum nor knowledge of when environment changes occur.

However, since $P_{WA}(t)$ is a relative measure based on the best and worst fitness values *found to date* during the search process, the measure can give misleading results if the difference between the best and worst entities in the population is low. The measure is also undefined if the population has converged to the same fitness value during the whole window since the denominator would become zero. $P_{WA}(t)$ takes a subjective parameter value which greatly affects the values of the measure, namely the size of the window, ω.

$P_{WA}(t)$ also takes an implicit parameter, namely knowledge about the iteration length of an algorithm. Algorithms that have different iteration sizes will in effect have more function evaluations in (seemingly) the same window length ω. Lastly, it is not immediately intuitive to convert the equations that constitute $P_{WA}(t)$ from maximization to minimization, which makes the measure harder to use than other measures that treat both types of optimization objectives uniformly.

11. *Mean tracking error* [182], P_{MTE}, records the Euclidean distance between the best candidate solution of each iteration and the global optimum, and is defined as

$$P_{MTE} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\sum_{j=1}^{n_x}(x_{best,j}^{t} - x_{opt,j}^{t})^2} \tag{2.20}$$

where $x_{best,j}^{t}$ is the j-th dimension of the best candidate solution at iteration t, and $x_{opt,j}^{t}$ is the j-th dimension of the location of the optimum at iteration t. Small values of P_{MTE} indicate that the distance from the best candidate solution and the global optimum is small, while larger values highlight that the best candidate is physically far away from the optimum (possibly on a local maximum/minimum).

The advantage of using a distance-based measure of optimality is that values are less affected by fitness rescaling (i.e. between landscape changes or across different instances of the same problem type). The frame of reference is the best individual

per iteration and algorithms are not penalized for exploring inferior (or distant) areas of the search space. P_{MTE} is very intuitive in how minimization and maximization problems are treated similarly - the distance to the optimum location is unaffected by the optimization goal itself. Focus on the best candidate solution also generally dampens the effect of outlier values. P_{MTE} is, however, affected by domain size changes as well as changes in the number of dimensions, since these changes will affect the Euclidean distance calculation.

The major disadvantage of P_{MTE} is that fitness information is not directly incorporated with distance information. This results in situations where good local optima that are located far away are unduly penalized, and steep gradient drop-offs of fitness values around the global optimum can lead to bad solutions near the optimum being erroneously viewed as good. The measure requires exact knowledge about the value of the optimum as well as the physical location of the optimum over time. The measure is undefined if multiple global optima locations are present, since the Euclidean distance to a single location is required by the measure. A salient point about P_{MTE} is that the measure depends on x^t_{best}, which makes algorithm iteration length an implicit parameter.

12. *Average minimum distance of population to optimum* [182], $P_{D_{min}}$, is the minimum Euclidean distance between the global optimum location and the closest candidate solutions in the population, defined as

$$P_{D_{min}} = \frac{1}{T}\sum_{t=1}^{T}\left(\min_{i\in[1,\dots,P]}\left(\sqrt{\sum_{j=1}^{n_x}(x^t_{i,j} - x^t_{opt,j})^2}\right)\right) \tag{2.21}$$

where P is the population size, $x^t_{i,j}$ is the j-th dimension of the i-th candidate solution at time t. $P_{D_{min}}$ measures how close any of the member of the population is to the global optimum location. Since $P_{D_{min}}$ is a distance-based measure, the measure shares most of the advantages and disadvantages of P_{MTE}. $P_{D_{min}}$, however, is not implicitly defined in terms of algorithm iterations, though care should still be taken when comparing population-based and non-population based algorithms.

13. *Normalized scores* [135], P_{norm}, normalizes the errors of optimization algorithms to the range of possible error values across multiple change periods and multiple

problem instances, and is defined as

$$P_{norm} = \frac{1}{n_c} \sum_{c=1}^{n_c} \frac{|e_{max}(c) - e(i,c)|}{|e_{max}(c) - e_{min}(c)|}, \quad \forall i = 1, \ldots, n_h \tag{2.22}$$

where $e(i,c)$ is the P_{MOE} (defined in equation (2.13)) of algorithm i in change period c, $e_{max}(c)$ and $e_{min}(c)$ are the largest and smallest errors among all algorithms in solving each landscape c, and n_h is the number of algorithms being compared. P_{norm} is seemingly easy to interpret: good performance is always characterized by values of $P_{norm} = 1$, while poor performance is characterized by values close to $P_{norm} = 0$. Equation (2.22) focuses on the error (which requires knowledge of the optimum) which results in minimization and maximization problems being treated uniformly.

A disadvantage is that P_{norm} can only show the relative performance of peer algorithms in the corresponding experiment, since scores are calculated relative to the other algorithms being measured. Any comparisons with one or more algorithms that were not part of the experiment would require a new normalization effort that includes all the new algorithms. This process will almost certainly result in each existing algorithm receiving new performance values in the range $[0, 1]$, which may cause confusion among observers. Outlier values affect P_{norm} in terms of its scaling: if at least one algorithm has highly deviating error values in one or more landscapes it will result in excessively large $e_{max}(c)$ and $e_{min}(c)$ values. This in turn will cause the non-outlier algorithms with relatively similar behaviors to take on performance values that are highly similar (and possibly statistically insignificantly different from one another). This situation is possible if even one algorithm deviates in even one change period. P_{norm} is not guarded against ill-defined results, since undefined values are possible if $e_{max}(j)$ and $e_{min}(j)$ are equal (for example if all algorithms have all their entities located on a plateau, which happens quite frequently in a problem such as the MPB if no candidate solutions are located on any of the peaks).

Discussion on Optimality Measures

The optimality measures discussed above tend to be the most commonly reported measures of optimality for SI algorithms that solve DOPs that can be found in the literature [40][93][119][136]. Each measure has various strengths and weaknesses, as outlined above, which makes it hard to decide which measure to use for experimentation.

A summary of the overall challenges faced by existing measures is as follows:

1. **Reliance on non-normalized values:** In many DOPs the range between the maximum and minimum possible evaluation function values can differ dramatically between environment changes. Scale changes like these affect the very meaning of any fitness or error value. Many measures that rely directly on the raw values can yield a spurious (i.e. statistically confounded) view of algorithm performance. Using non-normalized fitness or error values makes it hard to compare performance across landscape changes, problem instances, or problem types.

   For example, consider an environment with an initial MPB landscape that has an optimum value of 30 followed by another landscape where the optimum value is 60 (where both environments share a base function value of zero). The meaning of an error value of, say, 3 is 10% and 5% of the range of possible values, respectively. An error value of 3, therefor, carries twice the weight in the first landscape than in the second landscape.

2. **Reliance on relative normalization:** Many methods that apply normalization tend to produce relative performance scores that are only useful in the experiment at hand. The normalization process is usually not grounded in an absolute point of reference. Such measures make it difficult to compare performance directly against any published results without first re-running the algorithms behind the published results side by side with the new algorithms. An example is the *normalized scores* method, P_{norm}, discussed above which normalizes the performance of algorithms against each other. Another example is *window accuracy*, $P_{\text{WA}}(t)$, which normalizes performance relative to the largest and smallest fitness values observed during a subjective window of time.

   An example of a measure that normalizes performance against an objective point of

reference is the *relative error*, $P_{RE}(t)$, which expresses performance relative to the scale of the problem and time step t. This objectivity makes performance scores easier to compare across experiment boundaries.

3. **Reliance on parametric statistics:** Many measures use simple aggregation methods and parametric statistical methods (such as the mean or standard deviation) on measurement data that does not necessarily follow a Gaussian distribution. Applying these methods to results that violate the required data assumptions yields a warped view of performance. An example is the *average best error before change* measure (P_{ABEBC}) that reports the mean of all raw error values before each landscape change. Apart from the dependency on non-normalized error values, the measure loses all variance information by relying on the statistical mean of a series of measurement values that are not guaranteed to be normally distributed.

4. **Individual consideration of environment changes:** Related to the previous point, many measures do not capture the notion of variance in performance scores across subsequent search landscapes over time. An algorithm that performs erratically over time may receive the same performance score as another algorithm that shows little variation in performance over time. Studies that do want to focus on the volatility of algorithm performance across landscape changes within each algorithm run tend to use a second measure of variance (such as the standard deviation, or maximum versus minimum observed values). This complicates the statistical comparisons of algorithms since practitioners now need to consider two measures.

5. **Penalties for exploratory behavior:** Exploration and high population diversity are core tenets of good optimization algorithms aimed at solving DOPs, as discussed in section 2.5. Yet a surprising number of optimality measures penalize algorithms that exhibit exploratory behavior. An algorithm with mediocre performance that exudes little exploratory behavior can receive a better score than an algorithm with strong exploratory behavior that actually finds and tracks better solutions.

6. **Support for statistical significance testing:** Many measures are time depen-

dent and yield output values that are difficult to compare statistically. For example, measures such as the *best-of-generation* fitness, $P_{\text{BOG}}(t)$, or the *relative error*, $P_{RE}(t)$, yield a vector of scores for each time step t. The majority of studies in the literature subsequently apply simple aggregation methods or parametric statistical methods (such as the mean or standard deviation) in an effort to transform the series of values into a scalar value. This can lead to misleading results due to the reasons stated above.

Chapter 4 investigates the prevalence and applicability of these challenges in context of actual results generated in experiments. A new measure called the *relative error distance* is proposed in section 4.2 that aims to address these shortcomings.

2.7.3 Behavior-based Performance Measures

Behavioral measures are useful to determine *how* an algorithm operates and possibly even uncover *why* an algorithm behaves in a particular manner in certain situations. Nguyen *et al.* [136] stipulate that behavioral measures should capture aspects such as the diversity of the population over time, recovery of performance after environment changes, and the ease of reacquiring convergent behavior after environment changes. By recording these key aspects of how different algorithms operate it becomes possible to statistically analyze and correlate different types of behavior to increased or decreased algorithm performance.

The following popular behavioral metrics from DOP literature are discussed in more detail:

1. *Population diameter* [139], $B_{PD}(t)$, is the maximum Euclidean distance between any two candidate solutions in the population, defined as

$$B_{PD}(t) = \max_{(i \neq l) \in [1, \dots, P]} \left(\sqrt{\sum_{j=1}^{n_x} (x_{ij}^t - x_{lj}^t)^2} \right) \tag{2.23}$$

where x_{ij}^t is the j-th dimension of the i-th candidate solution at time step t. The focus of t can be at any time step of interest (such as the iteration before or after environment changes occur, all iterations, or other iterations of interest.).

The average of the series of $B_{PD}(t)$ can be computed if the average population diameter is needed.

$B_{PD}(t)$ measures the largest extent of distance between any candidate solutions in the population as a way to gauge the diversity of the population as a whole. $B_{PD}(t)$ is simple and intuitive to reason about. However, Olorunda and Engelbrecht [139] show that the measure is extremely sensitive to outliers and may yield an incorrect picture of the diversity of the population of an algorithm.

2. *Population radius* [139], $B_{PR}(t)$, is the radius from the center of mass of the population and the candidate solution that is furthest away from the center, defined as

$$B_{PR}(t) = \max_{i \in [1, \dots, P]} \left(\sqrt{\sum_{j=1}^{n_x} (x_{ij}^t - c_j^t)^2} \right) \tag{2.24}$$

where c_j^t is the j-th dimension of the population center position at time t, calculated as

$$c_j^t = \frac{1}{P} \sum_{l=1}^{P} x_{lj}^t \tag{2.25}$$

The focus of t can be at any time step of interest. As a measure of diversity, $B_{PR}(t)$ improves upon $B_{PD}(t)$ by taking the center of mass of the entire population into account. Olorunda and Engelbrecht [139] show that $B_{PR}(t)$ is, however, still susceptible to outlier candidate solutions, so the measure can give a misleading representation of diversity.

3. *Average distance around swarm center* [102], $B_{ADSC}(t)$, is the mean distance of the entire population around the center position of the population, defined as

$$B_{ADSC}(t) = \frac{1}{P} \sum_{i=1}^{P} \sqrt{\sum_{j=1}^{n_x} (x_{ij}^t - c_j^t)^2} \tag{2.26}$$

where c_j^t is the j-th dimension of the population center position at time t as calculated in equation (2.25). The focus of t can be at any time steps of interest.

Olorunda and Engelbrecht [139] show that $B_{ADSC}(t)$ is highly correlated with candidate solution dispersion in the population, which makes $B_{ADSC}(t)$ a good measure of population diversity. $B_{ADSC}(t)$ is relatively robust against outliers since the measure will only drift if extreme candidate solution positions deviate so far as to significantly affect the average distance calculation.

4. *Mean of the average distance around all population members* [139], $B_{ADP}(t)$, generalizes the average distance around the population center, $B_{ADSC}(t)$, by using each candidate solution as a center position and calculating the mean of all resulting average distances to each candidate solution in the population. $B_{ADP}(t)$ is defined as

$$B_{ADP}(t) = \frac{1}{P}\sum_{i=1}^{P}\left(\frac{1}{P}\sum_{l=1}^{P}\sqrt{\sum_{j=1}^{n_x}(x_{ij}^t - x_{lj}^t)^2}\right) \tag{2.27}$$

where the term

$$\frac{1}{P}\sum_{l=1}^{P}\sqrt{\sum_{j=1}^{n_x}(x_{ij}^t - x_{lj}^t)^2}$$

is the average Euclidean distance of the whole population around candidate solution $\mathbf{x}_i^t$. Olorunda and Engelbrecht [139] conclude that $B_{ADP}(t)$ is a more accurate measure of population dispersion than $B_{ADSC}(t)$, but that $B_{ADP}(t)$ is P times more expensive to calculate, making $B_{ADP}(t)$ practically unsuitable for larger populations (especially if the focus of t is every iteration).

5. *Moment-of-inertia* [126], $B_{MI}(t)$, measures how far the mass of the population of entities is away from the center of the population at time t, defined as

$$B_{MI}(t) = \sum_{i=1}^{P}\sum_{j=1}^{n_x}(x_{ij}^t - c_j^t)^2 \tag{2.28}$$

where c_j^t is the j-th dimension of the population centroid at time t as calculated in equation (2.25). The computational complexity of $B_{MI}(t)$ scales linearly with P, making $B_{MI}(t)$ a very efficient measure compared to other diversity measures that rely on pair-wise comparisons between candidate solutions. $B_{MI}(t)$ is, however,

not normalized across different population sizes, since different values of P will yield different values of $B_{MI}(t)$. $B_{MI}(t)$ is also susceptible to domain scaling, since large differences between the centroid position $\mathbf{c}^t$ and a candidate solution $\mathbf{x}^t$ will result in extremely large values of the term $(x_{ij}^t - c_j^t)^2$ which will be repeated P times.

6. *Stability* [181], $B_{stab}(t)$, measures how strongly environment changes affect the accuracy of the algorithm, defined as

$$B_{stab}(t) = \max\{0, P_{RE}(t-1) - P_{RE}(t)\} \tag{2.29}$$

where $P_{RE}(t)$ is the optimization accuracy defined in equation (2.18). An algorithm is considered stable if changes in the environment do not overly affect the optimization accuracy. $B_{stab}(t)$ is normalized since it relies on $P_{RE}(t)$, resulting in $B_{stab}(t) \in [0, 1]$, where values near 0 indicates high stability. Weicker [181] notes that $B_{stab}(t)$ should never be used in isolation since the measure does not reflect accuracy levels.

7. *Robustness* [151], $B_{rob}(t)$, is the ratio of the fitnesses of the best candidate solutions in the current and previous iterations, defined as

$$B_{rob}(t) = \min\left\{1, \frac{F(\mathbf{x}_{\text{best}}^t, t)}{F(\mathbf{x}_{\text{best}}^{t-1}, t-1)}\right\} \tag{2.30}$$

The initial value of $B_{rob}(t) = 1$. $B_{rob}(t)$ is not normalized since it relies on raw fitness values. This makes $B_{rob}(t)$ susceptible to incorrect readings in cases where landscape changes severely affect the maximum and minimum possible fitness values. $B_{rob}(t)$ is also undefined if $F(\mathbf{x}_{\text{best}}^{t-1}, t-1) = 0$. $B_{rob}(t)$ does not treat maximization and minimization problems similarly. $B_{rob}(t)$, however, does not require knowledge about the optimum nor when changes occur, making the measure widely applicable.

8. *Satisficability* (a term coined by Rand and Riolo [151]), $B_{sat}(\Theta)$, measures how often the algorithm can maintain a specified level of performance without dropping below a pre-set threshold, defined as (assuming maximization)

$$B_{sat}(\Theta) = \frac{1}{T}\sum_{t=1}^{T}\begin{cases} 1 & \text{if } F(\mathbf{x}_{\text{best}}^t, t) > (1-\Theta)F(\mathbf{x}_{\text{opt}}^t, t) \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

where $F(\mathbf{x}^t_{\text{opt}}, t)$ is the value of the global optimum at iteration t, and $\Theta \in [0, 1]$ controls the minimum fitness level the algorithm needs to achieve. $B_{sat}(\Theta)$ is an intuitive measure that makes it easy to compare algorithms solving the exact same instance of a problem.

However, it is hard to compare $B_{sat}(\Theta)$ across different problem instances: different DOP instances that have a disproportionate part of their search landscape values greater than Θ (i.e. large plateaus above the threshold of Θ) will tend to yield higher $B_{sat}(\Theta)$ values for that problem. $B_{sat}(\Theta)$ requires a domain-specific parameter, Θ, to define the minimum fitness threshold that the algorithm should exceed. The arbitrary choice of Θ makes $B_{sat}(\Theta)$ difficult to compare across different experiments without first understanding the domain context.

$B_{sat}(\Theta)$ requires modification between maximization and minimization problems. The measure, without modification, does not make sense in a maximization problem where the optimum value is negative, since subtracting $\Theta \times F(\mathbf{x}^t_{\text{opt}}, t)$ from the optimum will in fact *increase* the cut-off to a number between zero and the optimum (which will be impossible for the algorithm to reach).

9. *Reactivity* [181], $B_{react}(t, \varepsilon)$, measures an algorithm's ability to react quickly to changes. The ε-reactivity of an algorithm is defined as

$$B_{react}(t, \varepsilon) = \min \left\{ (t' - t) \left| \frac{P_{RE}(t')}{P_{RE}(t)} \geq (1 - \varepsilon)) \right. \right\} \cup \{T - t\} \tag{2.32}$$

where $t', t \in \mathbb{N}$, $t < t' < T$, $P_{RE}(t)$ is the *optimization accuracy* defined in equation (2.18), and the parameter ε sets the required relative accuracy threshold that must be met in order to consider the algorithm to have reacted to the landscape change. A lower value imply faster reaction times to changes. If there are no drops in (relative) performance after changes, the measure does not yield any insights into the algorithm's behavior. $B_{react}(t, \varepsilon)$ is undefined in any period where P_{RE} is zero. Knowledge about the optimum is required, but knowledge about environment changes is not needed.

10. *Absolute recovery rate* [137], B_{arr}, measures how long it takes for an algorithm

to start converging towards to a (possibly new) global optimum before the next change in the environment, defined as

$$B_{arr} = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{\sum_{j=1}^{p_i} (f_{best}^{i,j} - f_{best}^{i,1})}{p_i (f_{opt}^{i} - f_{best}^{i,1})} \tag{2.33}$$

where p_i is the total number of iterations in change period i, $f_{best}^{i,j}$ is the fitness of the best candidate solution found in change period i up until the j-th iteration of change period i, and f_{opt}^{i} is the global optimum value during change period p_i. B_{arr} is close to 1 if the algorithm is able to recover quickly after changes by immediately starting to converge on the optimum, and zero if the algorithm cannot recover at all. B_{arr} requires knowledge of the global optimum at every iteration of the search.

Discussion on Behavior Measures

The measures that characterize diversity through measuring the size of the extent of the population are deemed the most important in this study, since such measures will be used to characterize the diversity of various individual heuristics. Olorunda and Engelbrecht [139] conclude that the *population diameter* and *population radius* measures are both highly sensitive to outlier values that may subsequently misrepresent the true state of the population. They also conclude that the *mean of the average distance around all population members* measure is often too computationally expensive to calculate, especially for a larger population. They recommend the *average distance around swarm center* should be used, which balances computational complexity well and is more resilient to outlier values.

2.7.4 Statistical Comparison of Algorithm Performance

Researchers usually want to compare the performance of N different optimization algorithms that all solve a given set of benchmark problems. Their aim is to draw evidence-based conclusions about the superiority of one (or more) algorithm over others. Given the stochastic nature of CI algorithms, these conclusions are usually reached through statistical inference drawn from the analysis of empirical results. Generally, the hope is that strong enough statistical evidence can be gathered to make general conclusions

about the performance of algorithms on unseen problems. Recent literature reviews and surveys [40][93][119][136] highlight numerous studies that endeavor to evaluate the effectiveness of CI algorithms in solving DOPs.

Given the reliance on statistical significance testing, many recent studies have evaluated the applicability and validity of various statistical procedures in the field of CI [47][48][68][69][70]. Cruz *et al.* [40] and Nguyen *et al.* [136] confirm that nonparametric statistical significance testing is increasingly being used by more researchers to correctly compare algorithm performance in DOPs. All the studies above are unanimous in advocating the use of nonparametric statistical procedures to assess algorithm performance. The authors of the studies strongly recommend to avoid using parametric statistical procedures to analyze and compare the performance of CI algorithms (in DOPs as well as static problems). The overwhelming and repeated warning in these studies is that the conditions needed for parametric statistical methods to be applicable (independence, normality, and homoscedasticity [162]) are almost certainly violated by most CI algorithms. Nonparametric statistical procedures have less restrictive assumptions than their parametric counterparts and offer a more resilient analysis capability that avoids false discoveries.

Derrac *et al.* [48] provide an extensive overview of nonparametric procedures to correctly compare and contrast the performance of CI algorithms in various ways, namely:

- Pairwise comparisons between any two given algorithms.
- Multiple comparisons of one algorithm (the control method) against all $N-1$ other algorithms (*one versus all*).
- Multiple comparisons of each algorithm against all the remaining $N-1$ other algorithms (*all versus all*).

The rest of this section discusses the recommended methods from each of these comparison modalities as laid out by the studies above.

Pairwise comparisons between two algorithms

Pairwise comparisons assess whether two algorithms have significant differences between their respective result sets. This type of analysis is helpful to compare whether algorithm

a_1 is better than a_2 across a given set of repeated problem samples, or across different types of problems altogether. A situation where a pairwise test is especially useful is when the same algorithm is compared against itself across multiple samples with all things being equal except for a single algorithm configuration change. Such a test can provide insight into whether the configuration change consistently results in improved/worsened performance, along with an associated confidence in the observation (i.e. a statistical p-value [162]).

The *Wilcoxon signed ranks test* [183] (or simply the Wilcoxon test) is a pairwise nonparametric test that detects differences between the sample means of two algorithms. A requirement for the test is that all pairs of samples are dependent. When comparing the performance of two algorithms in DOPs, dependence between sample pairs implies using the exact same DOP search landscape with the same peaks and change dynamics over time, using the same population sizes, as well as having both algorithms start in the same starting locations. A related test for independent samples is the *Mann-Whitney U test* [117] where pairs of samples are independent from each other.

The Wilcoxon test computes the difference $d_i = m^i_{a_2} - m^i_{a_1}$ between the measurement scores m^i of the two algorithms a_1 and a_2 on the ith set of the total number of sample pairs. Normalized measurements should be used if the absolute differences between measurement instances are known to be different (which tends to be the case in DOPs). All differences d_i are ranked by absolute value with the best measurement receiving the rank of 1. Ties are broken by assigning the average rank, for example if two values are tied for rank 2, then both values receive the rank of 1.5.

The total rank of each of the two algorithms is computed by computing the sum of the ranks over all the samples. The total score R^+ is the sum of ranks for those samples where a_2 is better than a_1. The rank total R^- is the opposite sum over samples where a_1 performs better than a_2. Ties are split evenly across R^+ and R^- as follows:

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i)$$

$$R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i)$$

The Wilcoxon test statistic, z, is computed using the lesser of the two rank sums, T, where $T = \min\{R^+, R^-\}$. If the z value is less than the critical value for T for the

given number of samples, the null hypothesis is rejected with an associated p-value. The interpretation of the p-value is that the two algorithms a_1 and a_2 have noticeably different performance where the p-value is indicative of the probability that this performance difference is due to random chance (if in fact the null hypothesis holds true).

Pairwise comparisons between algorithms using nonparametric tests such as the Wilcoxon signed ranks test or Mann-Whitney U test are relatively intuitive for observers to understand, more appropriate to use in CI tests than the parametric alternative (t-tests), and outliers do not severely affect the outcome of the tests. However, a distinction should be made between *pairwise* tests and *multiple comparison* tests. When comparing multiple algorithms against each other, it is tempting to simply compute *Wilcoxon signed ranks test* outcomes between all possible pairs of algorithms. In such a situation the *family-wise error rate* (FWER)[162] is left uncontrolled, resulting in a cumulative error that dramatically increases the probability of false discoveries (*Type 1* errors). Garcia *at al.*[68] illustrate that a comparison between k methods at a significance level α has a probability of making a Type 1 error of $1-(1-\alpha)^{(k-1)}$. For $k=10$ and $\alpha=0.05$ the probability is 0.37, which is unacceptably high and will lead to false discoveries of seemingly superior algorithms.

The next sub-sections describe the use of multiple comparison tests and associated post hoc tests to correctly perform comparisons between multiple algorithms.

Multiple comparison among all algorithms

A frequent analysis performed by researchers is to compare one algorithm of interest (the *control method*) against $N-1$ other algorithms. The algorithm of interest could be a newly developed algorithm, a particular parameter or architecture configuration of a meta-heuristic, or any other criteria that isolates a particular algorithm so that it needs to be compared against the rest of the algorithms. Such an analysis comprises of two steps:

- Determine if at least two algorithms in the comparison set have significant differences in performance, or if the results of all algorithms are indistinguishable from each other.

- If significant differences are discovered between at least some of the algorithms in

the comparison set, apply a suitable *post hoc* test to determine which algorithms performed significantly better/worse than the control method.

The *Friedman two-way analysis of variance test* [65][66], or simply the Friedman test, and the associated Iman-Davenport extension [89] is one of the most well-known methods to determine if there are differences between the k algorithms being tested across n problem instances. The measurements of the algorithms are first ranked row-wise per problem, i.e. for each problem instance $i \in \{1, \dots, n\}$, each algorithm $j \in \{1, \dots, k\}$ is ranked from 1 to k (with 1 representing the best measurement). Ties are broken using average ranks. The average rank for each algorithm is then computed across all n problems. The result of the ranking procedure is a table of the ranks of all k algorithms against each other across all n problem instances, along with the average rank of each algorithm across all problems.

The average ranks table by itself is useful to gauge the overall relative performance of algorithms. However, the observer has no way of knowing if these differences are *significant* enough to conclude that any method is better/worse than any other method. The Friedman test proceeds by using the average ranks of all algorithms to compute the Friedman statistic F_f (or the Iman-Davenport modified statistic F_{ID}). The test statistic is subsequently compared against the relevant critical values for either statistic respectively to determine if the null hypothesis can be rejected. The result is a p-value that represents the probability that the observed difference between algorithms is due to random chance, assuming the null hypothesis in fact holds. Lower p-values indicate stronger evidence that differences exist.

Once the Friedman test determines that significant differences exist between the algorithms, the next step is to determine between which algorithms those differences exist. If the aim is to compare all other algorithms against the control method, any one of many suitable post hoc tests needs to be applied to obtain a p-value for each pairwise comparison between the control method and each of the other algorithms. A z-score is computed using the ranks table above which yields the *unadjusted p-value* for each comparison. Similar to the pairwise comparison methods, the FWER needs to be controlled if the control method has to be compared pairwise against $N - 1$ other methods. By adjusting the p-values to take the FWER into account, the resulting adjusted p-values can be compared against each other at a common significance level

α. This helps to prevent false discoveries of superiority from being made. The result of a post hoc test is a table of adjusted p-values of the pairwise comparisons between, respectively, a control method versus all $N-1$ other methods, or all $N(N-1)/2$ pairs of algorithms.

Various post hoc tests exist that differ in the way that the p-values are adjusted for $N-1$ comparisons, including the Bonferroni-Dunn [54], Holm [82], Holland [81], Finner [62], Hochberg [80], Hommel [83], Rom [155], and Li [109] procedures. Garcia *et al.*[68] and Derrac *et al.*[48] show that the Holm, Hochberg, Hommel, Holland and Rom tests all exhibit comparable levels of power. The studies advise that the Bonferroni-Dunn test should never be used, since it generally lacks in power versus the other procedures. The Finner test is noticeably better than the Holm, Hochberg, Hommel, Holland and Rom tests and is generally recommended [48][68].

Multiple comparisons among all algorithms

When the aim is to compare all N algorithms against each other instead of against a control method, an omnibus test such as the Friedman test should still be used to assess if the various algorithms show significant differences between at least two of the algorithms. If such differences are found, a suitable post hoc test can be applied to find the pairwise combinations across all algorithms where noticeable differences exist.

The Holm test discussed for the $N-1$ comparisons case above can be applied to perform $N(N-1)/2$ comparisons as well. Other suitable procedures include the Nemenyi procedure [131], Shaffer's static and dynamic procedures [159], as well as the Bergmann–Hommel procedure [8]. Derrac *et al.* [48] warn against using the Nemenyi procedure since it is very conservative and many clearly obvious differences will not be detected correctly. Derrac *et al.* comment that Shaffer's method has similar complexity to Holm's procedure, yet offers substantial benefits due to the manner in which the procedure reasons logically about the relationship between multiple hypothesis. Lastly, the Bergmann–Hommel procedure is computationally expensive and non-intuitive to explain, hence it is not recommended for general use unless the situation requires it (specifically, if the observed differences between algorithms are not very significant).

The result of an omnibus test such as Friedman combined with an $N(N-1)/2$ post

hoc procedure is a table of all pairwise comparisons between all algorithms along with the adjusted p-values for each comparison (that corrects the FWER as explained above). A researcher can assess which algorithms are superior/inferior/similar to each other by examining the adjusted p-values at a chosen level of significance i.e. $\alpha = 0.05$.

2.8 Summary

DOPs are challenging to solve since their search landscapes constantly change over time. Many CI algorithms are unable to adapt to such changing conditions, which in turn can lead to inferior solutions. Specialized CI algorithms solve DOPs by relying on one or more strategies to cope with the changing problem, namely *introducing/maintaining diversity*, *detection of environment changes*, *reacting to environment changes*, *use of explicit/implicit memory*, *multiple populations*, *predicting changes*, and using *self-adaptive mechanisms*. The CI sub-fields of SI and EC have developed numerous meta-heuristic implementations that address DOPs, and many good reviews and surveys summarize the state of the art [40][53][93][119][136].

The moving peak benchmark is a well-known DOP and is considered the most widely used benchmark function generator for dynamic environments in the literature. Duhain and Engelbrecht [53] combine the DOP classifications of Eberhart *et al.* [56][84] and Angeline [4] with spatial and temporal change severity classes (*quasi-static, abrupt, progressive, and chaotic*) into 27 unique DOP types. Parameter considerations for the moving peaks benchmark generator allow each of these different types of dynamics to be faithfully recreated in experimentation.

The comparison of the performance of stochastic CI optimization algorithms is a complex task which requires careful thought. Simpler measures of performance and behavior that work well for static optimization problems do not generally work well in dynamic environments. Various measures of performance and behavior have been tailored to correctly characterize the workings of algorithms that solve DOPs. Additionally, the use of parametric statistical tests to compare the performance of algorithms is generally regarded by experts as an incorrect approach to analyze CI algorithms. The use of nonparametric procedures is strongly encouraged.

The next chapter discusses hyper-heuristics as an approach to solve optimization

problems, and explains how selection hyper-heuristics adapt the search in real-time by selecting the most applicable heuristics to apply at time t.

Chapter 3

Selection Hyper-heuristics

> *"Everyone thinks of changing the world but no one thinks of changing himself."*
> – Leo Tolstoy

This chapter gives an overview of hyper-heuristics as an approach to solve optimization problems. A brief outline and classification of the field of hyper-heuristics is given along with related research as it pertains to selection hyper-heuristics. Hyper-heuristics are contrasted against other control adaptation approaches from the fields of SI and EC.

3.1 Introduction

The fields of *Operations Research*, *Computer Science*, and *Artificial Intelligence* have collectively produced a family of optimization methods called *hyper-heuristics* [24]. A hyper-heuristic adapts the optimization process by choosing the sequence in which low-level heuristics should be applied to a problem over time. Generally, the hope is that intelligent adaptation of the optimization process will allow superior solutions to be found for the current problem under consideration. Section 3.2 discusses the need for control adaptation methods in more detail.

A top-level split in the categorization of hyper-heuristics is the difference between *generative* and *selection* hyper-heuristics: generative methods generate new heuristics while solving the problem in contrast to selection hyper-heuristics that select the most

suitable heuristic from an pool of existing heuristics. Section 3.3 presents an overview and classification of the different types of hyper-heuristic approaches and related work around applying selection hyper-heuristics to solve DOPs.

Section 3.4 focuses on the use of selection hyper-heuristics in a multi-population setting where multiple candidate solutions are improved concurrently. Related work is presented along with an outline of the *heterogeneous meta-hyper-heuristic* (HMHH) framework for hyper-heuristics by Grobler and Engelbrecht [75][76][74]. The challenges faced when using hyper-heuristics with a population of multiple candidate solutions are explored in section 3.4.1. The visibility of information between different heuristics within the population is discussed in section 3.4.2. The management of heuristic diversity is explored in section 3.4.3. Strategies to decide when to trigger a change in heuristics for some/all of the members of the population are discussed in section 3.4.4. Section 3.4.6 discusses the process logic used by selection operators in deciding which heuristics to apply to candidate solutions. Multiple examples of both common and recently proposed selection operators are outlined.

Finally, section 3.5 discusses the similarities that hyper-heuristics share with other control adaptation techniques and positions why hyper-heuristics are different.

3.2 The Need for Adaptation in Optimization

The word *"heuristic"* is derived from the ancient Greek word "*heuriskein*" for "to discover". At their core, heuristics embody the strategies and techniques to exploit known information in control problem-solving processes in both machines and humans [146]. Over time researchers have created *meta-heuristics* as parameterized templates of heuristics in a bid to make heuristics more generally applicable to a larger array of problems. The definition of parameters for meta-heuristics has inadvertently led to something called the "tuning problem" [11][57][58][97] which entails finding those parameters that yield the best performance for any given meta-heuristic.

Section 3.2.1 discusses the static and dynamic versions of the tuning problem and how the practice of meta-heuristic parameter tuning relates to DOPs. Section 3.2.2 provides an overview of the *No Free Lunch* theorems for optimization [184] and how these theorems are practically less of an influential factor in reaching higher levels of

general performance than many practitioners believe.

3.2.1 Meta-heuristics Parameter Tuning

Sörensen and Glover [164] define a *meta-heuristic* as a high-level problem-independent set of guidelines and strategies to design heuristic optimization algorithms. Sörensen [163] reaffirms that a meta-heuristic is less of an explicit algorithm and more of a consistent set of complimentary ideas, concepts, and operators. Sörensen [163] argues that different meta-heuristics are congruent to different *styles* of cooking, such as French or Chinese cuisine, and not *recipes* for specific dishes per se. Asking "*are SI approaches better than EC approaches?*" is akin to asking "*Is Chinese cooking better than French cooking?*" The answer invariably is "*it depends on who is dining*" since each meta-heuristic brings a unique set of operators and design decisions to the table.

Birattari *et al.* [11] equate meta-heuristics to being *templates*, and that specific parameter choices are required to turn a meta-heuristic into a tangible algorithm. Different parameter choices dramatically alter the operation of any meta-heuristic. What the "best" parameter choice for any meta-heuristic is, can be a difficult question to answer. Karafotias *et al.* [97] present a survey of adaptive parameter control methods. The survey distinguishes between the *parameter tuning problem*, as the upfront, stationary task of finding appropriate values for meta-heuristic parameters, and the *parameter control problem*, as the ongoing, non-stationary task of adapting meta-heuristic parameter values over the run of the algorithm.

The *parameter tuning problem* is deemed essential by Karafotias *et al.* to the successful deployment of any meta-heuristic. Eiben and Smit [58] present a taxonomic breakdown of over 30 different parameter tuning methods, and propose a conceptual framework of tuning methods and a tuning-aware experimental methodology. However, meta-heuristic parameter tuning alone will not suffice in DOPs. Leonard and Engelbrecht [103] show that it is impossible to statically optimize PSO parameters for any given DOP. Each environment change results in a different optimal PSO parameter configuration. Intuitively, this finding extends to EC approaches and implies that a differently tuned method (or even a different method entirely) may be needed in each subsequent environment landscape [46]. These examples emphasize the importance of adapting algorithm

behavior over time *while* the problem is being solved (i.e. addressing the *parameter control problem* using a non-stationary tuning approach).

Karafotias *et al.* [97] note that the *parameter control problem* has not been adequately solved yet. The authors discuss the "*patchwork problem*" which describes the difficulty in creating good parameter combinations when haphazardly combining multiple individual parameter control methods into a single process. The resulting system is likely sub-optimal. Karafotias *et al.* recommend that future research should focus on reducing the patchwork problem by developing techniques that operate on multiple parameters simultaneously. Instead of (self-)adapting every individual parameter or component of meta-heuristic algorithms independently at the lowest levels of granularity, research should focus on swapping out entire components of well-proven units of functionality. Sörensen [163] comes to a similar conclusion, stating that a component-based view of meta-heuristics (where operators from diverse meta-heuristic frameworks can be combined into more powerful methods) is key to producing deep insights into why meta-heuristics work.

3.2.2 The No Free Lunch Theorems for Optimization

Wolpert and Macready [184] published the "No Free Lunch" (NFL) theorems for optimization in 1997 which, informally, state that all optimization algorithms have equal performance when evaluated over all possible problems (for any performance measure). The implication is clear: no method can be considered "generally better" than any other method in domains where the NFL theorems hold. Naturally, the NFL theorems cause a stir among researchers that aim to develop optimization algorithms which try to be "generally better" than other methods. According to the NFL theorems, even random search *will outperform* even the most sophisticated optimization algorithm in *some* subset of problems.

Uneasiness around the NFL theorems prompted researchers to further investigate in exactly which domains the NFL theorems hold. A given set of fitness functions (i.e. problems) is *closed under permutation* if, for every function f in the set, all possible rearrangements of mappings from search space values $\mathbf{x}_i$ to function values $f(\mathbf{x}_i)$ are also contained as another function in the set. Schumacher *et al.* [157] show that any

two algorithms will have the same performance over a set of fitness functions (for any performance measure) *if an only if* the set of fitness functions is closed under permutation. However, Igel and Toussaint [88] show that the sets of fitness functions that are closed under permutation comprise of a small part of the whole. Auger and Teytaud [5] show that the NFL theorems do not hold generally in continuous domains. Alabert *et al.* [3] sharpen the results of Auger and Teytaud by proving that there are indeed no NFL theorems for functions in continuous domains, except for a few extreme theoretical edge cases which require additional technical conditions that are simply too restrictive to be found in practice. In effect, it is practically impossible to find a real-world problem that meets the necessary conditions for the NFL theorems to hold.

Poli and Graff [149] show that the NFL theorems do not automatically apply to meta-search methods such as hyper-heuristics. Firstly, Poli and Graff illustrate how a hyper-heuristic approach makes no sense when the set of problems is closed under permutation: the heuristic search space landscape is flat if the set of problems under consideration is closed under permutation. However, given the results in the previous paragraph, Poli and Graff argue that the odds are exceedingly rare of encountering a set of problems for which there is no gain in using a hyper-heuristic (i.e. problems that are closed under permutation). Poli and Graff present a generalized counting argument that shows that for problems with n distinct fitness values $\{f_1, f_2, \ldots, f_n\}$ at points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ there are $n!$ possible permutations of all fitness assignments across all points. If the problem set under consideration contains fewer than $n!$ problems, then the set is not closed under permutation and the NFL theorems do not apply.

The results of Auger and Teytaud, Alabert *et al.*, and Poli and Graff imply the possibility that, in any practical situation, an intelligent selection hyper-heuristic could potentially improve performance over using heuristics in isolation. A recent algorithm selection survey by Kerschke *et al.* [98] echoes these findings in how the necessary conditions for the NFL theorem to hold are simply not found in problems of interest. Kerschke *et al.* remark that increased performance is possible by exploiting the complementary strengths of a set of algorithms through automatically selecting the most appropriate algorithm that is expected to perform best. Kerschke *et al.* list hyper-heuristics as one such promising field to solve the *online per-instance algorithm selection problem* and increase performance. It is this practical perspective on the theoretical underpinnings

of the discipline of optimization that encourages research into methods such as hyper-heuristics.

3.3 Overview of Hyper-heuristics

This section discusses the general classification of hyper-heuristics and how selection hyper-heuristics relate to generative hyper-heuristics, the domain barrier that separates the *problem space* (i.e. solution space) from the *heuristic space* (i.e. space of algorithms and methods), and the application of selection hyper-heuristics to DOPs.

3.3.1 Classification of Hyper-heuristics

Generally speaking, the field of *hyper-heuristics* seeks to automate the process of selecting, combining, generating and/or adapting a set of multiple simpler *heuristics* to solve a given problem in a problem-independent way. A key driver is to combine the strengths and weaknesses of multiple algorithms as optimally as possible.

The term *hyper-heuristic* was first used by Cowling *et al.* [38] in 2000, but early work in the field dates back to the probabilistic scheduling rules of Fisher and Thompson [63] in 1961. The roots of hyper-heuristics lie in disciplines such as job scheduling, time tabling, routing problems, and combinatorial optimization. Early studies employ hand-crafted operators that manage a pool of highly domain dependent heuristics (usually also manually created by humans). Recent works have used EC and SI meta-heuristics both as low-level heuristics and/or hyper-heuristic operators [24].

Burke *et al.* [26] review hyper-heuristic literature for both combinatorial and continuous optimization and distinguish between *selection* and *generative* hyper-heuristics. *Selection* hyper-heuristics use predefined selection operators inside a hyper-heuristic framework to choose a suitable existing heuristic to apply to a problem at time t. *Generative* hyper-heuristics iteratively evolve customized heuristics (mostly via *genetic programming*) that are tailored to a domain (or even a specific problem instance).

In addition to hyper-heuristics being either *selection* or *generation* based, Burke *et al.* [26] identify two additional dimensions to hyper-heuristic solutions as shown in figure 3.1, namely:

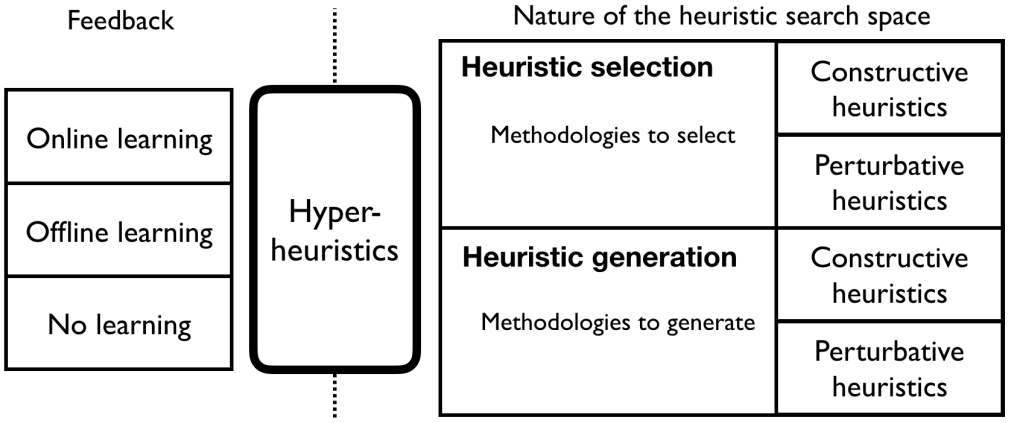


Figure 3.1: Hyper-heuristics classification as presented by Burke *et al.* [26]

1. The nature of how a hyper-heuristic learns from domain feedback is either via *on-line* learning, where the hyper-heuristic (self-)adapts while solving the problem, *off-line* learning, where the hyper-heuristic has access to training examples from the domain beforehand, or no learning at all.

2. Whether the heuristics are *constructive* that grow viable complete solution(s) from empty solution(s), or *perturbative* that start with full solution(s) which are incrementally improved.

A wide range of hyper-heuristic approaches are possible that combine different styles of learning and either heuristic selection or generation methods to manage a set of either constructive or perturbative heuristics. Various reviews of the state of the art of hyper-heuristics [24][26][25] provide insights into the most prominent examples in literature. The focus of this thesis is restricted to the application of *selection* hyper-heuristics to meta-heuristics for DOPs, and generative hyper-heuristics are out of scope. Furthermore this thesis will investigate hyper-heuristic selection methods that are classified by Burke *et al.* [26] as employing either *on-line* learning or *no learning*. All meta-heuristics in the heuristic pool will be examples of *perturbative* heuristics under the same classification.

3.3.2 Problem Space versus Heuristic Space

A *hyper-heuristic* is defined by Chakhlevitch and Cowling [30] as a high-level control mechanism that uses limited problem-level knowledge to search a set of low-level heuristics for good *methods* and not good solutions. The *problem space* comprises the domain of the objective function of the optimization problem. Any point in the problem space represents a solution to the optimization problem. The search space of all applicable methods and their associated utility at time t is called the *heuristic space*. Any point in the heuristic space represents a proposal of which heuristic(s) should be applied to improve the problem space solution vectors. The domain of available heuristics is also referred to as the *pool* of heuristics. Hyper-heuristics offer an attractive "off-the-peg" control mechanism that does not directly rely on a full understanding of the problem space, but instead uses continual performance feedback to search the heuristic space for the best method to use at time t.

Feedback about the performance of the heuristics currently in use enables the hyper-heuristic to choose the most appropriate heuristics to apply next. Burke *et al.* [26] present a conceptual framework (shown in figure 3.2) of how selection hyper-heuristics operate in the heuristic space to find the best methods to apply in the problem space. Hyper-heuristics can employ either *single-point* search, where a single candidate solution is improved in an iterative manner, or *multi-point* search, where a population of multiple candidate solutions are improved together over time [24][26].

After a heuristic is applied to a problem, the hyper-heuristic can perform a *move acceptance* step to decide if the heuristic's newly proposed solution should be accepted or rejected. Move acceptance strategies are critical in single-point search to ensure that the single candidate solution steadily improves over time [24][26]. Move acceptance can also be applied to multi-point search hyper-heuristics. Different move acceptance strategies exist, including simple deterministic strategies such as *only improving* that only accepts improving changes, *improving and equal* that accepts moves of the same or better quality, and *all moves* that accept any moves [38]. More sophisticated non-deterministic move acceptance strategies include tabu search [29], simulated annealing [7], and late acceptance [141], among many other techniques.

Single-point search hyper-heuristics offer the advantage that a single candidate solu-

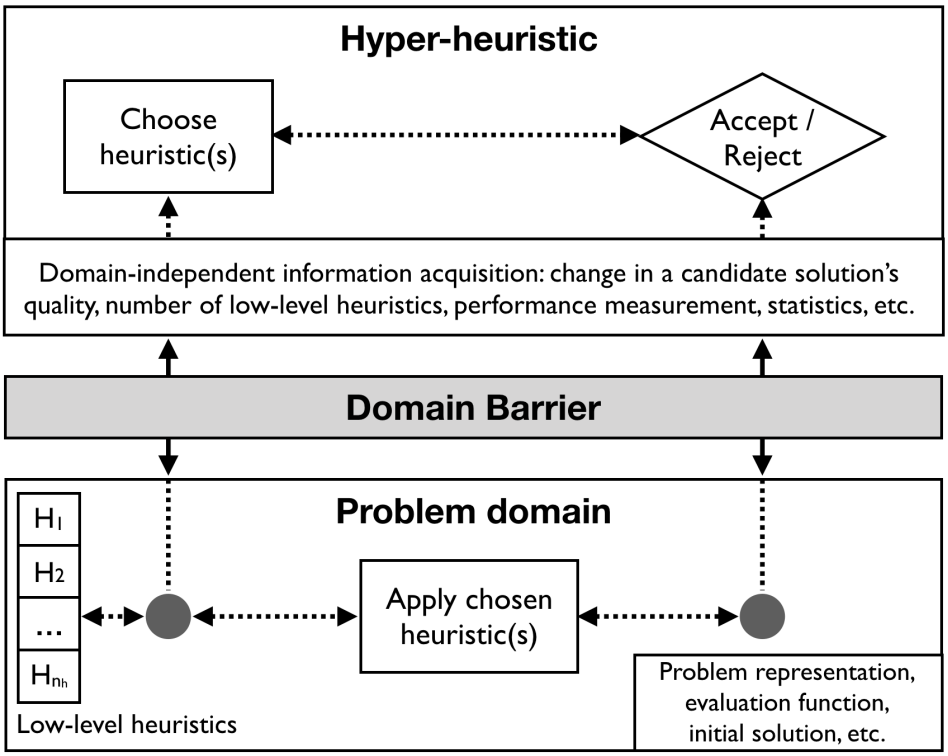


Figure 3.2: The conceptual framework for selection hyper-heuristics as presented by Burke *et al.* [26].

tion can be extensively explored (using either perturbative or constructive heuristics), while a move acceptance operator ensures steady progression towards an increasingly better solution. However, since there is only a single candidate solution being improved, single-point search hyper-heuristics are unable to simultaneously explore different areas of the problem space. Any such alternative exploratory complexity would need to be managed by the move acceptance operator. Multi-point search hyper-heuristics, on the other hand, solves this problem by maintaining a population of multiple candidate solutions and allowing meta-heuristics to exploit the information of other candidate solutions. However, move acceptance strategies are harder to manage in multi-point search, since using move acceptance criteria to revert members of the population back to prior states may inadvertently affect the operation of meta-heuristics.

Ongoing research into hyper-heuristics continues to refine the working definition, goals, and constraints of what constitutes a hyper-heuristic approach. A growing num-

ber of researchers are revisiting the assumption that hyper-heuristics require a strict separation between the problem space and the heuristic space. Recently, Swan *et al.* [168] show that a "maximally restrictive" barrier between the hyper-heuristic and problem domain is counter-productive. The authors argue that certain types of a priori problem information (what the authors call "*analytic* information") can be exploited by a hyper-heuristic in a problem-independent manner to aid in the heuristic selection process. This broader view allows the incorporation of heuristic-to-domain mapping information, declarative domain descriptions, and constraint languages as data instead of code changes [168]. A research agenda has been defined to promote this view and gain consensus across the field as to what such an architectural vision for hyper-heuristics would look like [167][168].

The next section discusses how selection hyper-heuristics can be applied in continuous dynamic environments.

3.3.3 Selection Hyper-heuristics for Real-valued Dynamic Environments

DOPs change as time goes by, which makes it hard for non-specialized optimization algorithms to cope with the changing search environment. Given the ability of selection hyper-heuristics to continually select the most appropriate low-level heuristic to apply to a problem, the natural question is if hyper-heuristic approaches could adapt to the changing DOP and avoid the pitfalls faced by other optimization algorithms.

A number of studies investigate the application of hyper-heuristics to DOPs . Özcan *et al.* [143] apply a greedy selection-based hyper-heuristic to manage a mix of simple mutators and hill climbers. Kiraz *et al.* [100] investigate the performance of a number of hyper-heuristics that were proposed by Cowling *et al.* [38] on DOPs, including basic methods such as *simple random*, *random descent*, *random permutation*, *random permutation descent*, and *greedy* selection. Kiraz *et al.* also investigate the performance of more complex hyper-heuristics on DOPs, such as *Choice function*, where selection of a heuristic is based on the heuristic's historical performance, performance as a successor to other heuristics, and the number of iterations since the heuristic was previously applied [38]. Kiraz *et al.* also apply a *reinforcement learning* hyper-heuristic based on the work

by Özcan *et al.* [142], where each heuristic carries as rank value that gets updated based on the success of the heuristic. Worsening moves are penalized while improving moves are rewarded using a set rate. Simple Gaussian mutation operators (GMOs), where candidate solutions are modified by adding unbiased random noise sampled from a Gaussian distribution with zero mean and a specified standard deviation value σ, are used as the low-level heuristics. Multiple implementations are possible, based on the specific choice of σ, and it is the task of the hyper-heuristic to select the most appropriate operator at time t.

In a later study, Kiraz *et al.* [99] investigate an ant-based hyper-heuristic that maintains pheromone concentrations between all pairs of low-level heuristics, again managing Gaussian mutator heuristics. Uludağ *et al.* [173] present a framework that hybridizes population based incremental learning (PBIL) algorithms and selection hyper-heuristics. Topcuoglu *et al.* [171] combine various hyper-heuristic operators with a memory/search algorithm to increase the effectiveness of memory/search.

These early studies all rely on relatively simple heuristics such as Gaussian mutation operators or simple hill-climbers. While the studies show how hyper-heuristics are able to adapt to the state of the problem space at time t, the studies do not investigate the use of domain-specific heuristics that are specialized to solve DOPs. Almost all of the studies listed above also rely on single-point search hyper-heuristic methods. More recent studies by Van der Stockt and Engelbrecht [175][174][176] investigate the application of multi-point search hyper-heuristics to manage a pool of DOP-specific meta-heuristics.

Section 2.5 in chapter 2 outlined a number of building blocks that meta-heuristics focusing on solving DOPs should employ. Section 2.6 presented a number of successful DOP-specific meta-heuristics that differ in their implementation of each DOP building block. This thesis shows that selection hyper-heuristics offer a viable method to intelligently combine multiple well-understood meta-heuristics for DOPs to improve the performance of the optimization process beyond what any of the individual meta-heuristics can manage in isolation.

The next section explores how population-based meta-heuristics for DOPs can be used in conjunction with a selection hyper-heuristic framework.

3.4 Selection Hyper-heuristics for Population-based Algorithms

Section 3.4.1 expands on the notion of multi-point search hyper-heuristics and presents various multi-point search frameworks. Section 3.4.2 discusses the difference between global and island neighborhood topologies for a pool of heuristics. Section 3.4.3 outlines how the spread of heuristics assigned to entities can be measured using heuristic diversity measures. Section 3.4.4 outlines the considerations to take before triggering a change in the heuristics assigned to manage the candidate solutions. Section 3.4.5 discusses how algorithm state information is maintained across heuristics over time. Finally, section 3.4.6 discusses examples of selection operators that perform heuristic selection in different ways. Lastly, section 3.4.7 provides an overview of how to select a set of complementary heuristics that maximizes the effectiveness of a hyper-heuristic.

3.4.1 Multi-point Search Hyper-heuristics

Multi-point search hyper-heuristics improve multiple candidate solutions concurrently, in contrast to *single-point* search hyper-heuristics that improve a single solution. Burke *et al.* [24][26] reveal in their survey that the majority of early selection hyper-heuristics employ single-point search, while more recent studies tend to use multi-point search. Key examples of approaches that employ multi-point search are outlined in the paragraphs below.

A personnel scheduling problem is a combination problem where the aim is to find the optimal assignment of personnel to shifts, given a set of constraints [24]. Cowling *et al.* [37] introduce a system called *Hyper-GA* to address the personnel scheduling problem. Hyper-GA uses a genetic algorithm to systematically learn which heuristics to select next. The practitioner provides a set of low-level problem-specific heuristics and Hyper-GA will find a solution to the problem by learning the best ordering of the heuristics. Hyper-GA is extended by Han and Kendall [77] to use guided mutation and dynamic chromosome lengths which allows the system to make the evolution of heuristic sequences more effective.

The ant colony algorithm [49] is used as a hyper-heuristic by a number of authors.

Generally, an ant-based hyper-heuristic finds the best sequence of applying heuristics by traversing the fully connected graph between all heuristics. The directed edge between any two heuristics h_i and h_j represents the suitability of applying h_j after h_i. Each ant traversing the graph represents a specific sequence of heuristics. Ants may move from vertex to vertex in the graph to select and apply a single heuristic at a time, or the entire sequence of moves made by the ant can be learned.

Burke *et al.* [27] investigate how ant algorithms can construct the best sequences of heuristic moves to solve a project presentation scheduling problem. Chen *et al.* [31] apply ant-based approaches to solve the traveling tournament problem. Ren *et al.* [153] replaces the fully connected graph topology with a bipartite graph structure drawn from the Cartesian product between two heuristic sets. Ren *et al.* define one set to be exploitation heuristics while the other set contains exploration heuristics. This system is used to solve the p-median problem.

The use of EC methods as hyper-heuristics has also attracted attention. Vrugt and Robinson [177] propose an evolutionary optimization-based method called AMALGAM. The focus of AMALGAM is to manage a pool of multi-objective-specific population-based heuristics to solve multi-objective real-valued optimization problems. AMALGAM combines multi-method search and self-adaptive offspring creation techniques where the framework manages the proportion of new solutions that are retained from each managed heuristic. Cobos *et al.* [36] employ a mix of evolutionary approaches to cluster documents together with multiple types of selection and move acceptance strategies.

Certain hyper-heuristic frameworks endeavor to run the managed algorithms in true parallel mode across multiple worker nodes. Crainic and Toulouse [39] present a parallel hyper-heuristic that uses multiple threads to allow several heuristics to guide the search using information sharing. Biazzini *et al.* [10] propose a distributed island model framework that can combine multiple real-valued optimization algorithms running on different nodes. Meignan *et al.* [121] propose an agent-based hyper-heuristic for vehicle routing that is both distributed and self-adaptive. Multiple agents explore the search space jointly using a set of operators.

Heterogeneous Meta-Hyper-Heuristics

Grobler *et al.* [75][76][74] present the *heterogeneous meta-hyper-heuristic* (HMHH) as a selection hyper-heuristic framework that manages a pool of population-based meta-heuristics. Each heuristic is a meta-heuristic algorithm configuration comprising of specific logic, parameter values, operator functionality, and other design decisions. HMHH treats each heuristic as a "*sealed unit*" and never adapts any of the aforementioned components. This approach is analogous to practitioners providing manually crafted domain-specific low-level heuristics, except that the heuristics in HMHH are specifc instances of meta-heuristics that are suitable in the problem domain. The intention is to have a pool of several different (yet specific) algorithm configurations with known behaviors.

Adaptation in HMHH occurs by assigning a specific heuristic to manage and modify each candidate solution (or *entity*) in the population. Different heuristics, in essence, become distinct behaviors that yield different outcomes when applied to any given candidate solution. Generally, the position and fitness of a entity is noticeably altered in different ways depending on which heuristic acts upon the entity. For example, an entity modified for one iteration by a DE algorithm variant will generally have a noticeably different output candidate solution compared to if the entity was instead modified by a PSO or GA variant. HMHH strives to give the most promising heuristics every opportunity to succeed by letting more entities be updated by the most suitable heuristics.

HMHH is shown in algorithm 4 with parts of the original notation adapted to align with section 3.4.6. Every k iterations HMHH employs a selection operator, ς, to assign entities in the parent population E to heuristics. The performance feedback of each heuristic h_m, namely Q_{δ_m}, may be used by ς as part of the deliberation of determining entity-to-heuristic allocations.

The selection operator of the HMHH framework determines which entity-to-heuristic mapping is expected to yield the best performance. Different types of selection operators implement unique ways of allocating heuristics to entities that

1. may or may not rely on *performance feedback* from entities to influence the selection process,

2. may or may not employ *learning* or *memory mechanisms* to adapt selection sensi-

Algorithm 4 Heterogeneous Meta-Hyper-Heuristic [76]

$E \leftarrow$ Initialize parent population of n_s solution entities.
$h_j(t) \leftarrow$ the heuristic algorithm applied to entity e_j at iteration t.
$k \leftarrow$ the number of algorithm iterations between entity-to-heuristic assignments.[1]
for all entities $e_j \in E$ **do**
  $h_j(1) \leftarrow$ choose random initial heuristic algorithm for e_j.
end for
$t = 1$.
while a stopping condition is not met **do**
  for all entities $e_j \in E$ **do**
    Apply $h_j(t)$ to entity j for k iterations.[2]
    $Q_{\delta m}(t) \leftarrow$ total improvement of entities assigned to h_m for the last k iterations.[3]
  end for
  for all entities $e_j \in E$ **do**
    $h_j(t+k) \leftarrow$ Select next heuristic for entity e_j using selection operator $\varsigma(e_j, Q\delta_m(t))$.[4]
  end for
  $t = t + k$.
end while

Notes:

1. $k = 5$ is used in [76].

2. In the *island* neighborhood topology, all the entities assigned to h_m collectively form a distinct sub-population $s_m \subset E$ that h_m operates on exclusively for k iterations. In the original *global* topology, h_m has read-only access to all entities $e_j \in E$, but can only alter the entities assigned the h_m, namely $s_m \subset E$. Neighborhood topologies are an extension to HMHH made in this thesis, and is not part of the original algorithm listing in [76].

3. In [76], $\varsigma(e_j, Q\delta_m(t))$ is rank-based tabu search [28]. This thesis expands HMHH with many other hyper-heuristic selection operators.

4. Entities may require additional heuristic-specific state information to operate as part of specific types of meta-heuristics (see section 3.4.5).

tivity to the problem under consideration, and

3. may apply *deterministic* or *stochastic* selection schemes to allocate heuristics to entities.

The choice of selection operator allows HMHH to exhibit radically different heuristic

allocation behavior.

Extending HMHH for Dynamic Environments

This thesis uses the HMHH framework by Grobler *et al.* [75][76][74] as the base to explore the effectiveness and behavior of various different hyper-heuristic selection operators. HMHH was originally devised for static environments. Van der Stockt and Engelbrecht [174][175][176] investigate the effectiveness of various HMHH selection operators to manage a pool of DOP-specific meta-heuristics across 27 different types of DOPs (as discussed in section 2.3.1 in chapter 2). Van der Stockt and Engelbrecht [176] show that using HMHH to manage DOP-specific meta-heuristics generally increases performance over using any of the individual meta-heuristics in isolation. Additionally, Van der Stockt and Engelbrecht [176] show that different selection operators for HMHH have dramatically varied performance outcomes.

Van der Stockt and Engelbrecht [175] extend HMHH to support two types of *neighborhood topologies*, namely a *global* topology where all heuristics always have access to the information of all entities across the entire HMHH parent population, and an *island* topology where entities have limited visibility of only the information of other entities that share the same assigned heuristic. Section 3.4.2 expands on the notion of *neighborhood topologies* in HMHH.

The differences in heuristic space diversity and heuristic-to-entity allocation rates across different HMHH selection operators are also explored by Van der Stockt and Engelbrecht [176]. Many of the best performing selection operators show strikingly different behavior profiles: certain methods frequently assign new heuristics to a small to moderate percentage of the parent population, while other methods infrequently assign almost all entities a single heuristic. Behavior in the study is based on aggregate values of measures. This thesis investigates the behavior of HMHH selection operators over time. Section 3.4.3 discusses heuristic space diversity and heuristic allocation rates in more detail.

The original HMHH algorithm selects new heuristics for all entities every k iterations. The HMHH parameter k needs to be explicitly set by practitioners, and Grobler *et al.* [75][76][74] typically use values of $k \in \{5, 10\}$ iterations. Nepomuceno and Engelbrecht

[133] propose a number of behavior changing schedules for PSO algorithms that stipulate when each particle is allocated different position or velocity update equations. Inspired by their approach, this thesis extends HMHH by supporting different types of heuristic change triggering functionality, namely the original *periodic trigger* that changes heuristic allocations every k iterations for all entities, a *stagnation trigger* that changes an entity's assigned heuristic if the performance of that individual entity has not improved over the last k iterations, and a *random trigger* that randomly triggers a heuristic change for any individual based on a probability parameter. Section 3.4.4 explores each type of trigger in more detail.

3.4.2 Neighborhood Topologies for Population-based Hyper-heuristics

HMHH assigns each entity in the parent population to a single heuristic to be modified. Heuristics may be population-based meta-heuristics that require a population of entities and (potentially) entity state information to operate as designed. Examples include a DE that needs other individuals to calculate a difference vector, or a PSO that requires social information from other particles to update an entity's velocity and position information. HMHH may assign any combination of entities to be operated upon by any combination of heuristics. The question arises whether all entities in the HMHH parent population should be accessible to all heuristics all the time (in a read-only fashion), or if each heuristic should only be allowed to access the information of those entities that are assigned to the heuristic.

The *neighborhood topology* of entities in the population of the HMHH framework defines the purview that each entity has of other entities in the HMHH parent population. Two possible *neighborhood topologies* arise naturally, namely a *global* and an *island* topology. The two topologies have noticeably different modes of operation:

- In the *global* topology, the population size of every heuristic appears to be the entire population of entities in the HMHH parent population. A heuristic can access the entity candidate solution and state information of the entire population of entities. However, a heuristic only updates the entities that HMHH assigns to the heuristic. Every heuristic is fully informed.

Allocation of different heuristics to any of the entities in the population does not alter the population size of any heuristic. The population of each heuristic appears to be constant even though each heuristic only modifies certain entities. This is an important property for some meta-heuristics such as DE and PSO where population size is an important factor to ensure good exploratory and exploiting capabilities.

- In contrast, the *island* topology effectively runs every heuristic in isolation. Entities assigned to each heuristic form individual sub-populations that each heuristic operates on independently. Information is only visible between members in the same sub-population, and information is only shared between heuristics when entities are reassigned to other heuristics. Any new entity-to-heuristic assignments may appear identical to moving entities between the sub-populations of the individual heuristics.

  Heuristic population sizes constantly fluctuate, which may impact the operation of heuristics. Certain heuristics such as DE require a minimum population size [60]. Such constraints need to be managed by setting the minimum allowed population size that HMHH must ensure is never violated when heuristic allocation is performed.

Distributed evolutionary algorithms (dEAs) bear some resemblance to the *island* neighborhood topology of the HMHH framework, particularly population-based dEAs based on the heterogeneous island model [71]. HMHH is different in that entities do not migrate between heuristic populations ("*islands*") in the dEA sense. Each heuristic in HMHH operates exclusively on its own disjoint sub-population of dedicated entities for k iterations. The collective performance of all entities assigned to each heuristic m over the previous k iterations is used to calculate the performance feedback Q_{δ_m} for that heuristic h_m. Every k iterations, all heuristic populations are "evacuated" and new populations are chosen from a common parent entity population E. Depending on the selection operator used by HMHH, the new sub-population associated with each heuristic may be drastically different compared to a steady island-style model with migration between islands.

3.4.3 Heuristic Space Diversity

Good HMHH selection operators prevent a downward spiral called *heuristic space convergence* where one heuristic "takes over" by holding on to all assigned entities forever. If this happens the optimization process devolves to using only that one dominating heuristic. Grobler and Engelbrecht [76] show that *heuristic space diversity* (HSD) plays an important role in hyper-heuristic performance in static environments. The HSD metric, $\mathcal{H}(t)$, as proposed by Grobler and Engelbrecht measures the spread of the population of entities across heuristics in the heuristic space as

$$\mathcal{H}(t) = \alpha \left(1 - \frac{\sum_{m=1}^{n_h} |T - n_m(t)|}{1.5 n_s}\right) \tag{3.1}$$

where n_s is the total number of entities, n_h is the number of managed heuristics, n_m is the number of entities assigned to heuristic m, α is a scaling factor (here $\alpha = 100$), and $T = n_s/n_h$. Values of $\mathcal{H}(t) \approx 1$ indicate that entities are balanced equally across all heuristics while $\mathcal{H}(t) \approx 0$ show that a few heuristics are controlling almost all of the entities. $\mathcal{H}(t)$ does not give any indication as to which heuristic has the greatest number of entities assigned, only that an imbalance is present.

Note that equation (3.1) has a drawback in that $\mathcal{H}(t)$ can become less than zero if there are more than four heuristics and all n_s entities are assigned to a single heuristic, since equation (3.1) would then simplify to $\alpha(1 - 2(n_h - 1)/1.5n_h)$. To avoid this situation any hyper-heuristic must maintain $n_m > 1$ at all times to ensure that $\mathcal{H}(t) > 0$. Another option is to use a different method to calculate the disparity between the entity assignments of different heuristics. Budescu and Budescu [23] discuss a measure called *normalized entropy* which is based on a measure of diversity of populations of people by Teachman [170] which uses Shannon's entropy theory [160]. Normalized entropy is extended below as a measure called $\mathcal{N}(t)$ that can be used as an alternative to $\mathcal{H}(t)$, where $\mathcal{N}(t)$ is defined as

$$\mathcal{N}(t) = -\sum_{m=1}^{n_h} \frac{p(t)\log_2(p(t))}{log_2(n_h)} \tag{3.2}$$

where

$$p(t) = \begin{cases} \dfrac{n_m(t)}{n_s} & \text{if } n_m(t) > 0 \\ \epsilon & \text{otherwise} \end{cases}$$

where ϵ is a very small positive constant. Similar to $\mathcal{H}(t)$, values of $\mathcal{N}(t) \approx 1$ when heuristic space diversity is high and $\mathcal{N}(t) \approx 0$ if one heuristic dominates. The $\mathcal{N}(t)$ measure improves upon $\mathcal{H}(t)$ since $\mathcal{N}(t)$ can never be negative, and $\mathcal{N}(t)$ is resilient in cases where a heuristic has zero entities assigned (i.e. $n_m(t) = 0$).

3.4.4 Triggering Heuristic Allocation Changes

A *heuristic change trigger* comprises of the logic to determine *when* a new heuristic change should be considered, and at which point heuristic selection logic selects *which* heuristic to apply. Early single-point search hyper-heuristic approaches embedded the heuristic change triggering decision logic implicitly in the heuristic selection logic itself. Examples include the *random gradient* hyper-heuristic and *choice function* hyper-heuristic by Cowling *et al.* [38]. The *random gradient* hyper-heuristic repeatedly applies the same heuristic until no improvement occurs, at which point a new heuristic is chosen randomly. The *choice function* hyper-heuristic uses a more elaborate selection mechanism that is based on the heuristic's performance, performance as a successor to other heuristics, and the number of iterations since the heuristic was previously applied. Both methods trigger the heuristic selection process to be performed every algorithm iteration.

Multi-point search hyper-heuristics are more complex in that the performance of each heuristic depends on the joint interactions between entities in a population. Every entity requires individual consideration about when and how a new heuristic should be used. Arguably, the simplest approach is to trigger heuristic selection after every application of the heuristic to the entity (i.e. after every algorithm iteration), or perhaps after every k algorithm iterations. The original HMHH definition by Grobler *et al.* [75][76][74] reassigns all entities to new heuristics every k algorithm iterations where common choices are $k \in \{5, 10\}$. The heuristic selection logic of *which* heuristic to use is decoupled from the triggering logic of *when* a new heuristic should be assigned.

In the SI literature, Nepomuceno and Engelbrecht [133] present a number of behavior changing schedules for heterogeneous PSO that select the most appropriate behavior (i.e. position and velocity update equations) for each particle. Every iteration, zero or more particles are *triggered* by a behavior changing schedule and are subsequently assigned new particle behaviors. Possible scheduling strategies include:

- **Periodic** behavior selection, where a new particle behavior is assigned to a particle every k iterations.
- **Randomized** behavior selection, where a new particle behavior is chosen randomly from a particle behavior pool based on a probability. The probability loosely corresponds to k iterations. For example a probability of $p = 0.1$ implies that behavior changes occur, in expectation, every $\frac{1}{p} = 10$ iterations. The parameter k represents the approximate number of iterations before a behavior change is desired, which is used to set the probability, i.e. $p = \frac{1}{k}$.
- **Stagnation-based** behavior trigger, where a new particle behavior is selected if the fitness (or personal best fitness) of a particle does not improve over k iterations.

This thesis extends HMHH with more elaborate heuristic allocation triggering mechanisms. The behavior changing schedules approach of Nepomuceno and Engelbrecht [133] is adapted to allow HMHH to trigger heuristic selection for each individual entity in combination with any suitable heuristic selection operator.

3.4.5 Managing Heuristic State Information

Management of heuristic state information is an important function that HMHH must address. State information consists of any meta-data associated with a candidate solution such as, for example, PSO memory of personal best positions, particle velocities, and any other state needed by any heuristics. Entities contain both a candidate solution vector together with all of the associated state-based information required by heuristics.

Different types of entity state exists:

- **Candidate solution vectors**: The candidate solution vector is arguably the most important state carried by an entity, because it represents a solution to the optimization problem. Consequently, only the heuristic assigned to an entity is allowed to alter the entity's candidate solution vector. This is true in both the *island* as well as *global* topologies, although each topology differs in how heuristics may *view* candidate solution vector information across the entire population E.

- **Static heuristic algorithm state**: Some heuristics may assign permanent state values to entities that are used as part of the heuristic's operation. Examples include the charge coefficients of an atomic PSO algorithm or the standard deviation values of Brownian individuals in DE [60]. Many other heuristics may have similar fixed configuration values for entities that never change over the course of the optimization run. These types of state values are simply associated with each entity and are made available to any heuristic in a read-only fashion.

- **Dynamic heuristic algorithm state**: Certain heuristics may maintain dynamically changing state information such as, for example, a PSO that maintains a personal best position and velocity information. Such heuristics depend on the state associated with each entity to allow normal heuristic operation. Some dynamic state may be located centrally (i.e. apart from the entities), for example, the hall of fame of a GA [60]. Generally, each heuristic is responsible for updating the dynamic state of their assigned entities as well as any central state.

  A pressing question is how each heuristic's state variables should be updated in external entities that are *not* assigned to the heuristic. For example, consider a *global* topology where a PSO heuristic refers to the (outdated) personal best positions of external entities managed by other heuristics such as DE or a GA. The PSO's velocity and subsequent position updates will be negatively affected by referring to global best information that is out of date. Another example is in an *island* topology where the personal best and velocity state of an external entity that is reassigned to a PSO heuristic still refers to outdated values from the previous period when the entity was last modified by a PSO.

  The following strategies are possible options to manage state across the heuristic pool:

  - **Share state**: In many cases heuristics are able to use and update shared state variables that carry the same meaning across any heuristic. Shared state variables allow heuristics configured in either a *global* or *island* topology to always have access to the latest state information. Shared state variables require that the semantic meaning of the variable is identical across heuristics

and that all heuristics manage the state variable similarly. For example, the personal best positions found by entities can generally be shared between all PSOs under HMHH.

Certain heuristics may need (non-logic altering) implementation modifications to cater for state updates that (technically) are not part of normal operation of the heuristics[1]. For example, a DE or GA may be required to update the personal best position of their assigned entities. This kind of modification does not affect the operation of the DE or GA, but is critical to the operation of any PSO algorithms that access the entities later.

– **Re-initialize state**: Some state variables may not be shareable between heuristics (practically or semantically), because the state is too algorithm-specific or complex to manage. A good example is particle velocity in a CPSO that computes a computationally complex acceleration coefficient as explained in section 2.6.2 in chapter 2. Requiring the HMHH framework or another heuristic such as DE or GA to maintain such complex state as an extra step is not practical. The simplest strategy to cope with such complex state is to reinitialize the state variable of an entity when it is reassigned to a new heuristic (the candidate solution vector is, however, not re-initialized). An example would be setting the PSO velocity to zero when an entity is reassigned to a PSO.

Resetting state information is not ideal since it prevents any learned insights from being exploited by the heuristic. However, state reinitialization prevents any bias that may be introduced by other more deterministic methods and is simple to perform for most heuristics (the mechanism is usually provided as part of the heuristic' original algorithm listing).

[1]This mechanism is more of an implementation concern than a modification of any heuristic's logic. A logical architectural choice is to implement such updates as part of the HMHH implementation. Various object-oriented design patterns [67] could be utilized to enable such a system, including *template method*, *strategy*, and *listener*. Swan *et al.* [168] and Woodward *et al.* [185] show that the *composite* design pattern can also be used to implement selection hyper-heuristics.

3.4.6 Selection Operators for HMHH

The HMHH framework requires that the selection hyper-heuristic logic be supplied as a configurable component called a *selection operator*. Different kinds of selection operators change how heuristics are assigned to entities. Each selection operator may use different types of feedback, may employ a mix of deterministic or non-deterministic selection mechanisms, may possibly utilize internal memory to keep track of good entity-to-heuristic assignments, and may even apply learning mechanisms to predict good assignments. This section outlines a number of selection operators for HMHH that have been adapted from the hyper-heuristic, SI, and EC literature. The following definitions are used:

- The set E contains the parent population of all entities in HMHH. Every entity $e_j \in E$ is a candidate solution along with any associated heuristic-specific state.
- $n_s = |E|$ is the total number of entities in E.
- The fitness of any entity e_j at time t is denoted as $f_j(t)$.
- The set H contains one or more heuristics $h_m \in H$.
- $n_h = |H|$ is the total number of heuristics in H.
- The set $s_m(t) \subseteq E$ contains the subset of entities in E that have been uniquely assigned to heuristic h_m at time t. There are n_h of these disjoint subsets of E. The union of all the disjoint sets is equal to E, i.e. $\cup_{i=1}^{n_h} s_i(t) = E$.
- $Q_{\delta_m}(t)$ is a problem space measure that serves as feedback for the selection operator to determine the quality of solutions produced by heuristic h_m at time t. $Q_{\delta_m}(t)$ is calculated in different ways by each selection operator.
- $n_m(t) = |s_m(t)|$ is the number of entities assigned to heuristic h_m at time t. The minimum size of $n_m(t)$ at any time t is kept at one, so as to avoid the situation where $Q_{\delta_m}(t)$ cannot be calculated for heuristic h_m because s_m is empty.
- $E^*(t) \subseteq E$ is the subset of entities in E that are to be assigned new heuristics at time t. Membership in $E^*(t)$ is determined by a heuristic selection trigger mechanism as discussed is section 3.4.4. The size of $E^*(t)$ is in the range $\{0, n_s\}$

depending on the trigger type. Entities $e_j \in E^*(t)$ may have a diverse mix of existing heuristic assignments, i.e. each e_j may form part of any sub-set $s_m(t)$.

- The selection operator $\varsigma(e_j, Q_{\delta_m}(t))$ assigns entity $e_j \in E^*(t)$ to a heuristic $h_i \in H$ at time t. The implication is that $\varsigma(e_j, Q_{\delta_m}(t))$ is a set membership function that may make use of $Q_{\delta_m}(t)$ to assign entity e_j to set s_i.

- $\Theta(t) = (P_1, P_2, ..., P_{n_h})$ is a list of probabilities of assigning a given entity to each of the n_h heuristics at time t. Selection operators may explicitly calculate and use $\Theta(t)$ as part of the logic of the operator. If $\Theta(t)$ is not directly calculated, the probabilities are implied by the logic of the selection operator.

In the original HMHH algorithm the selection operator is triggered every k iterations. This thesis extends HMHH to support periodic, stagnation and random triggers as outlined in section 3.4.4. The type of trigger that is used determines which subset of entities $E^*(t) \subseteq E$ will be considered for reassigned by the selection operator. The selection operator, in turn, assigns all triggered entities to heuristic and could, depending on the exact selection logic that is employed, assign an entity to the same heuristic that operated on the entity previously. Periodic triggers place all n_s entities in E into the set $E^*(t)$ to simultaneously be assigned a new heuristic every kth iteration (and leaves heuristic assignments untouched for the $k-1$ iterations between selection periods). Stagnation and random triggers apply membership functions every iteration to determine which entities in E should form part of the subset $E^*(t)$ to be assigned new heuristics. The set $E^*(t)$ may be empty some iterations, indicating that no entities require new heuristic assignments during those iterations. The type of trigger and the value of the parameter k for every trigger become configuration parameters for HMHH.

Additionally, this thesis extends HMHH to allow the use of either a global or island neighborhood topology as discussed in section 3.4.2. In the *global* topology each heuristic h_m only modifies entities in $s_m(t)$, but h_m has access to all entities in E. This full visibility model results in h_m immediately observing the effects of all other heuristics in every algorithm iteration. In the *island* topology each heuristic h_m only modifies entities in $s_m(t)$, but h_m only has visibility of entities in $s_m(t)$. Changes made by other heuristics will not be usable by h_m until those other entities are assigned to $s_m(t)$.

Heuristic space convergence occurs when all entities are allocated to a single heuristic, $h_c \in H$. Other heuristics $h_m \in H$ where $m \neq c$ will subsequently have zero assigned entities. Heuristics with zero entities are unable to ever improve any entities or produce valid performance scores $Q_{\delta_m}(t)$. Certain selection operators may subsequently be unable to ever assign any entities to any of the heuristics $h_m \in H$ where $m \neq c$. To avoid this situation, each heuristic maintains a minimum entity count of at least one entity to ensure that each heuristic always produces a valid performance score, $Q_{\delta_m}(t)$. The minimum entity count may be higher in the *island* topology if certain heuristics require a minimum population size (such as DE [60]).

Heuristic selection operators for HMHH

The following heuristic selection operators for HMHH are examined in this thesis:

1. **Fixed** selection never changes the original entity allocations, i.e. s_m remains constant for each heuristic h_m. Probabilities of selection P_m in the list $\Theta(t)$ are initially set to $P_m = \frac{1}{n_h}$ for $t = 0$, which is used to allocate all entities randomly across heuristics, where-after all P_m in the list $\Theta(t)$ are ignored for other iterations.

2. **Simple random** selection assigns each entity $e_j \in E^*(t)$ to heuristic h_m with equal probability. The probabilities of selection of every heuristic remain constant, where each P_m in the list $\Theta(t)$ is set to $P_m = \frac{1}{n_h}$.

3. **Permutation** selection is inspired by *random permutation descent* selection of Cowling *et al.* [38]. A randomized bipartite graph is generated that maps every heuristic $h_m \in H$ back onto the set of heuristics H, creating n_h one-to-one mappings between heuristics h_i and h_j where $i, j \in \{1, \ldots, n_h\}$. All triggered entities $e_i \in E^*(t)$ that are currently assigned to heuristic h_i are reassigned to heuristic h_j, as directed by the bipartite graph. Reassigned entities that previously shared a heuristic h_i will share the same new heuristic h_j. Isomorphic mappings are allowed (i.e. where h_i is the same as h_j) which effectively results in no entity changes occurring for any entities assigned to heuristic h_i. Each triggered entity e_i currently assigned to heuristic h_i has a probability of $P_j = 1$ for the bipartite graph-mapped heuristic h_j, while $P_m = 0$ for $m \neq j$.

4. **Roulette wheel** selection assigns each entity $e_j \in E^*(t)$ to heuristic h_m with a probability based on the performance of all entities in s_m relative to the performance of all entities in all other heuristics. The aggregated performance of each s_j, where $j = \{1, \ldots, n_h\}$, can be computed in two ways:

   - The mean fitness of all entities in the subset s_m is
   $$\mu_m(t) = \frac{\sum_{j=1}^{n_m} f_j(t)}{n_m} \tag{3.3}$$
   - The maximum fitness of all entities in the subset s_m is used (assuming maximization), i.e.
   $$\Upsilon_m(t) = \max_{j=1,\ldots,n_m} \{f_j(t)\} \tag{3.4}$$

   The probability of selection of each heuristic h_m, namely P_m in the list $\Theta(t)$, is respectively set to either $P_m = \frac{\mu_m(t)}{\sum_{l=1}^{n_h} \mu_l(t)}$ or $P_m = \frac{\Upsilon_m(t)}{\sum_{l=1}^{n_h} \Upsilon_m(t)}$. A new heuristic is randomly selected for each triggered entity relative to the probabilities of selecting each heuristic.

   Roulette wheel selection using the $\mu_m(t)$ term rewards heuristics that yield good fitness values for the majority of entities, and punishes heuristics that explore inferior parts of the search space. Heuristics with smaller and more uniform populations where most entities have relatively good fitness have a high probability of being assigned more entities. Conversely, heuristics with large and diverse populations tend to have a lower probability of being assigned more entities. The $\Upsilon_m(t)$ term rewards heuristics with the best exploitation behavior and does not penalize exploratory behavior nor large populations.

5. **Heuristic tournament** selection is inspired by tournament selection in evolutionary algorithms [60]. Heuristic selection holds tournaments to select a new heuristic for each entity $e_j \in E^*(t)$. The contestants in each tournament is a subset of heuristics from the heuristic pool H. A tournament set of heuristics, $T_j \subset H$, is randomly selected for each entity $e_j \in E^*(t)$. The size of T_j affects the behavior of tournament selection: a size of one is the same as random selection while a size of n_h is the same as elitist selection. The performance score of each heuristic in the tournament set, $h_m \in T_j$, is computed using either equation (3.3) or (3.4). The

winner of the tournament for entity e_j is that heuristic, $h_w \in T_j$, that has the best $\mu_m(t)$ or $\Upsilon_m(t)$ score, respectively.

The probabilities of selection, P_m in the list $\Theta(t)$, varies per entity and are set to $P_m = 1$ for $m = w$ and $P_m = 0$ for $m \neq w$. Different tournament sizes in the range $\{2, \ldots, n_h\}$ are possible.

6. **Entity tournament** selection is inspired by tournament selection in EAs [60]. Tournaments are conducted between the entities themselves. For every entity, $e_j \in E^*(t)$, a tournament set of entities, $T_j \in E$, is randomly drawn from the overall population of entities. The entity $e_w \in T_j$ with the best fitness score $f_w(t)$ is considered the winner. The triggered entity e_j under consideration is assigned the same heuristic as e_w, and e_w remains unchanged. The probabilities of selection P_m in the list $\Theta(t)$ are set to $P_m = 1$ for $m = w$ and $P_m = 0$ for $m \neq w$ and $\Theta(t)$ may vary per entity. Different tournament sizes in the range $\{2, \ldots, n_s\}$ are possible.

   Entity tournament does not rely on aggregate fitness information of heuristics as heuristic tournament does. Entities directly mimic the heuristic assignments of other successful entities via direct peer-to-peer comparison of fitness scores.

7. **Ant-inspired rank-based** selection [132] is inspired by the fundamental version of the ant colony optimization meta-heuristic (ACO-MH) [49][50]. Each heuristic h_m is assigned a *pheromone concentration* $\rho_m(t)$ as a relevancy score that is used to calculate P_m in the list $\Theta(t)$ using

$$P_m(t) = \frac{\rho_m(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \tag{3.5}$$

   Roulette wheel selection assigns each entity $e_j \in E^*(t)$ to a heuristic relative to the probabilities P_m in the list $\Theta(t)$. Initially, each heuristic h_m has a pheromone concentration $\rho_m(1) = \frac{1}{n_h}$. Pheromone levels are updated based on whether the fitness of each entity $e_j \in s_m$ increased, decreased, or stagnated, i.e.

$$\rho_m(t) = \rho_m(t-1) + \sum_{j=1}^{n_m} \begin{cases} 1.0 & \text{if } f_j(t) \text{ improved} \\ 0.5 & \text{if } f_j(t) \text{ remained the same} \\ 0.0 & \text{if } f_j(t) \text{ worsened} \end{cases} \tag{3.6}$$

Pheromone concentrations are partially evaporated every k iterations to avoid the build-up of extremely large scores, i.e.

$$\rho_m(t) \leftarrow \frac{\sum_{l=1,l\neq m}^{n_h} \rho_l(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \times \rho_m(t) \tag{3.7}$$

8. **Normalized ant-inspired rank-based** selection, proposed in this thesis, extends ant-inspired rank-based by normalizing the rank scores relative to the number of entities assigned to each heuristic before updating the pheromone, i.e.

$$\rho_m(t) = \rho_m(t-1) + \frac{1}{n_m} \times \sum_{j=1}^{n_m} \begin{cases} 1.0 & \text{if } f_j(t) \text{ improved} \\ 0.5 & \text{if } f_j(t) \text{ remained the same} \\ 0.0 & \text{if } f_j(t) \text{ worsened} \end{cases} \tag{3.8}$$

The rest of the selection operator is identical to ant-inspired rank-based selection. Normalization aims to prevent heuristics with large populations from overpowering heuristics with fewer entities, especially in situations where heuristics with smaller populations have a higher concentration of well-performing entities compared to heuristics with larger relatively poor-performing populations.

9. **Ant-inspired fitness proportional** selection [132] is identical to ant-inspired rank-based selection, but $\rho_m(t)$ is updated using the fitness improvement of entities $e_j \in s_m$. For function maximization $\rho_m(t)$ is updated as

$$\rho_m(t) = \rho_m(t-1) + \sum_{j=1}^{n_m} \max\{0, f_j(t) - f_j(t-1)\} \tag{3.9}$$

Ant-inspired fitness proportional selection emphasizes the magnitude of the raw fitness value improvements, while ant-inspired rank-based selection assigns the same rank value to any fitness improvement regardless of magnitude.

10. **Normalized ant-inspired fitness proportional** selection, proposed in this thesis, extends ant-inspired fitness proportional selection by normalizing the improvement scores relative to the number of entities assigned to the heuristic before updating the pheromone, i.e.

$$\rho_m(t) = \rho_m(t-1) + \frac{1}{n_m} \times \sum_{j=1}^{n_m} \max\{0, f_j(t) - f_j(t-1)\} \tag{3.10}$$

The rest of the selection operator is identical to *ant-inspired fitness proportional* selection. Normalization helps to avoid heuristics with large populations from overpowering heuristics with fewer entities, especially in situations where the larger population has a disproportionate number of very poor entities that improve greatly (as is frequently the case with randomly reinitialized entities that start to converge on the optimum).

11. **Frequency improvement** selection is based on Nepomuceno and Engelbrecht's *frequency-based heterogeneous PSO* behavior selection scheme (FB-HPSO) [134]. FB-HPSO selects new particle behaviors based on the frequency with which each behavior improves the fitness of particles over the previous k iterations. FB-HPSO is adapted here to select the best heuristic for each entity $e_j \in E^*(t)$. A frequency score, $\chi_m(t)$, is calculated for each heuristic h_m based on the number of times each entity $e_j \in s_m$ improved its fitness since the current heuristic was assigned, i.e.

$$\chi_m(t) = \sum_{i=1}^{k} \sum_{j=1}^{n_m} \begin{cases} +1 & \text{if } f_j(t-i) \text{ improved} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

A maximum of k prior iterations are considered to prevent historical feedback values from overshadowing more recent feedback. A value of $k = 10$ is used as per the guidance of Nepomuceno and Engelbrecht [134]. A heuristic's frequency score, $\chi_m(t)$, is calculated using only those historical fitness that were produced while the entity was assigned to the heuristic. Entities $e_j \in E^*(t)$ are assigned a heuristic via heuristic tournament selection using $\chi_m(t)$ to determine the winning heuristic.

12. **Frequency improvement reinforcement learning** selection applies a reinforcement learning approach similar to Narayek [130] and Burke *et al.* [28]. A rank score r_m is maintained for each heuristic. Heuristics are rewarded or punished based on the frequency with which entity fitness values improve. Initially, each heuristic h_m has a rank $r_m(1) = 0$. The change in rank, $\Delta r_m(t)$, is defined as the net count of how many times entities $e_j \in s_m$ improved compared to remaining the same or stagnating, i.e.

$$\Delta r_m(t) = \sum_{j=1}^{n_m} \begin{cases} +1 & \text{if } f_j(t) \text{ improved} \\ -1 & \text{if } f_j(t) \text{ remained the same or worsened} \end{cases} \tag{3.12}$$

Ranks are updated as $r_m(t) = r_m(t-1) + \Delta r_m(t)$. The maximum rank is $r_{max} = n_s$ and the minimum rank is $r_{min} = -n_s$. Every entity $e_j \in E^*(t)$ is assigned to the highest ranked heuristic (essentially all other heuristics are on the tabu list as per Burke's approach [28]). The probability values of P_m in the list $\Theta(t)$ are set to $P_m = 1$ for $m = r$ for the highest ranked heuristic, $h_r \in H$, while $P_m = 0$ for $m \neq r$. Rank ties are broken randomly. Equation (3.12) is subtly different than equation (3.11) in that the same or worse performance is punished with a negative score to avoid monotonically increasing rank values.

13. **Fitness proportional reinforcement learning** selection is based on a similar mechanism as frequency improvement reinforcement learning, but uses the mean *change* in fitness of entities $e_j \in s_m$, namely $\phi_m(t)$, to reinforce ranks. For function maximization, $\phi_m(t)$ is defined as

$$\phi_m(t) = \frac{\sum_{j=1}^{n_m}(f_j(t) - f_j(t-1))}{n_m} \tag{3.13}$$

and the change in rank, $\Delta r_m(t)$, for each heuristic h_m is

$$\Delta r_m(t) = \begin{cases} +1 & \text{if } \phi_m(t) > 0 \\ -1 & \text{if } \phi_m(t) \leq 0 \end{cases} \tag{3.14}$$

yielding a maximum change in rank per iteration for any heuristic of $\Delta r_m(t) = \pm 1$. Each heuristic's rank is updated as $r_m(t) = r_m(t-1) + \Delta r_m(t)$. The maximum rank is $r_{max} = n_h$ while the minimum rank is $r_{max} = -n_h$. The rest of fitness proportional reinforcement learning is identical to frequency improvement reinforcement learning.

Fitness proportional reinforcement learning ranks heuristics relative to the magnitude of entity fitness changes (regardless of the number of improving moves), while frequency improvement reinforcement learning ranks heuristics relative to the frequency with which entity fitness values improve (regardless of fitness change magnitudes).

14. **Difference proportional** selection [165] by Spanevello and Montes de Oca probabilistically assigns each entity $e_j \in E^*(t)$ to that heuristic h_b that contains the

fittest entity $e_b \in E$, where the probability of reassignment is relative to the performance of e_j and e_b. The probability $P_b(t)$ of reassigning entity e_j from the entity's currently assigned heuristic h_j to the heuristic of the best entity h_b at time t is

$$P_b(t) = \frac{1}{1 + \exp\left(-\beta \frac{f_b(t) - f_j(t)}{|f_b(t)|}\right)} \tag{3.15}$$

where $\beta = 5$ as recommended by Spanevello and Montes de Oca. The function $P_b(t)$ has a sigmoidal shape that enables difference proportional selection to increase the probability of assigning poor performing entities to h_b, and lower the probability of reassigning well-performing entities to different heuristics. The probabilities of selection P_m in the list $\Theta(t)$ are $P_m = P_b(t)$ for $m = b$, $1 - P_b(t)$ for $m = j$, and $P_m = 0$ otherwise. $\Theta(t)$ is recomputed for every entity.

15. **Competitive population** selection is inspired by the *Competitive Population Evaluation* (CPE) algorithm for DE by Du Plessis and Engelbrecht [51]. CPE is adapted here to alter the probability of selecting each heuristic h_m based on two factors: the performance of the best entity assigned to h_m, namely $e_{b,m} \in s_m(t)$, and the magnitude of the fitness improvement of $e_{b,m}$. The performance $\mathcal{P}_m(t)$ of h_m at time t is defined as

$$\mathcal{P}_m(t) = (\Delta f_{b,m}(t) + 1) \times (R_M(t) + 1) \tag{3.16}$$

where

$$\Delta f_{b,m}(t) = |f_{b,m}(t) - f_{b,m}(t-1)| \tag{3.17}$$

and

$$R_m(t) = |f_{b,m}(t) - \min_{q=1,..,n_h} \{f_{b,q}(t)\}| \tag{3.18}$$

Higher values for $\mathcal{P}_m(t)$ indicate that h_m is better suited to receive more function evaluations at time t than other heuristics with lower values for $\mathcal{P}(t)$. The probabilities of selection P_m in the list $\Theta(t)$ is maintained as follows

$$P_m(t) = \frac{\mathcal{P}_m(t)}{\sum_{q=1}^{n_h} \mathcal{P}_q(t)} \tag{3.19}$$

where-after Roulette wheel selection is employed to assign all entities, $e_j \in E^*(t)$, new heuristics using $P_m(t)$.

16. **Soft-max** selection is based on the soft-max choice rule attributed to Luce [114]. A Gibbs-Boltzmann distribution is used to assign a new heuristic to every entity $e_j \in E^*(t)$. Each heuristic $h_m \in H$ is assigned a performance score of the best fitness managed by h_m, namely $\Upsilon_m(t)$ as calculated using equation (3.4). All performance scores across all heuristics are normalized to sum to one to yield $\Upsilon_m^{\text{norm}}(t)$, where $\sum_{i=1}^{n_h} \Upsilon_i^{\text{norm}}(t) = 1$. The soft-max choice function is applied to the normalized scores to yield the probability $P_m(t)$ of reassigning entity e_j to h_m as follows:

$$P_m(t) = \frac{e^{\Upsilon_m^{\text{norm}}(t)/\tau}}{\sum_{i=1}^{m} e^{\Upsilon_m^{\text{norm}}(t)/\tau}} \tag{3.20}$$

where the temperature parameter is $\tau = 1$, because all scores are normalized already. Roulette wheel selection assigns every entity $e_j \in E^*(t)$ to a heuristic using $P_m(t)$.

3.4.7 Complementary Heuristics

The facets that impact what are deemed good candidate heuristics to include in the pool of heuristics are critical to the success of hyper-heuristic approaches. Informally, heuristics are *complementary* to each other if each method "covers the weak spots of the other heuristics". Grobler [73] discusses how a complementary set of heuristics under HMHH (for static environments) should compensate for each other's strengths and weaknesses.

Peng *et al.* [147] propose a metric (for static environments) to compare the risk associated with any two heuristics on a set of problems. Considering a set of problems, $F = \{f_k | k = 1, 2, \ldots, n\}$, and a set of heuristics, $H = \{h_i | i = 1, 2, \ldots, m\}$, the risk associated with h_i on F indicates the likelihood of h_i *failing* on one or more problems $f_k \in F$. Assuming equal prior probability $P(f_k)$ for all $f_k \in F$, the probability, $P(h_i \text{ fails})$, of h_i failing to solve a problem in F is

$$P(h_i \text{ fails on } F) = \frac{1}{n} \sum_{k=1}^{n} P(h_i \text{ fails to solve } f_k | f_k), f_k \in F \tag{3.21}$$

In practice, however, it is non-trivial to define what the "*failure*" of a heuristic looks like. Peng *et al.* propose to compare the risk of heuristics using their solution quality on

previous representative problem instances. Let $q_{i,k}$ denote the quality of heuristic h_i on problem $f_k \in F$. Given two heuristics h_i and h_j, then h_i is less risky than h_j if and only if the conditional probability of h_i outperforming h_j on F, namely $P(q_i > q_j|F)$, is greater than the conditional probability of h_j outperforming h_i on F, namely $P(q_i < q_j|F)$, which gives

$$P(q_i > q_j|F) = \frac{1}{n}\sum_{k=1}^{n} P(q_{i,k} > q_{j,k}|f_k), \forall f_k \in F \tag{3.22}$$

and

$$P(q_i < q_j|F) = \frac{1}{n}\sum_{k=1}^{n} P(q_{i,k} < q_{j,k}|f_k), \forall f_k \in F \tag{3.23}$$

with

$$P(q_i > q_j|F) + P(q_i < q_j|F) + P(q_i = q_j|F) = 1.0 \tag{3.24}$$

$P(q_{i,k} > q_{j,k}|f_k)$ for any two heuristics $h_i, h_j \in H$ can be estimated by running both h_i and h_j on some $f_k \in F$ for s_i and s_j times respectively. This gives $s_i \times s_j$ pairs of solutions, allowing the probability that h_i outperforms h_j on f_k to be estimated by counting the number of times a solution of h_i beats a solution of h_j and dividing by $s_i \times s_j$. This procedure can be repeated for multiple $f_k \in F$ to yield $P(q_i > q_j|F)$. Peng *et al.* note that the estimation of $P(q_{i,k} > q_{j,k})$ is closely related to the statistical test U in the Wilcoxon rank-sum test [117], which means equations (3.22) and (3.23) "normalize" the average performance of h_i and h_j on F without bias to any specific f_k. This allows fair comparisons between algorithms on a class of problems.

Given this reasoning about algorithm risk, Peng *et al.* [147] elaborate on what a *complementary* set of heuristics means (again, for static environments). How can two constituent heuristics h_1 and h_2 be chosen for a heuristic pool such that a hyper-heuristic using both h_1 and h_2 together yields better performance than any other heuristic $h_3 \in H$? The optimal pool of heuristics can be found by minimizing the probability P_{worse} that the hyper-heuristic obtains worse results than h_3. If $P_{1,k}$ and $P_{2,k}$ are the probabilities of h_1 and h_2 respectively performing better than h_3 on problem f_k, then this minimization

becomes

$$\begin{aligned}\min\{P_{\text{worse}}\} &= \frac{1}{n}\sum_{k=1}^{n}(1-P_{1,k})(1-P_{2,k}) \\ &= (1-\bar{P}_1)(1-\bar{P}_2) + \frac{1}{n}\sum_{k=1}^{n}(P_{1,k}-\bar{P}_1)(P_{2,k}-\bar{P}_2)\end{aligned} \tag{3.25}$$

where $\bar{P}_1 = \frac{1}{n}\sum_{k=1}^{n} P_{1,k}$ and $\bar{P}_1 = \frac{1}{n}\sum_{k=1}^{n} P_{2,k}$. Equation (3.25) shows that larger $P_{1,k}$ and $P_{2,k}$ imply that h_1 and h_2 should perform well on the problem set F. Similarly, the ideal situation would be if $(P_{1,k} - \bar{P}_1) > 0$ when $(P_{2,k} - \bar{P}_2) < 0$ and vice versa. In other words, for any set of problems F, two heuristics h_1 and h_2 are *complementary* on a given problem $f_k \in F$ if the performance of h_1 on f_k is above the "typical" average of h_1 on F when the performance of h_2 on f_k is below the "typical" average of h_2 on F.

Peng *et al.* [147] show how the above analysis can be extrapolated to finding m complementary heuristics to include in a hyper-heuristic's heuristic pool. For any m candidate heuristics, a pair of heuristics h_1 and h_2 are established as a basis. Any subsequent candidate heuristic h_c should be complementary to the heuristics already selected as well as the resulting hyper-heuristic as a whole. The performance of the hyper-heuristic improves so long as additional *complementary* heuristics can be identified and added to the heuristic pool. Tang *et al.* [169] provide an approach (for static environments) that extends equation (3.25) to automatically select the optimal pool of complementary heuristics from a candidate set of m heuristics. The approach employs as *estimated performance matrix* (EPM) to construct the optimal pool of heuristics for a given set of problems F and not a single instance of a problem.

All of the above studies focused on static environments. DOPs are more nuanced in that the characteristics of the problem may change over time. Consequently, the complementary nature of the pool of heuristics may also change over time. Further research should be performed to formally define what a set of "complementary" heuristics looks like for hyper-heuristics that solve DOPs, and ways to automatically (or at least through a rigorous manual process) find the right number and set of complementary heuristics to include in the heuristic pool of a hyper-heuristic.

3.5 Related Approaches

A core function of selection hyper-heuristics is to continuously adapt the optimization algorithm while it is running to yield increased performance over using any of the heuristics in isolation. The goal of dynamically modifying the algorithm at run-time places selection hyper-heuristics in the category of *parameter control methods*. In the recent survey by Karafotias *et al.* [97], a number of parameter control adaptation approaches are reviewed. Burke *et al.* [24] also review a variety of parameter control adaptation methods that have similar aims to selection hyper-heuristics.

Burke *et al.* [24] consider hyper-heuristics to be different than other control adaptation mechanisms. Hyper-heuristics unify the most promising ideas in the CI field (such as machine learning and meta-heuristic-based optimization methods) with the wealth of human-created insights that have been gathered across various problem domains over decades. Hyper-heuristics have the goal of solving complex real-world problems in a more general fashion, to produce reusable technologies easier and, importantly, to incorporate as much human insight as possible without resorting to "reinventing the wheel with machine learning" in cases where prior domain knowledge exists.

The selection hyper-heuristics employed in this thesis manage a pool of heterogeneous, independent, and self-contained heuristics that are treated as separate "black-box" optimization methods. A heuristic may be as simple as a Gaussian mutation operator, or be a complex meta-heuristic implementation that employs state-of-the-art optimization techniques that contain bespoke algorithm parameters, design decisions, and self-adaptation mechanisms. Each heuristic comprises of various components that have been proven to work well together, either from empirical performance studies and/or human curation. The complexity inherent to each heuristic is fully encapsulated inside the heuristic, i.e. a hyper-heuristic has no knowledge of the low-level details of any heuristic. Instead, a hyper-heuristic relies solely on performance feedback from heuristics operating on the problem domain to decide which heuristics to apply next. The hyper-heuristic can focus on giving the most promising heuristics every opportunity to succeed, while reducing the computational effort wasted on inferior heuristics. In a DOP, the combination of which heuristics are deemed "most promising" is dependent on time t.

The paragraphs below briefly outline some of the main approaches in the SI and EA

fields that have similar goals to selection hyper-heuristics.

Parameter control (PC) for EAs [57][113] optimize various EA parameters at runtime. Karafotias *et al.* [97] classify PC mechanisms into three categories:

1. *Parameter specific* methods that adapt specific EA parameters such as population size, variation, selection, fitness function modification, or parallel EA parameters.

2. *Multiple parameter ensembles* that combine multiple heterogeneous control mechanisms into either a) variation and population, or b) variation and selection combinations.

3. *Parameter independent* methods applicable to any (numeric) EA parameter.

Adaptive operator selection (AOS) [118], a specific type of PC, continuously selects the most appropriate EA variation operator to use at time t. A *credit assignment* mechanism rewards different EA variation operators based on observed quality. Sources of feedback can include individuals' fitness or fitness improvement, diversity measures, or EA-specific measures such as offspring survival and population tenure of individuals [97].

Adaptive memetic algorithms (MA) [101] is a hybrid EA approach that self-adaptively combines population-based global search methods together with individual local learning approaches. Ong *et al.* [140] provide a classification of adaptation mechanisms for MAs.

Self-learning PSO [107][179] and *heterogeneous PSO* [132][134] are examples of techniques that adapt the search behaviors of individual particles in a PSO scheme based on performance feedback received during the search.

Algorithm portfolios, first proposed by Huberman [86], run different algorithms concurrently. A time-sharing mechanism determines how much computation time each algorithm should receive. Peng *et al.* [147] observe that the performance of various heuristics may vary greatly between different problems, implying a *risk* in selecting any particular algorithm. Peng *et al.* developed a population based algorithm portfolio (PAP) that offsets the *risk* of spending too much computational *budget* on inferior methods by *diversifying* the optimization process across multiple SI and EC approaches. Constituent algorithms in PAP interact on a regular basis by migrating entities between sub-populations. For each constituent algorithm A_i, the sub-populations of all other remaining constituent algorithms are combined and the best e entities are copied into

the population of A_i. The worst e entities of the population of A_i are subsequently discarded.

Ensemble methods diversify the optimization process by utilizing multiple search strategies and/or parameters. Wu *et al.* [187] present a high-level *ensemble of DE variants* (EDEV) that divides a population of individuals into smaller *indicator* sub-populations and a *larger* reward sub-population. Each DE variant manages an indicator sub-population. The DE variants compete against each other using performance on their respective indicator sub-populations. After a set number of generations, the best performing DE variant is assigned the reward population, thereby allowing the best DE variant to command the most computational resources until the next selection phase. EDEV is inspired by *multi-population ensemble DE* (MPEDE) [186] that uses a similar approach on low-level DE mutation strategies.

3.6 Summary

Hyper-heuristics as a field offers a promising control adaptation approach that combines prior human domain understanding together with machine learning techniques to yield more generally applicable optimization methods. Hyper-heuristics are classified broadly into *selection* hyper-heuristics that continually select the next best heuristic to apply to a problem, and *generative* hyper-heuristics that evolve reusable heuristics that are tailored for either a single problem instance or a class of problems.

Heuristic performance feedback may be used by hyper-heuristic approaches to learn the most appropriate combination and/or sequence of heuristics to apply at time t. Learning can occur in three major ways: *off-line* if a sufficient number of example problem instances and associated feedback are available; *on-line* while the hyper-heuristic is solving the problem through (self-)adaptation based on feedback received; or no learning is performed by the hyper-heuristic which instead relies on fixed internal selection logic to assign heuristics based on the received feedback.

A hyper-heuristic never operates directly on the candidate solutions in the problem space, but instead searches for the most suitable method to apply to the problem next. Candidate solutions can be modified by either *constructive* heuristics that build up a final solution from an empty state, or *perturbative* heuristics that mutate existing candidate

solutions in the hopes of improving those solutions.

The remainder of this thesis focuses on the HMHH by Grobler *et al.*, as outlined in section 3.4.1, as an example of a multi-point search selection hyper-heuristic with on-line learning and perturbative heuristics. HMHH is extended with the notions of *neighborhood topologies* as well as *heuristic selection triggers*. The island topology restricts each heuristic to operate completely independently, only allowing information to be shared between entities if they are operated on by the same heuristic and form part of the same sub-population. The global topology allows information of all entities to be visible to all heuristics at all times, even though each heuristic still only modifies the entities that have been assigned to the heuristic. The original HMHH algorithm supports a *periodic trigger* which performs heuristic selection for all entities every k iterations. This thesis extends HMHH with a *stagnation trigger* that changes an individual entity's heuristic if the entity's performance does not improve over k iterations. A *random trigger* is also introduced that probabilistically changes an entity's heuristic approximately every k iterations.

The next chapter defines various control methods and establishes the baseline performance of each control group to ground the analysis of all investigated hyper-heuristics against.

Chapter 4

Estimation of Performance Baselines for Control Groups

"A goal properly set is halfway reached."
– Abraham Lincoln

This chapter establishes the performance of various control methods, which grounds the analysis of all investigated hyper-heuristics against an objective set of baselines. The control methods alleviate concerns around whether or not any increased performance by hyper-heuristics is due to intelligent selection, or simply due to the use of multiple sub-populations, multiple methods, or random heuristic assignments. The approach described below serves as the foundation for all experimental work in the rest of the thesis.

4.1 Introduction

To objectively measure whether any hyper-heuristic improves performance significantly, the baseline performance and behavior profiles of various control methods need to be established first. Section 4.2 proposes a new performance measure for algorithms aimed at solving DOPs. The proposed measure does not assume normally distributed performance data across an algorithm run, is resilient against fitness landscape scale changes, better incorporates performance variance across multiple fitness landscape changes, and

allows easier algorithm comparisons using established nonparametric statistical methods.

Section 4.3 outlines the experimental procedure used to establish all performance baselines in detail, including the definition of control groups, how the DOP algorithm building blocks that were outlined in section 2.5 are incorporated into the approach, the exact parameter values and settings of the heuristic algorithm implementations used in experiments, the moving peaks benchmark implementation details, and the considerations around assessing algorithm performance.

Section 4.4 presents the results of experimentation as a series of three distinct research questions that, respectively, characterize the newly proposed error measure, and establishes the performance baselines of the individual heuristics and speciated versions of the heuristics using the new measure. Section 4.5 concludes the chapter.

4.2 Measuring and Comparing DOP Performance

Section 2.7.2 discussed the most widely used performance measures for DOPs, along with their relative strengths and weaknesses, as reviewed in literature [40][119][125][136][151]. A summary of the shortcomings of the most commonly reported measures in literature is that, generally, most existing measures

- rely on non-normalized fitness or error values that make it hard to compare methods across landscape changes, problem instances, or problem types due to possible fitness scale changes,
- depend on simple aggregations and parametric statistical methods (such as the mean or standard deviation) on measurement data that does not necessarily follow a Gaussian distribution,
- do not consider the variance of measured performance values over time for an algorithm run,
- do not allow for simple statistical significance testing using established methods such as the nonparametric tests outlined in section 2.7.4, and
- penalize algorithms that show more exploratory behavior.

This thesis proposes a new performance measure called *relative error distance*, or P_r, that helps to address the short-comings of existing performance measures. Relative error distance shows how close to perfect an algorithm performs across all n_c performance values being measured, where n_c is the total number of environment change events that occur over time in an algorithm run. P_r is defined as

$$P_r(\mathbf{b}) = \frac{\sqrt{\sum_{i=1}^{n_c}(1-b_i)^2}}{|\mathbf{d}|} \tag{4.1}$$

$$= \frac{\sqrt{\sum_{i=1}^{n_c}(1-b_i)^2}}{\sqrt{n_c}}$$

where the vector $\mathbf{b} = (b_1, b_2, \ldots, b_{n_c})$ represents all measured performance values of a single execution (or run) of an algorithm. The components of $\mathbf{b}$, namely b_i, consist of the n_c different *relative error* values, $P_{RE}(t)$ (as defined in equation (2.18)), obtained just before the end of each change period. The n_c-dimensional vector $\mathbf{d} = (1, \ldots, 1)$ represents the best possible performance for each search landscape, i.e., where the $P_{RE}(t) = 1,\ \forall t = 1, \ldots, n_c$. Since all n_c components of $\mathbf{d}$ are equal to one, the magnitude of $\mathbf{d}$ is $|\mathbf{d}| = \sqrt{n_c}$.

The P_r measure considers the distance between the vector $\mathbf{b}$ and the vector $\mathbf{d}$. The definition of P_r in equation (4.1) relies on the $L2$ norm (Euclidean distance). Beyer *et al.* [9] show that the concept of distance in high dimensional spaces may not even be meaningful. Specifically, they show that the ratio between the nearest and the farthest elements approaches one in higher dimensional spaces for most distance measures and reasonable data distributions. Aggarwal *et al.* [1] demonstrate that the meaningfulness of the L_k norm is sensitive to larger values of k, and conclude that the $L1$ norm (Manhattan distance) or even fractional L_k norms (where $k \sim [0, 1]$) may be preferable for very large dimensions. The experiments in this thesis use the $L2$ norm, since the number of dimensions of P_r is relatively low (i.e., only the $P_{RE}(t)$ value of the final iteration of each search landscape resulting from environment changes are considered). In situations where a large number of comparison points are used, it may be preferable to redefine the

relative error distance in terms of the $L1$ norm as follows:

$$P_{r,L1}(\mathbf{b}) = \frac{\sum_{i=1}^{n_c} (|1 - b_i| + |r_i|)}{n_c} \tag{4.2}$$

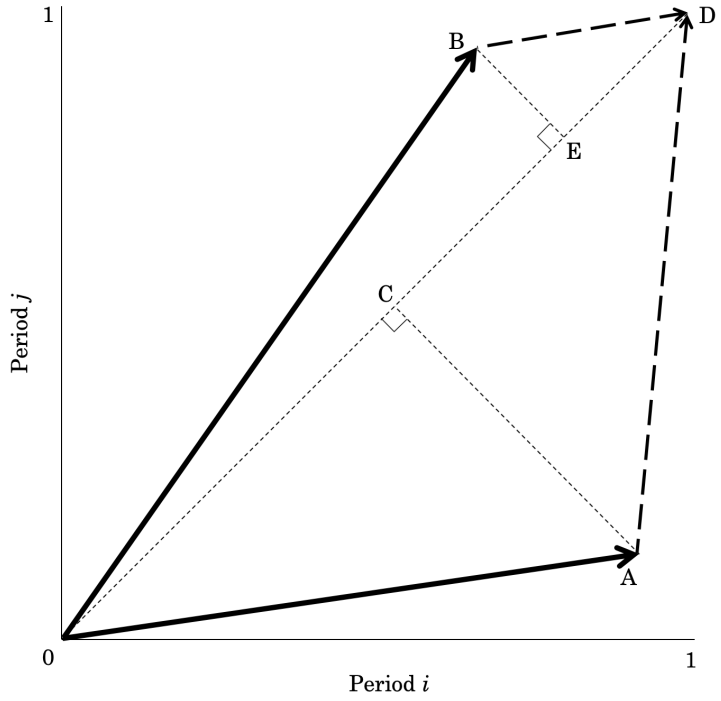
where the vector $\mathbf{r} = (r_1, \ldots, r_{n_c})$ is the rejection vector that results when $\mathbf{b}$ is projected onto $\mathbf{d}$. If $\mathbf{d}^*$ is the projection vector of $\mathbf{b}$ onto $\mathbf{d}$, then the rejection vector, $\mathbf{r}$, is the orthogonal vector that satisfies $\mathbf{r} = \mathbf{b}$ - $\mathbf{d}^*$ [148].

Figure 4.1 illustrates how P_r works in a two-dimensional space. Sub-figures 4.1a and 4.1b show the operation of the P_r measure using the L_2 and L_1 norm, respectively. Each axis represents the measured $P_{RE}(t)$ value for every landscape resulting from environment changes, as measured at the iteration just before an environment change occurs. The label D in the figure shows point $\mathbf{d} = (1, 1)$, which is the best possible point of performance of all $P_{RE}(t)$ scores across all periods of interest.

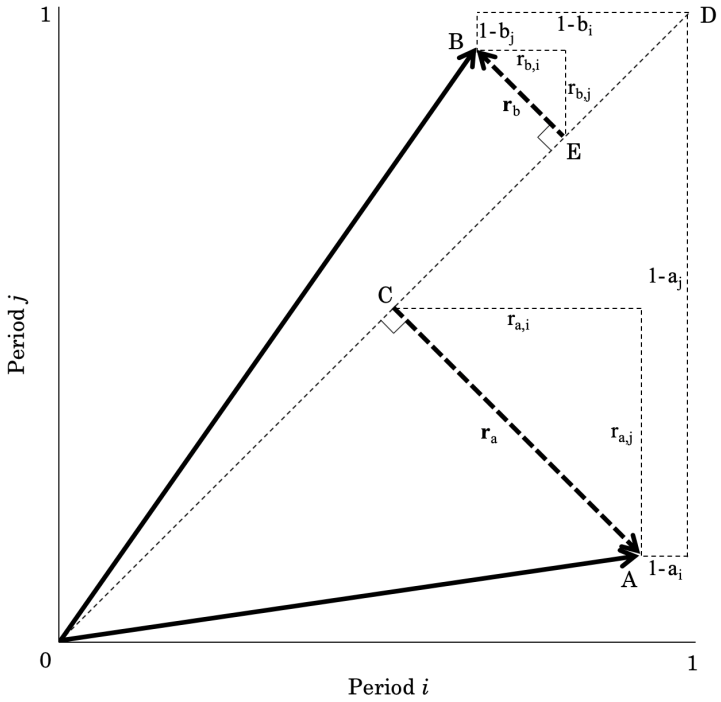
In sub-figure 4.1a, the points representing two algorithm runs are represented by labels A and B for points $\mathbf{a} = (a_i, a_j)$ and $\mathbf{b} = (b_i, b_j)$, respectively. An algorithm's performance may be inconsistent across different environment changes. One algorithm might perform well in certain search landscapes and badly in others, while another algorithm might show little variation in performance over time (regardless of whether performance is good or bad). Algorithm A clearly performed very well in period i, but performed poorly in period j. Algorithm B did relatively well in period i (although worse than A), and very well in period j. The dashed line segments AD and BD show the Euclidean distance to point $\mathbf{d}$ that P_r relies on. The shorter line segment BD shows that, across both periods i and j, algorithm B performed better than A. A geometric interpretation of the performance variance of algorithms is possible:

- High variance in $P_{RE}(t)$ values across dimensions results in the n_c dimensional point of the algorithm's performance lying further away from the diagonal line. As an example, point A in figure 4.1a shows good performance in dimension i and bad performance in dimension j. Consequently, the line segment AC is relatively long.

- Low variance in $P_{RE}(t)$ values across dimensions leads to the n_c dimensional point of the algorithm's performance lying near the diagonal line. For example, point B



(a) P_r defined in terms of the $L2$ norm.



(b) P_r defined in terms of the $L1$ norm.

Figure 4.1: Illustration of how P_r works on two hypothetical algorithm runs A and B over two landscape change periods i and j using, respectively, the $L1$ and $L2$ norms.

in figure 4.1a has relatively similar values for dimensions i and j and, consequently, the line BE is relatively short.

The P_r measure captures the variance in the performance of an algorithm across different change landscapes better than simple statistical aggregations such as the mean.

In sub-figure 4.1b, the $P_{r,L1}$ value for algorithm A at point $\mathbf{a} = (a_i, a_j)$ consists of the average of the absolute differences between the components of $\mathbf{a}$ and the components of $\mathbf{d}$, added to the average of the components of the rejection vector of $\mathbf{a}$, namely $\mathbf{r}_a = (r_{a,i}, r_{a,j})$. The $P_{r,L1}$ value for algorithm B at point $\mathbf{b} = (b_i, b_j)$ is calculated in a similar way using the rejection vector for $\mathbf{b}$, namely $\mathbf{r}_b = (r_{b,i}, r_{b,j})$. The rejection vector for algorithm A is much larger than the rejection vector for algorithm B, which results in A having a larger $P_{r,L1}$ value than B.

P_r has the following characteristics that make it a desirable measure for comparing DOP-focused algorithms:

1. **Normally distributed data values are not assumed**: The P_r measure does not assume that the measured performance data has a Gaussian distribution (or any other specific distribution), building on the findings of many recent studies that indicate how parametric statistical procedures are frequently the wrong choice to evaluate the performance of CI algorithms [47][48][68][69][70]. Instead, P_r relies on the n_c-dimensional distance of all measured points to a common, objective point of reference. Using a norm such as the Euclidean distance metric results in a scalar value that produces an aggregating effect, without the disadvantage of skewing results by assuming normally distributed data.

2. **Normalize errors with varying scales**: The $P_{RE}(t)$ measure transforms the raw error value at time t to fall in the range $[0, 1]$ relative to the minimum and maximum possible fitness values at time t. The value $P_{RE}(t) = 1$ always represents perfect performance and $P_{RE}(t) = 0$ indicates the worst possible performance, regardless of the value of the global optimum at time t. The P_r measure shares in the benefits of using $P_{RE}(t)$ (as discussed in section 2.7.2), namely that $P_{RE}(t)$ is resilient to fitness magnitude differences across different landscapes, $P_{RE}(t)$ allows for easier comparison between maximization and minimization problems, and that $P_{RE}(t)$ is unbiased towards algorithms that show more exploratory behavior. The

sequence of $P_{RE}(t)$ values resulting from each subsequent search landscape between environment changes can be compared against each other more objectively.

3. **Better representation of performance variance over time**: Each run of a CI algorithm on a DOP yields a sequence of interrelated search landscapes. An algorithm might do relatively well in certain landscapes, but perform poorly in a subset of landscapes in a run. Existing measures that rely on simple aggregations, such as the mean over all considered points, may lose important variance information across landscape changes. Considering variance using methods such as the standard deviation, results in a second scalar value to report. For P_r, the n_c distinct $P_{RE}(t)$ measurements are not simply averaged to yield an average performance score. Instead, the joint impact of all n_c of the $P_{RE}(t)$ values are considered individually by examining the n_c-dimensional distance between the vector of $P_{RE}(t)$ values and the point of best possible performance. Any volatility/variance in $P_{RE}(t)$ values is better captured by P_r as a single scalar value.

   For example, consider three algorithm runs with $P_{RE}(t)$ values of $a_1 = [0.8, 0.8]$, $a_2 = [0.9, 0.7]$, and $a_3 = [1, 0.6]$, where each run has two distinct $P_{RE}(t)$ values for each of two environment landscapes. All three runs have the same arithmetic mean value of 0.8, but the Euclidean distances to the point $(1, 1)$ are 0.2828, 0.3162, and 0.4, respectively. Consequently, the P_r scores, as calculated using equation (4.1), are 0.2, 0.2236, and 0.2828 for a_1, a_2, and a_3, respectively. The P_r measure better captures the variance of the $P_{RE}(t)$ samples over time, whereas the arithmetic mean loses all variance information. Figure 4.2 illustrates the outlined example.

4. **Sound comparisons using proven statistical procedures**: P_r allows multiple relative error values to be combined into a single value that allows for easy comparison and significance testing. The point $\mathbf{d}$ represents perfect performance across all n_c periods being considered. The P_r measure transforms all n_c dimensions of $\mathbf{b}$ into a single scalar number that represents the "distance that $\mathbf{b}$ is from perfect performance" as measured in absolute terms. Multiple P_r values can be compared directly, and the scalar nature of P_r allows for statistical significance testing using tried and true procedures such as the methods described in section 2.7.4.

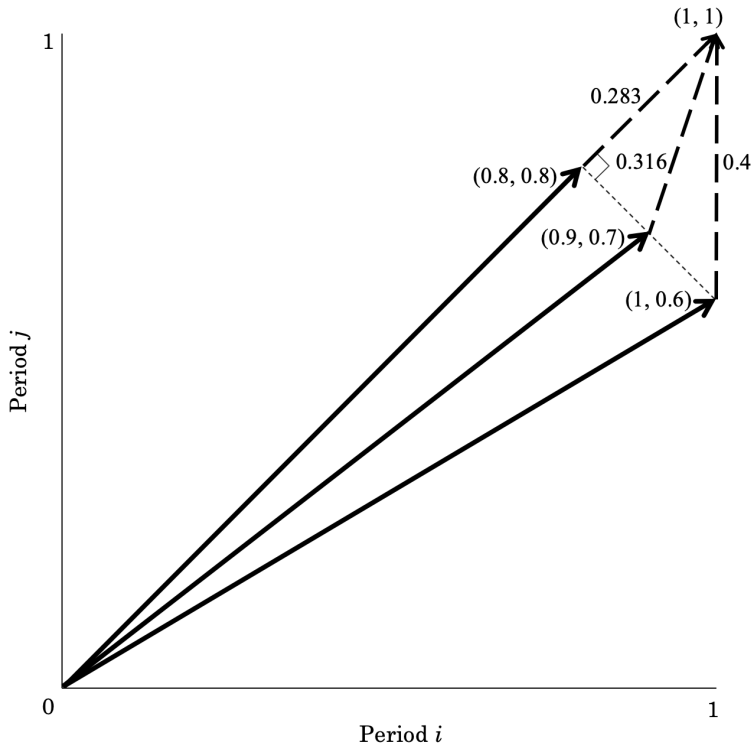


Figure 4.2: Illustration of how P_r works in two dimensions, i.e., over two landscape change periods i and j.

Considering the facets that are deemed to be desirable in DOP performance measures (as discussed in section 2.7.1), the P_r measure is intuitive, gives equal consideration to all n_c search landscapes, treats maximization and minimization problems uniformly, contains no subjective parameters, is protected against ill-defined results, does not assume normally distributed measurement data, is robust against fitness scale changes, incorporates variance of performance over time, does not penalize exploration, and reports performance as a scalar value that can more readily be compared to P_r scores of other algorithms outside of a study. Section 4.4.2 empirically validates whether the data assumptions for P_r do, in fact, hold for actual algorithm performance data.

The proposed measure does not assume normally distributed performance data across an algorithm run, is resilient against fitness landscape scale changes, better incorporates performance variance across multiple fitness landscape changes, and allows easier algorithm comparisons using established nonparametric statistical methods.

4.3 Experimental Procedure

The goal of this thesis is to investigate how and why certain HMHH selection operators are able to outperform the low-level heuristics they manage. The term *hyper-heuristic* is used interchangeably with "HMHH selection operator" in this chapter. This section outlines the overall experimental approach used in this thesis to evaluate the performance and behavior of an assortment of heuristic and hyper-heuristic algorithms.

Section 4.3.1 discusses how the DOP algorithm building blocks outlined in section 2.5 are incorporated into the considered hyper-heuristic approach. Section 4.3.2 outlines a selection of representative meta-heuristics from the SI and EC fields that make up a pool of nine heuristics. Suitable parameter choices are motivated for each heuristic. Considerations for the benchmark function generator and performance measures are discussed in sections 4.3.3 and 4.3.4, respectively. Section 4.3.5 provides an overview of the control methods used in experiments in this thesis. Finally, section 4.3.6 explains the experimental method that is used in the empirical analysis of algorithm performance, and includes a motivation around what sampling sizes to employ based on the use of specific statistical methods and procedures.

4.3.1 Incorporation of DOP Algorithm Building Blocks

Section 2.5 presents the *building blocks* used by successful optimization algorithms to better solve DOPs. The following design decisions are made around support for each of the building blocks in all experiments in this thesis:

- *Detection of environment changes* is out of scope in this thesis to avoid any biased interpretation of performance results due to inaccurate change detection strategies. In this thesis, changes occur periodically at regular known intervals, as stipulated in section 2.4. All algorithms are notified when environment changes occur.

- Each heuristic can *react to change* using any suitable refreshing strategy as set out by the inventors of each given heuristic. For example, the PSO implementations in this thesis rely on the *gbest* topology [59] which connects all particles using a star topology. The *gbest* topology allows all particles to access information about the overall global best particle. Consequently, the *gbest* position memory of each

entity needs to be re-evaluated after environment changes to refresh any outdated information. Furthermore, static configuration information about entities is shared between heuristics, and any dynamic state is reinitialized when an entity is assigned a new heuristic, as discussed in section 3.4.5.

The HMHH framework's *reaction to change* is simply to carry on evaluating feedback from heuristics at time t and to allocate entities accordingly using the configured selection operator. In this thesis, the hyper-heuristics are problem-agnostic and subsequently carry no problem-specific knowledge (strictly speaking, the hyper-heuristic layer is not even aware that the underlying problem is dynamic).

Recent trends identified by Swan *et al.* [168] reveal that an increasing number of hyper-heuristic practitioners consider the use of environment change information as viable knowledge that may be made known to the hyper-heuristic layer, as discussed in section 3.3.2. Access to such types of insights would allow hyper-heuristics that rely on memory and learning mechanisms to react to changes. Informing the hyper-heuristic about environment changes is, however, out of scope in this thesis.

- *Memory management* is handled at both the problem space and the heuristic space levels. In the problem space, methods used in this thesis rely on the *implicit memory* that inherently forms part of population-based methods, such as those found in EC and SI approaches. The use of *explicit memory* schemes (such as maintaining a look-up table of previous good solutions and the problem space conditions that led to them) would produce biased performance in highly cyclical environments, since performance would be a function of memory size, memory update strategy, and problem dynamics. Consequently, *explicit memory* schemes are out of scope in this thesis.

  Similarly, heuristics that rely on the *prediction of environment changes* are not used to avoid any biased interpretation of performance results. Nguyen *et al.* [136] highlight a number of studies that show how the wrong training data, lack of training data, or the very nature of the DOP could all lead to extremely poor performance by wrongly biasing the search to certain areas. The goal of this study is to investigate a broadly applicable hyper-heuristic approach, and not to specialize the method to a subset of DOP types that exhibit predictable behavior.

At the heuristic space level, many of the HMHH selection operators in this thesis maintain implicit memories of learned behavior, for example pheromone concentrations or rank values. This implicit memory is critical to the success of intelligent hyper-heuristics.

- *Multiple sub-populations*: Heuristics in HMHH could potentially be meta-heuristics that each maintain their own sub-populations. In the *global* topology such sub-population assignment information is managed as state variables, as discussed in section 3.4.5. Heuristics that manage multiple sub-populations face challenges in the *island* topology. The heuristic's designers may not have provided a mechanism to alter the heuristic's sub-populations apart from the normal operation of the heuristic. HMHH could assign entities to such a heuristic without the heuristic being able to correctly associate the new entities to any heuristic-specific sub-populations. For certain heuristics it may be tedious or even impossible to add or remove entities at will. A simple example is the Von Neumann particle neighborhood structure in PSO [59] that is very specific in how entities are related.

  To prevent bias in comparing the *island* and *global* neighborhood topologies, only single-population heuristics are used in the heuristic pool in this thesis. While this excludes current state of the art multi-population methods presented in section 2.6, it does allow a more fair performance and behavior comparison between the *island* and *global* topologies.

- *Self-adaptive* heuristics use feedback to dynamically adapt their strategies throughout the search to, for example, adopt a more exploratory or exploitative behavior. Section 2.6 provides many examples of SI and EC methods that alter their behavior based on feedback received from the problem space. Self-adaptive methods such as these are not used as part of the heuristic pool in this thesis to prevent the self-adaptive mechanisms of the heuristics from "competing with" or counteracting the operation of the hyper-heuristic.

  The use of self-adaptation in the heuristic pool together with the adaptation at the hyper-heuristic level has intuitive potential, and methods that optimally combine both adaptation approaches encompass a promising area of research. Future

studies should investigate the trade-off and synergies between the two levels of self-adaptation, especially in the management of which parameters are adapted, and in what manner.

The reasons above exclude many state-of-the-art self-adaptive and/or multi-population methods discussed in section 2.6. However, the main goal of this thesis is to investigate how well various hyper-heuristic selection operators can continually balance computational resources across different population-based meta-heuristics in order to solve a DOP better than the individual meta-heuristics can. The exact makeup of a complementary heuristic pool is complex, as discussed in section 3.4.7, and the overall aim of studying the performance and behavior of different heuristic selection mechanisms is not compromised by excluding (self-)adaptive heuristics.

4.3.2 Heuristic Pool Algorithms

Section 3.2.1 discussed why static parameter tuning is generally not effective for algorithms that solve DOPs. Care should be taken not to exacerbate the *patchwork problem* when dynamically altering the optimization algorithm over time. HMHH is an example of an approach that directly addresses the patchwork problem by grouping together well-proven "units of functionality" (i.e. heuristics) into a coherent system that dynamically adapts the optimization process based on continual feedback from the problem environment.

As outlined in section 3.4.7, it is a non-trivial task to construct a pool of heuristics that are complementary to each other when the aim is solve DOPs. The heuristic pool in this thesis consists of nine meta-heuristics which, together, comprise of diverse methods that approach exploration and exploitation in different ways. No static parameter tuning is performed on any heuristic for any of the 27 MPB environments or any hyper-heuristic for the reasons outlined above. Parameter values for each meta-heuristic are instead chosen based on studies found in literature for each method, as explained below. Setting of parameter values relies on *a priori* knowledge about the domain, such as the dimensions, search bounds, constraints, and any other information that practitioners realistically have access to. This is consistent with the recent views surrounding the role of hyper-heuristics as stated by Swan *et al.* [168].

The heuristics outlined below are configured in the same way across all control groups and the heuristic pools of all hyper-heuristics in all experiments in this thesis.

Charged PSO and Atomic PSO

Section 2.6.2 provided an overview of *charged particle swarm optimization* (CPSO) and *atomic particle swarm optimization* (APSO). Both algorithms use charged particles with modified PSO velocity equations that rely on Coulomb force-like repulsion to maintain diversity throughout the run. The major difference between APSO and CPSO is that 100% of the CPSO swarm contains charged particles while only 50% of the APSO swarm is charged. CPSO has a greater focus on exploration and is stronger in environments with severe spatial changes, while APSO balances exploration and exploitation and is better in environments with severe temporal changes [12][14]. Both variants are included in the heuristic pool to establish how different hyper-heuristics make use of the different behaviors at time t.

The CPSO and APSO heuristics generally follow the standard PSO algorithm shown in algorithm listing 3 in chapter 2 with the appropriate modifications to particle velocity updates. Algorithm listing 5 shows the APSO and CPSO algorithms with the required velocity update modifications for easy reference. The following specific implementation choices are made:

- The standard PSO parameters are set to the generally acceptable values of $c_1 = 1.496180$, $c_2 = 1.496180$, and $w = 0.729844$ as recommended in [55][59]. These values are known to be stable configurations [34]. The *gbest* neighborhood topology is used, which ensures that all entities are always visible to each other.

- Both the CPSO and APSO heuristic implementations use modified PSO velocity updates that include an acceleration term calculated using equation (2.6), as outlined in section 2.6.2.

- Blackwell and Branke [16] explain that extremely large acceleration terms (i.e. equation (2.5)) are possible if two charged particles get too close to each other. They propose clamping the acceleration term as a solution. Both the CPSO and APSO implementations clamp the acceleration term components (in each dimen-

sion) of equation (2.5) to the range $[-100, 100]$. This limits the maximum acceleration in any dimension to the extent of the domain's magnitude.

- In addition to the standard PSO parameters, CPSO and APSO have parameters for the *charge* value, Q, the core limit, R_c, and the perception limit, R_p, of each particle. These parameters are domain dependent. Through inspection of equation (2.6) in the context of the domain size of $(0, 100)^{n_x} \subset \mathbb{R}^{n_x}$, the values of $R_c = 1$, $R_p = 10$, and $Q = 5$ are reasonable choices for this problem domain. With these values, any two charged entities separated by distances in the range $[R_c, R_p]$ will have matching acceleration values (per dimension) in the range $[0.025, 25]$, while any two charged entities separated by a distance less than R_c will have an acceleration value capped at 25 per dimension.

  The value of 25 is chosen arbitrarily as a measurement one quarter the size of the domain. Ultimately, the PSO parameters are problem dependent and careful consideration should be given to each domain.

- The standard PSO approach of using an inertia weight is used in velocity updates, as shown in equation (2.7).

The stand-alone version of APSO and CPSO are respectively labeled as **APSO** and **CPSO** in experiments. Additionally, **S_APSO** and **S_CPSO** represent the speciated versions of APSO and CPSO, respectively, as discussed above.

Quantum PSO

Quantum particle swarm optimization (QPSO) is described in section 2.6.2, and relies on *quantum individuals* that are randomly reinitialized inside a hyper-sphere of radius r_{cloud}. Setting the quantum radius to $r_{cloud} = 25$ results in quantum particle displacement behavior that is comparable to the acceleration experienced between charged particles that are a distance of R_c apart in **APSO** and **CPSO**. Recently, Harrison *et al.* [79] found that relatively small values of $r_{cloud} \approx 1$ help QPSO exploit the problem better, which opens new possibilities for QPSO beyond the classic envisioned usage as a mostly exploratory heuristic.

Algorithm 5 Charged particle swarm optimization algorithm.

Initialize an n_x-dimensional swarm $\mathcal{C}(0)$ of n_s particles
Let f be the optimization function (assume maximization)
Let $\mathbf{y}_i$ be the personal best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\hat{\mathbf{y}}_i$ be the neighborhood best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\mathbf{v}_i$ be the velocity vector of particle i (initialized to $\mathbf{0}$)
repeat
  for each particle $i = 1, \ldots, n_s$ **do**
    //Set the personal best position, $\mathbf{y}_i$, of each particle i
    if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
      $\mathbf{y}_i = \mathbf{x}_i$
    end if
    for each particle j containing particle i in their neighborhood **do**
      //Set the neighborhood best position, $\hat{\mathbf{y}}_i$, of each particle i
      if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}}_j)$ **then**
        $\hat{\mathbf{y}}_j = \mathbf{y}_i$
      end if
    end for
  end for
  for each particle $i = 1, \ldots, n_s$ **do**
    Calculate the acceleration, $\mathbf{a}_i$, of particle i using equation (2.6)
    Update the velocity, $\mathbf{v}_i$, of particle i using equation (2.7)
    Update the position, $\mathbf{x}_i$, of particle i
  end for
until a stopping condition is met
Return the global best solution as the most optimal solution

For the experiments, the labels **QPSO1** and **QPSO25** represent QPSO configured with $r_{cloud} = 1$ and $r_{cloud} = 25$, respectively. The intention is that **QPSO1** serves as a local exploiter while **QPSO25** is set to approximately match the exploitation capabilities of **CPSO** and **APSO**. The speciated versions are denoted as **S_QPSO1** and **S_QPSO25**, respectively. The standard PSO parameters values are used in all QPSO variations, namely $c_1 = 1.496180$, $c_2 = 1.496180$, and $w = 0.729844$ as recommended in

Algorithm 6 Quantum particle swarm optimization algorithm,

Initialize an n_x-dimensional swarm $\mathcal{C}(0)$ of n_s particles
Let f be the optimization function (assume maximization)
Let $\mathbf{y}_i$ be the personal best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\hat{\mathbf{y}}_i$ be the neighborhood best position of particle i (initialized to $\mathbf{x}_i(0)$)
Let $\mathbf{v}_i$ be the velocity vector of particle i (initialized to $\mathbf{0}$)
repeat
  for each particle $i = 1, \ldots, n_s$ **do**
    //Set the personal best position, $\mathbf{y}_i$, of each particle i
    if $f(\mathbf{x}_i) > f(\mathbf{y}_i)$ **then**
      $\mathbf{y}_i = \mathbf{x}_i$
    end if
    for each particle j containing particle i in their neighborhood **do**
      //Set the neighborhood best position, $\hat{\mathbf{y}}_i$, of each particle i
      if $f(\mathbf{y}_i) > f(\hat{\mathbf{y}}_j)$ **then**
        $\hat{\mathbf{y}}_j = \mathbf{y}_i$
      end if
    end for
  end for
  for each particle $i = 1, \ldots, n_s$ **do**
    Update the velocity, $\mathbf{v}_i$, of neutral particle i using equation (2.7)
    Update the position, $\mathbf{x}_i$, of particle i using equation (2.8)
  end for
until a stopping condition is met
Return the global best solution as the most optimal solution

[55][59]. The QPSO algorithm is outlined in algorithm listing 6.

Random Immigrant Genetic Algorithm

The *random immigrant genetic algorithm* (RIGA), discussed in section 2.6.1, is a method that constantly introduces new genetic material in an effort to maintain higher diversity. RIGA represents a different type of diversification operator that uses hereditary information instead of relying on trajectory information as DE or the various PSO variants do.

The key parameter is the *replacement rate*, R_r, of existing entities with randomly reinitialized entities. The recommendations of Grefenstette [72] and Cobb and Grefenstette [35] are to use a replacement rate of $R_r = 0.1$, a very low mutation rate of $M_r = 0.001$ to prevent too much perturbation of solutions, and a crossover rate of $C_r = 0.6$.

Algorithm listing 7 gives the outline of the variant of RIGA used in the experiments of this thesis. A number of design decisions are made to allow RIGA to operate in HMHH, namely:

- The envisioned role of RIGA is to continually enhance diversity of the HMHH framework as a whole. Consequently, it is important that RIGA not jeopardize the existing best solution at time t. *Elitism* is used to promote the top solution managed by RIGA to the new generation without any modification.

- Two-point arithmetic crossover, as proposed by Michalewicz [124], is used to allow RIGA to work with real-valued candidate solution vectors. Each entity in the RIGA population is considered as a main parent. A crossover rate of $C_r = 0.6$ is used to decide if a parent will engage in reproduction. If crossover is triggered, the other parent is selected randomly from the population and a single offspring is created.

- Two different *replacement strategies* are used to decide if an offspring should replace the main parent in the population. *Fittest replacement* selects the fitter of the main parent or the offspring to add to the next generation. *Proportional replacement* uses roulette wheel selection to decide whether the main parent or the offspring survives based on probabilities proportional to each entity's fitness.

Two versions of RIGA are included in the heuristic pool using the notation **RIGA_B** and **RIGA_P** to denote RIGA with fittest replacement and proportional replacement, respectively. The labels of the corresponding speciated versions are **S_RIGA_B** and **S_RIGA_P**, respectively.

Differential Evolution

Although classic *differential evolution* (DE) has been shown to perform poorly in DOPs, numerous extensions have made it possible to use DE on DOPs. Many of the approaches

Algorithm 7 Random Immigrant GA heuristic

Let $t = 0$ count the generations
Let $R_r = 0.1$ be the replacement rate
Let $M_r = 0.001$ be the mutation rate
Let $C_r = 0.6$ be the crossover rate
Initialize an n_x-dimensional population $\mathcal{C}(0)$ of n_s individuals
while a stopping condition is not met **do**
  Evaluate $f(\mathbf{x}_i(t))$ of any unevaluated individual i in $\mathcal{C}(t)$
  Use elitism to promote the top individual in $\mathcal{C}(t)$ to $\mathcal{C}(t+1)$
  Randomly replace a R_r proportion of $\mathcal{C}(t)$ with reinitialized individuals
  Determine if each individual will reproduce using a crossover rate of C_r
  Perform reproduction using two-point arithmetic crossover to create offspring $\mathbf{o}_i(t)$
  Mutate the offspring $\mathbf{o}_i(t)$ using a mutation rate of M_r
  Evaluate the fitness $f(\mathbf{o}_i(t))$ of the offspring
  Select a new population $\mathcal{C}(t+1)$ using either fittest or proportionate replacement
  $t = t + 1$
end while
Return the most fit individual $\mathbf{x}_i(t)$ from $\mathcal{C}(t)$ as the optimal solution

were discussed in section 2.6.1. DE is included in the heuristic pool with the intention that DE will act as a focused exploiter heuristic. The expectation is that the inclusion of DE would foster much faster convergence after environment changes.

The algorithm for the adaptive DE heuristic is outlined in algorithm listing 8. The following design decisions are taken for DE:

- **Scaling factor** β: Segura *et al.* [158] show how state-of-the-art trial vector generation methods are ineffective in situations where high diversity is needed to maintain good exploration. They show that drawing the value for the DE mutation scaling factor, β, from a Cauchy or Gaussian distribution frequently yields superior results to setting the value based on feedback from the search. Consequently, the DE implementation in this thesis uses a dynamic scaling factor, β, that is drawn from a Cauchy distribution, $C(0.5, 0.1)$, with a location factor of 0.5 and a scale parameter of 0.1 as recommended by Segura *et al.* [158].

- **Probability of Crossover** C_r: A low C_r value generally decreases convergence speed yet increases robustness, while a larger C_r value often results in faster convergence [60]. Islam *et al.* [90] propose a crossover probability adaptation scheme that continually draws the crossover rate from a Gaussian distribution. Islam *et al.* show that repeatedly sampling C_r values in this manner yields superior results compared to using static values or using many well-known self-adaptive schemes. The DE implementation in experiments in this thesis uses an adaptive crossover rate sampled from a Gaussian distribution, where $C_r \sim N(0.6, 0.1)$ as proposed by Islam *et al.* In addition, Islam *et al.* [90] also propose a self-adaptive scheme where the mean of the Gaussian distribution is updated based on the number of successful crossover operations achieved each generation. This self-adaptive mechanism is not used in the experiments of this thesis as explained in section 4.3.6.

- **Mutation strategy**: Mezura-Montes *et al.* [123] show that *best/1/bin* is generally a very competitive mutation strategy regardless of the characteristics of the problem being solved, and generally yields robust and high quality results. The recent DE survey by Das *et al.* [42] highlights numerous studies that adaptively incorporate *best/1/bin* as the preferred mutation strategy. Consequently, the heuristic pool in the experiments in this thesis also uses a DE variant based on *best/1/bin*.

The label **DE_Best1Bin** refers to a DE heuristic configured using *best/1/bin* and **S_DE_Best1Bin** is the speciated counterpart.

DE requires that $n_s > 2n_v + 1$, where n_s is the population size and n_v is the number of difference vectors ($n_v = 1$ for *best/1/bin*) [60]. A HMHH instance configured with a *global* topology allows any DE heuristic to have visibility across the entire population regardless of how many entities are assigned to DE, so the minimum entities for DE is set to one. In contrast, a DE heuristic requires a minimum of four assigned entities at any time t in an *island* topology to avoid the DE population size from ever falling below $2n_v + 1$.

Gaussian Mutation Operators

Simple Gaussian mutation operators (GMOs) are primarily used as heuristics in early studies that apply hyper-heuristic to solve DOPs [99][100][143][171]. GMOs offer an

Algorithm 8 Differential Evolution with adaptive scaling factor and crossover rate

Let $t = 0$ count the generations
Let $\beta \sim C(0.5, 0.1)$ be the adaptive scaling factor, continually resampled before each use
Let $C_r \sim N(0.6, 0.1)$ be the adaptive crossover rate, continually resampled before each use
Let $\mathcal{M}$ be the *best/1/bin* DE scheme
Initialize an n_x-dimensional population $\mathcal{C}(0)$ of n_s individuals
while a stopping condition is not met **do**
    Evaluate $f(\mathbf{x}_i(t))$ of each unevaluated individual $\mathbf{x}_i$ in $\mathcal{C}(t)$
    Create the trial vector using the mutation strategy $\mathcal{M}$
    For each $\mathbf{x}_i$, generate an offspring $\mathbf{x}'_i(t)$ using crossover probability of C_r
    if $f(\mathbf{x}'_i(t)) > f(\mathbf{x}_i(t))$ **then**
        Add $\mathbf{x}'_i(t)$ to $\mathcal{C}(t+1)$
    else
        Add $\mathbf{x}_i(t)$ to $\mathcal{C}(t+1)$
    end if
    $t = t + 1$
end while
Return the most fit individual $\mathbf{x}_i(t)$ from $\mathcal{C}(t)$ as the optimal solution

unbiased method of modifying a candidate solution vector in a predictable manner. Each GMO is configured with a zero mean and a specific standard deviation value σ. Different values of σ result in *a priori* knowledge of the statistical probabilities that any mutated candidate solution moves, on average, a distance of 1σ, 2σ, or 3σ away from the current position of the entity [162]. Two GMO instances are used in the heuristic pool, namely **Gauss1** and **Gauss10** with $\sigma = 1$ and $\sigma = 10$, respectively.

When an entity is mutated, the new candidate solution (i.e. position) may or may not be superior to the entity's current position. The new position may not even be located within the search space bounds. The approach taken in this thesis is to check if a new position falls within the search space bounds. Any dimensions of the position that lie outside the search space are clamped (i.e. are assigned the value 0 or 100, respectively, if the mutated dimension lies either below or above the domain of $[0, 100]$). A mutated position is only accepted if the fitness of the new candidate solution is equal to or better than the fitness of the entity's current position. If the new position is not deemed an

Algorithm 9 Gaussian mutation-based heuristic

Let $t = 0$ count the generations
Let the standard deviation be either $\sigma = 1$ or $\sigma = 10$, respectively
Initialize an n_x-dimensional population $\mathcal{C}(0)$ of n_s entities
while a stopping condition is not met **do**
  Evaluate $f(\mathbf{x}_i(t))$ of each unevaluated entity $\mathbf{x}_i$ in $\mathcal{C}(t)$
  Create offspring $\mathbf{x}'_i(t)$ from $\mathbf{x}_i(t)$ using Gaussian mutation with distribution $N(0, \sigma)$
  if $\mathbf{x}'_i(t)$ falls outside the domain boundary constraints **then**
    Clamp each dimension of $\mathbf{x}'_i(t)$ to the domain
  end if
  if $f(\mathbf{x}'_i(t)) > f(\mathbf{x}_i(t))$ **then**
    Add $\mathbf{x}'_i(t)$ to $\mathcal{C}(t+1)$
  else
    Add $\mathbf{x}_i(t)$ to $\mathcal{C}(t+1)$
  end if
  $t = t + 1$
end while
Return the most fit entity $\mathbf{x}_i(t)$ from $\mathcal{C}(t)$ as the optimal solution

improvement, then the current entity position is left unmodified. The algorithm outline for the Gaussian mutation operator heuristic is given in algorithm 9.

Strictly speaking, a GMO is not a population-based search algorithm: each entity is mutated and updated in isolation without relying on any information shared by other members of the population. Instead, the GMO update logic is simply applied individually to every assigned entity in a sequential manner. In that sense, even a single-point search heuristic such as a GMO can operate within the multi-point search HMHH framework as a population-based heuristic.

4.3.3 Benchmark Function Generator

The MPB is used to create representative instances of the 27 different types of DOPs as presented in section 2.4.1. The MPB parameters for all 27 environments are set to the values presented in tables 2.1 and 2.2.

Each heuristic and hyper-heuristic algorithm configuration is run on multiple randomized instances for each of the 27 DOP types. While each problem instance is created randomly, every problem instance is recreated identically for each algorithm. Every algorithm has to solve the same initial search landscape with the same exact peaks which all change in the exact same way over time. The implementation relies on a Scala monadic construct defined in the *CIlib* library[1] that allows algorithms and problem landscapes to utilize completely independent streams of randomness (or entropy). The exact same problem landscape and subsequent changes can be recreated for every algorithm, regardless of any stochastic algorithm behavior.

Starting positions are randomized across the instances and across the 27 different problem types. Entity positions are initialized within the search space bounds in a uniform random manner. Moreover, each algorithm solving the same problem instance always shares the exact same starting location for all entities.

4.3.4 Performance Measures

The global optimum value of each MPB instance is known at each point in time t. The exact time steps where environment changes occur are also known in every problem instance. Consequently, the fitness and error values of the best entity in every iteration are recorded for every algorithm in each problem instance. For each run, the n_c best fitness values in the iteration just before environment changes occur are extracted as a vector of n_c fitness values. The *relative error* measure, $P_{RE}(t)$, is used to normalize each of the n_c fitness values of a run using equation (2.18). This results in each run are represented as an ordered vector with n_c components consisting of the $P_{RE}(t)$ values just before environment changes occur.

The *relative error distance* measure, P_r, is used to compute the n_c-dimensional distance of the sequence of $P_{RE}(t)$ values to the theoretical perfect performance score using equation (4.1), as proposed in section 4.2.

[1] See https://cilib.net.

4.3.5 Control Methods

It is vital to determine if any increased performance of any hyper-heuristic is simply due to random chance, speciation across multiple instances of the same heuristic, or a result of using an intelligent selection operator and/or architecture. As recommended by Karafotias *et al.* [97], multiple control groups are used in this thesis:

1. *Single-population stand-alone* configurations of the nine individual heuristics that are used to determine what the baseline performance of individual heuristics are without any hyper-heuristic interference. A comparison of the results of the hyper-heuristics against the baseline results of the individual heuristics tests whether hyper-heuristic selection yields any benefit over simply using the individual heuristics in isolation.

2. *Homogeneous speciation* configurations where the *same* heuristic algorithm is used to manage nine independent, fixed sub-populations of entities. Speciation baseline results are used to test whether any increased performance of a hyper-heuristic is due to speciation alone, or if intelligent heuristic selection increases performance beyond simply using multiple independent sub-populations. The population of entities is divided into nine disjoint sub-populations. Each sub-population is independently managed by the same heuristic. Entity allocations remain fixed, and no entities are ever reassigned between the sub-populations.

3. *Fixed heuristic allocation* methods, denoted in experiments by the labels **IFix** and **GFix**, assign entities uniformly across the nine heuristics at the start of the algorithm run, and then never change heuristic allocations. **IFix** uses the *island* neighborhood topology while **GFix** uses the *global* topology. **IFix** and **GFix** test whether intelligent hyper-heuristic selection yields any benefit over not taking any action at all by simply spreading entities allocations out equally over multiple different heuristics.

4. *Random heuristic selection*, denoted as **Rand** in experiments, acts as a randomized control strategy to test whether intelligent heuristic selection raises performance above randomly assigning heuristics to entities.

The same number of entities is always used in the overall entity population of each configuration, namely 50.

As an important point to reiterate: the precise composition of the heuristic pool is out of scope in this study. Section 3.4.7 elaborated on the importance of selecting a pool of complementary heuristics for each hyper-heuristic. While numerous methods exist to assess how well heuristics complement each other to solve static optimization problems, such methods are largely still lacing for DOPs. Since the characteristics of a DOP may change over time, the complementary nature of the pool of heuristics may also change over time. Heuristics that worked well together in certain types of landscapes and/or dynamics at some point in time may not work well in subsequent landscapes.

To avoid making the investigation intractable, no separate control group is defined to account for different heuristic pool configurations. The same fixed pool of heuristics is used by all hyper-heuristics at all times. Qualitative assessment of the domain and heuristic algorithms was used to select a pool of heuristics that complement each other with diverse types of behavior. Since each algorithm in this thesis operates the exact same, reproducible problem instances (as explained in section 4.3.3), no algorithm is advantaged or disadvantaged by having a different set of heuristics than another algorithm.

4.3.6 Experimental Method

The overall experimental approach in this thesis generally relies on an empirical analysis of algorithm performance and behavior. Algorithms comprise of the control group configurations that are described in section 4.3.5, namely stand-alone heuristics, speciated heuristics, or configurations of the HMHH framework with different selection operators. Altogether the thesis investigates nine stand-alone heuristics, nine speciated heuristics, two configurations where heuristic allocation is fixed across all heuristics, and 21 triggered hyper-heuristic configurations where HMHH is configured with different heuristics selection operators.

The empirical analysis in chapters 4, 5, and 6 is arranged as a series of focused research questions. Each research question presents self-contained results that answer specific inquiries. The results of research questions may build on previously answered research questions, and results may be referred to in future chapters. The analysis

generally relies on the nonparametric statistical tests as outlined in section section 2.7.4. Each research question may also describe a bespoke method of analysis that is used only in that specific research question. All nonparametric statistics are computed using the STAC platform [154] and the SciPy platform [94].

A common type of analysis employed in many of the research questions is to compare the performance of all algorithms in a group against each other. The Friedman nonparametric test and Shaffer post hoc test, presented in section 2.7.4, is used to assess if significant performance differences are present between algorithms. Derrac *et al.* [48] and Garcia *et al.* [68] recommend that the number of samples, s, used for any Friedman nonparametric test-based analysis must satisfy $2a \leq s \leq 8a$, where a is the number of algorithms being compared. Consequently, the choice of $s = 71$ algorithm run samples per environment type simultaneously satisfies the requirements for the analysis of the nine stand-alone heuristics (i.e. $a = 9$ satisfies $s \leq 8 \times 9$), the analysis of 21 hyper-heuristics (i.e. $a = 21$ yields $s \geq 2 \times 21$), as well as any joint analysis of all heuristics and hyper-heuristics (i.e. $a = 32$ results in $s \geq 2 \times 32$). Figure 4.3 illustrates the different algorithm groups and the number of run samples resulting from the above process.

Research questions that rely on the Friedman and Shaffer tests report performance using the following types of tables and graphs:

1. **Average Friedman ranks:** The output of a Friedman test is the average rank for each algorithm and a statistical p-value. The p-value indicates the propensity that the differences observed in the group of algorithms is coincidental (i.e. that the algorithms are different while the null hypothesis holds true). Smaller p-values are indicative that the algorithms are truly significantly different from one another. An example of Friedman ranks can be seen in table 4.3.

2. **Wins-draws-losses tables:** The Friedman test only shows whether significant differences are present in the group of algorithms. For each possible pair of algorithms, the Shaffer post hoc test determines whether one algorithm is significantly better than the other or not. A win and a loss is allocated accordingly. If any comparison shows that there is no significant difference between the two algorithms, then both algorithms receive a draw.

   Each algorithm's wins, draws, and losses against all other algorithms in the group

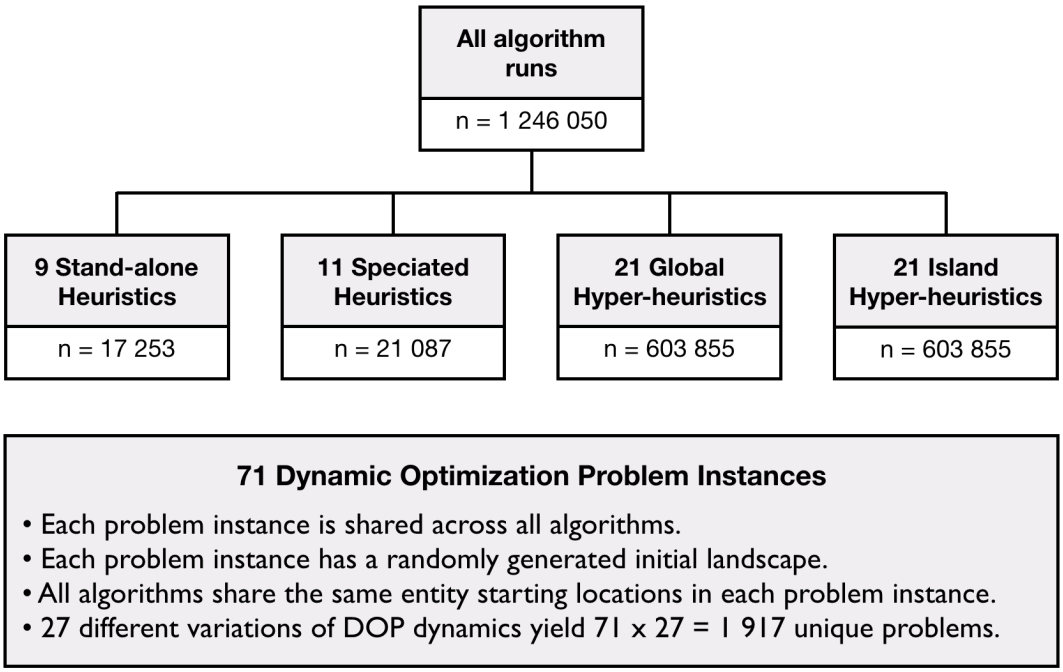


Figure 4.3: Outline of the different algorithm groups and the number of executions performed for each group.

can be expressed as a 3-tuple. The *wins-draws-losses* tuple for each algorithm in each environment can be succinctly shown in a single table. An example can be seen in table 4.4.

3. **Sankey diagrams:** Sankey diagrams [156] offer an intuitive visual overview of the Shaffer post hoc test results. For each algorithm, the proportion of comparisons that yield either wins, draws, or losses across all 27 environments are shown as *stream flows* from left to right. Thicker lines indicate that a higher proportion of the comparisons follow a particular path. An example of a Sankey diagram can be seen in figure 4.12.

4. **Box whisker plots:** A box whisker plot shows the distribution of a set of values as vertical bars, where the upper and lower ends of each bar correspond to the 75^{th} and 25^{th} percentiles of the distribution, respectively. The median value is represented as a horizontal line inside each bar. The upper and lower whisker lines,

respectively, represent the values that fall within 1.5 *interquartile range* (IQR)[2] from the median. Box whisker plots are used extensively in chapters 4, 5, and 6 to show value distributions for performance scores, diversity measures, Friedman ranks, and wins in Shaffer post hoc tests. An example of a box whisker plot can be seen in figure 4.6.

Each research question in chapters 4, 5, and 6 may employ one or more of the types of tables and graphs outlined above, along with bespoke graphs that are introduced as appropriated.

4.4 Experimental Results

The results of all experiments are organized into the distinct research questions listed in section 4.4.1. All stand-alone heuristics and speciated heuristics were configured and executed using the parameters, benchmark functions, and algorithm choices outlined in section 4.3.

4.4.1 Research Questions

The following research questions drive the experimental analysis:

1. *Do the characteristics of the observed error values in experiments warrant the need for the proposed relative error distance performance measure (P_r)?* Do problem fitness scale changes warrant the need to normalize error values? Do the measured error values of each individual algorithm run follow a normal distribution or not? Does the resulting collection of P_r values across all samples of each algorithm follow a normal distribution? How do P_r values relate to statistical aggregation methods of the underlying error values such as the mean, standard deviation, or (nonparametric) measures such as interquartile range (IQR) or range [162]?

2. *What is the performance of each stand-alone heuristic?* How well does each heuristic solve a DOP when applied individually without any hyper-heuristic adaptation?

[2]The IQR is the range between the 25^{th} and 75^{th} percentiles [162].

Which heuristics have significantly different performance scores than other heuristics across the different types of DOPs? Which heuristics do not show any significant differences in performance? Do the heuristics achieve their envisioned goals?

3. *What is the performance of each speciated heuristic?* How well do the speciated versions of each heuristic perform in the same analysis described in the previous research question above?

4.4.2 Research Question 1

Do the characteristics of the observed error values in experiments warrant the need for the proposed relative error distance performance measure (P_r)?

The relative error distance, P_r (proposed in section 4.2), offers significant benefits over existing performance measures, such as using normalized error values to provide isolation from fitness scale fluctuations between environment changes, not assuming normal distributions in the underlying performance data, incorporating the variance across performance values in a run, and yielding a representative scalar value that allows for easy significance testing using established statistical procedures.

This research question investigates whether the error data generated using the experimental approach proposed in section 4.3 displays characteristics that justify the need for using P_r instead of existing performance measures. Specifically, the analysis considers the sequence of error values consisting of those iterations just before environment changes occur. In order for the new P_r measure to provide additional value, the following circumstances and/or conditions need to be validated in the data:

1. Are fitness scale changes *actually* present in the data to such an extent so as to warrant the normalization of error values?

2. Do the underlying set of $P_{RE}(t)$ performance values for individual algorithm runs follow a normal distribution?

3. Does the resulting collection of P_r values across all samples of any particular algorithm follow a normal distribution?

4. How do P_r values relate to values generated by statistical aggregations of the underlying error values, such as the mean, standard deviation, or (nonparametric) measures such as interquartile range (IQR) or range [162]?

To answer the questions above, the analysis considers the output of each run of each algorithm in every environment in this thesis. This amounts to 1 246 050 individual algorithm runs, as outlined in section 4.3.6 and illustrated in figure 4.3. The exact settings for the hyper-heuristics are only provided in section 5.2.2. However, the samples can be considered here simply for the purposes of illustrating the working of the P_r measure.

Confirming the severity of fitness scale changes across DOP types

The MPB dynamics produced a global optimum fitness value for each search landscape between environment changes. Over time, each instance contained up to n_c unique global optimum values. The global optimum value in any given MPB instance could differ dramatically in the range $[30, 70]$, as shown in table 2.1. The minimum fitness value that any algorithm could produce was zero (the base landscape function value), which would have happened if an algorithm did not manage to locate *any* of the MPB peaks. Therefore, for any given algorithm run, the subsequent range of possible fitness values could have been as low as $[0, 30]$ and as high as $[0, 70]$.

The ratio between the maximum and minimum fitness scale range that was observed in any given problem instance characterizes the severity of scale changes. A low ratio that was approximately equal to 1.0 implies that the fitness scale did not change dramatically over time in the problem instance. A larger ratio means that the fitness scale did change dramatically over time. In such cases, any direct comparisons of error or fitness values would result in a skewed view of performance. For example, if the ratio between the maximum and minimum fitness scale range was 1.4, then any direct examination of (or measure that depends on) a raw error value could be misinterpreted by a factor of up to 1.4.

Figure 4.4 shows the distribution of fitness scale ratios across the 71 sample instances of each of the 27 environment types (as defined in table 2.2). Bear in mind that there were 71 samples and that each sample's random number generator was linked across the

27 environments (as explained in section 4.3.6). This ensured the exact same stream of entropy was used to create each sample across the 27 environment variations. The only difference (per sample) across the 27 environments was the *dynamics* that governed peak movements.

The following points are noteworthy pertaining to the observed fitness scale ratios:

1. The 27 environment types showed varied fitness scale ratios which ranged from 1.0 to as high as approximately 1.6. The bulk of observed fitness scale ratios were located roughly in the interval $[1.1, 1.3]$ for the majority of environment types.

2. All nine environments that comprised of *Type I* dynamics according to the classification of Eberhart *et al.* [56][84] showed no fitness scale changes (i.e. all ratios were equal to 1.0), since no peak height changes occurred.

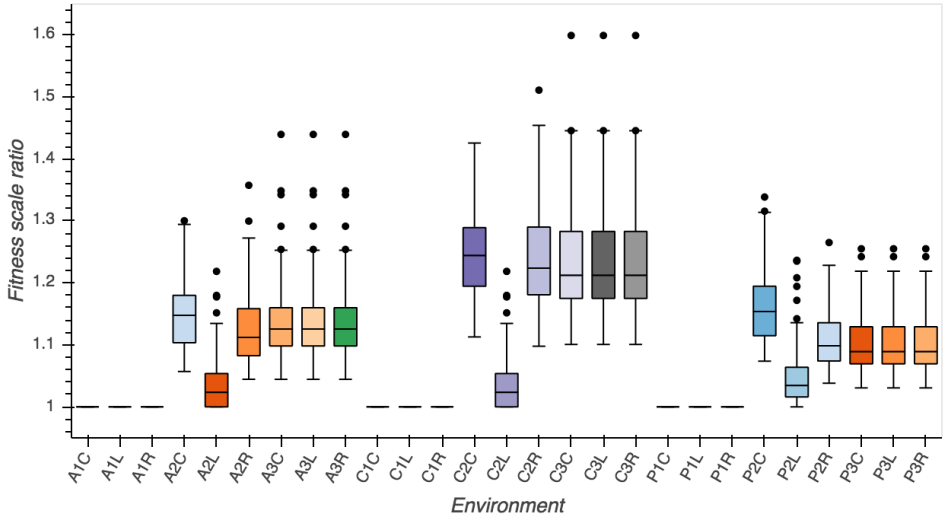


Figure 4.4: Observed fitness scale ratios for each environment type. Each environment had 71 different environment samples.

3. All *Type II* environments (where only peak heights changed) showed different fitness scale ratio distributions. Those *Type II* environments that had *linear* peak height change dynamics showed noticeably lower ratios. This makes sense, given that peak heights either increased or decreased asymptotically towards either an upper or lower bound, which resulted in more uniform peak heights compared to

random or *circular* change patterns. Fitness scale changes were small accordingly, which yielded low ratios with a tighter distribution.

4. All *Type III* environments showed exactly the same fitness scale distributions across *circular*, *linear*, and *random* environment types (within *abrupt*, *chaotic*, and *progressive* environments, respectively). This observation is expected, since all three variations in each group shared the same height severity settings and had the same stream of randomness. The only difference between the *circular*, *linear*, and *random* environment types consisted in how peak locations moved around across the search space.

5. *Abrupt*, *chaotic*, and *progressive* environments showed distinct differences in the distributions of fitness scale ratios. The reason for this is that the number of environment changes differed between the three types of dynamics, which yielded a different number of peaks for *abrupt*, *chaotic*, and *progressive* environments, respectively. This yielded more opportunity to create fitness scale changes, which is reflected in the plot.

6. *Progressive* environments had smaller height severity parameter values, which resulted in narrower distributions of fitness scale ratios. *Abrupt* environments had the same number of environment changes as *progressive* environments, but the much larger height severity parameter setting for *abrupt* environments resulted in a large difference in fitness scale ratios between the two types of environments.

It is clear that fitness scale normalization was required for the observed data due to fitness scale ratio differences of up to 1.6. Simpler measures that rely on non-normalized raw error or fitness values may have misreported performance by a factor of up to 1.6. The P_r measure relies on the relative error $P_{RE}(t)$, as calculated using equation (2.18), and is resilient against such types of fitness scale changes.

Testing the normality of $P_{RE}(t)$ values for each individual algorithm run

Collections of values that do not follow a typical Gaussian "bell curve" distribution tend to give spurious results when parametric statistical measures, such as a mean or standard

Table 4.1: Percentages of individual algorithm runs (at various significance levels) that showed non-normal distributions of $P_{RE}(t)$ error values across all DOP change periods.

Group	**Runs**	$p < 0.05$	$p < 0.01$	$p < 0.001$	$p < 0.0001$
Stand-alone heuristics	17 253	12 289 (71%)	10 780 (62%)	9 115 (53%)	7 795 (45%)
Speciated heuristics	21 087	16 133 (77%)	14 498 (69%)	12 582 (60%)	10 878 (52%)
Global hyper-heuristics	603 855	503 429 (83%)	460 854 (76%)	405 277 (67%)	346 838 (57%)
Island hyper-heuristics	603 855	510 559 (85%)	469 085 (78%)	414 362 (69%)	362 579 (60%)

deviation, are used to aggregate, describe, or summarize the data [162]. The Shapiro-Wilk normality test [161] assesses whether a set of values is drawn from a normally distributed population. The null hypothesis of the test is that the data is normally distributed. The output of the test is a p-value that represents the propensity that a distribution *appears* non-normal but is, in fact, drawn from a normal distribution. In other words lower p-values indicate a higher degree of certainty that the data sample is drawn from a non-normal distribution. Razali and Wah [152] show that the Shapiro-Wilk test has high statistical power compared to other prominent normality tests.

An individual Shapiro-Wilk normality test was conducted for each of the 1 246 050 algorithm runs. The test was applied to the sequence of $P_{RE}(t)$ values generated in the iterations just before environment changes occurred. Table 4.1 shows the number of tests performed and the proportion of tests that resulted in p-values less than a significance level of $\alpha \in [0.05, 0.01, 0.001, 0.0001]$. The results in table 4.1 confirm that the majority of measured $P_{RE}(t)$ values were not normally distributed. Even at a significance level of $\alpha = 0.0001$ roughly half of all samples failed the normality test.

The implication is that any performance measures that rely on parametric statistical descriptions of error values (such as the mean or the standard deviation) are almost certainly invalid and/or unreliable approximately half of the time (if not more often). The P_r measure is an appropriate measure to characterize the sequence of $P_{RE}(t)$ values of an algorithm run, since the measure does not require or assume normality in the underlying error values.

Table 4.2: Percentages of algorithm run samples (at various significance levels) that showed non-normal distributions of P_r values across the 71 algorithm runs in each sample. Runs were made up using 9 stand-alone heuristics, 11 speciated heuristics, and 21 hyper-heuristics using, respectively, three different triggers together with the *global* and *island* topology (refer to figure 4.3). The exact parameters are not important here, only the number of algorithm sample sets.

Group	**Sample sets**	$p < 0.05$	$p < 0.01$	$p < 0.001$	$p < 0.0001$
Stand-alone heuristics	243	211 (87%)	193 (79%)	165 (68%)	144 (59%)
Speciated heuristics	297	270 (91%)	256 (86%)	228 (77%)	182 (61%)
Global hyper-heuristics	8 505	7 654 (90%)	7 028 (83%)	5 848 (69%)	4 062 (48%)
Island hyper-heuristics	8 505	8 023 (94%)	7 547 (89%)	6 712 (79%)	5 418 (64%)

Testing the normality of P_r values of all algorithm samples

The normality of the resulting set of P_r scores of all 71 runs of each algorithm also had to be determined. A separate Shapiro-Wilk test was conducted on each distribution of resulting P_r values from the 71 sample runs of each algorithm for each environment. Table 4.2 confirms that the resulting distribution of P_r scores of repeated algorithm executions followed non-normal distributions more often than not. The nonparametric statistical analysis methods and procedures outlined in section 2.7.4 are a better choice to analyze the performance of the data, given the strong tendencies of the data to be non-normally distributed.

Relationship between P_r and common statistical measures of the error

The proposed P_r measure promises to produce a single, yet representative, scalar value that better reflects the nature of the sequence of error values. To be representative, the notions of density and variance of the sequence of $P_{RE}(t)$ values must be faithfully characterized by P_r. Specifically, an inverse correlation should be evident between P_r and the mean of the sequence of $P_{RE}(t)$ values: higher mean $P_{RE}(t)$ values should be correlated with lower P_r distances to the theoretical point of perfect performance. There should also be a positive correlation between P_r and the standard deviation, interquartile range (IQR), and range (max - min) of $P_{RE}(t)$ values, since the distance to perfect performance should increase the more volatile the sequence of $P_{RE}(t)$ values is.

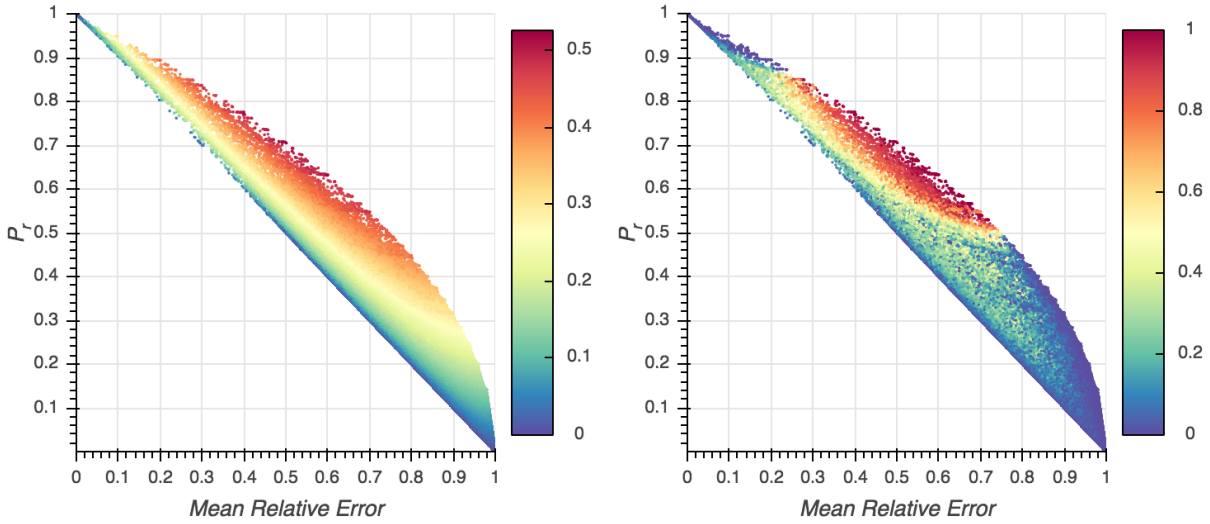
Sub-figure 4.5a shows the relationship between the proposed P_r measure (y-axis) and the mean of the sequence of $P_{RE}(t)$ values (x-axis). Every dot in the graph represents a single algorithm run, and all 1 246 050 algorithm runs are represented. The standard deviation, σ, of each sequence of $P_{RE}(t)$ values is represented by a color spectrum: dark blue indicates $\sigma \approx 0$ and red indicates $\sigma \approx 0.5$. Sub-figures 4.5b, 4.5c, and 4.5d show the same graphs, but with color representing the IQR, statistical skew, and range of the distribution, respectively.

The following observations are drawn from the charts in sub-figures 4.5a, 4.5b, 4.5c, and 4.5d:

1. A striking feature that is present in all sub-figures is the "halved teardrop" shape of the relationship between P_r values and mean $P_{RE}(t)$ values. The teardrop shape is split in half by the perfect linear relationship that is observable between P_r and mean $P_{RE}(t)$ values (i.e. the diagonal line in each sub-figure). Points near the diagonals showed approximately zero values for standard deviation, IQR, and range sizes. These runs corresponded to points near the diagonal in figure 4.1, which resulted in the P_r and mean of the $P_{RE}(t)$ being nearly equal.

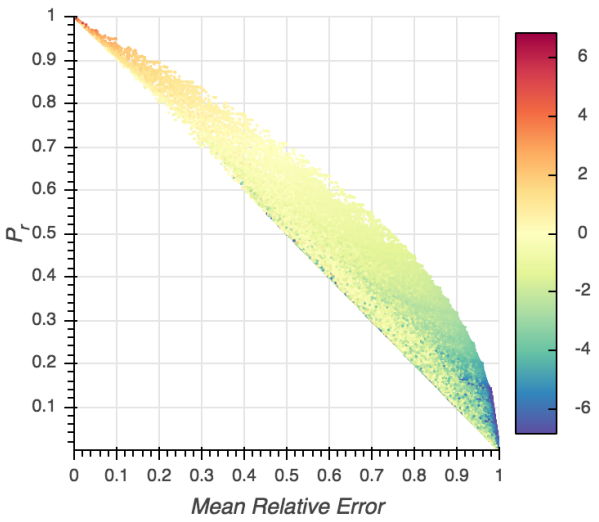
2. Higher volatility in any given sequence of $P_{RE}(t)$ values of a single algorithm run was characterized by higher standard deviation, IQR, and range size values. The sub-figures show that runs with higher volatility (i.e. points that lie near the outer edge of the teardrop) showed P_r values that were up to 0.25 higher than runs with lower volatility (i.e. points that lie near the diagonal). In other words, the P_r measure proportionately penalized runs that showed higher variance in their underlying $P_{RE}(t)$ values.

3. A notable exception to the previous point can be observed in sub-figure 4.5b where mean $P_{RE}(t)$ values lie in the approximate range of $[0.75, 0.95]$: for any given mean $P_{RE}(t)$ value, the associated P_r values showed monotonically increasing IQR values at first, but eventually showed a trend reversal where IQR values decreased to nearly zero again as P_r values increased. This produced a distinct visual effect of "blue points curling around the green and yellow points" in the lower-right quadrant of sub-figure 4.5b. The same trend is not visible in the standard deviation values in sub-figure 4.5a, where corresponding standard deviation values for

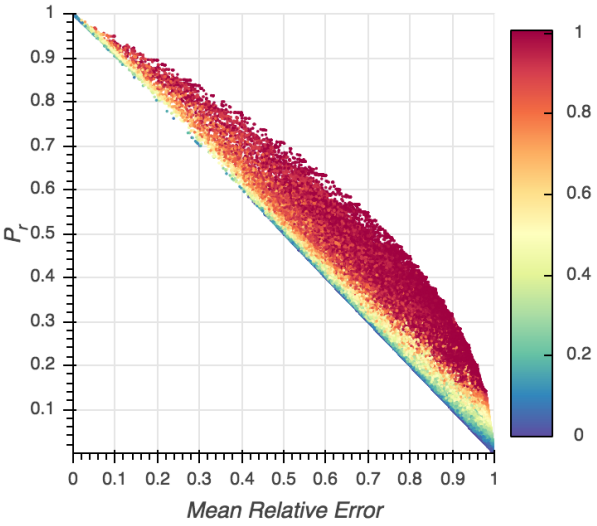


(a) Standard deviation

(b) IQR



(c) Skewness



(d) Range

Figure 4.5: Relationship between P_r and the mean relative error, $P_{RE}(t)$, for all algorithm runs. Colors in sub-figures reflect various statistical measures applied to each individual $P_{RE}(t)$ sequence.

the anomalous IQR values were nearer to the high end of the standard deviation spectrum (yellows and greens).

The explanation lies in the non-normal distribution of values in each sequence of $P_{RE}(t)$ values, as illustrated earlier in this section. A given algorithm run could

have had high mean and standard deviation values for $P_{RE}(t)$, yet still have had a low IQR value for $P_{RE}(t)$. Sub-figure 4.5d validates this conclusion by showing how all the points in question had overall $P_{RE}(t)$ value ranges close to 1.0, but also maintained very high mean $P_{RE}(t)$ values. Sub-figure 4.5c further corroborates this conclusion by showing increasingly negative skewness values (green and blue hues) for the points in question, which reveals that a disproportionate number of high $P_{RE}(t)$ values were present in the distribution of each algorithm run.

The proposed relative error distance performance measure, P_r, offers a distance-based alternative calculation method that is unaffected by the lack of normally distributed data. The new measure better incorporates the variance of the underlying $P_{RE}(t)$ values by increasing the distance to the theoretical optimal point as variance increases. In cases with low variance across the $P_{RE}(t)$ distribution, the P_r measure returns values that are inversely correlated to the mean of the $P_{RE}(t)$ sequence.

4.4.3 Research Question 2

What is the performance of each stand-alone heuristic?

This research question assesses the performance of all individual heuristics running in a stand-alone fashion without any hyper-heuristic managing entity allocations. The goals are to determine the performance of every individual heuristic in each type of DOP as a baseline control group, and to establish if any heuristics have statistically significantly better performance than others for each of the DOPs. The analysis follows the experimental method presented in section 4.3.6. The nine heuristics are compared against each other, both in terms of P_r performance values as well as the *average distance around swarm center* ($ADSC$), as calculated using equation (2.26).

Figures 4.6, 4.7, and 4.8 show box whisker plots of the distribution of P_r values obtained by each heuristic for each environment. Each box whisker bar of P_r values shows the combined distribution over all 71 problem instances. Figures 4.9, 4.10, and 4.11 show the corresponding box whisker plots of the distributions of *average distance to swarm center* ($ADSC$) values of each heuristic for each environment. Each box whisker bar of $ADSC$ values shows the combined distribution of the 1000 individual $ADSC$

values computed at each algorithm iteration t using equation (2.26) over all 71 problem instances.

Table 4.3 shows the average Friedman ranks obtained by each heuristic in each environment. Table 4.4 shows the number of wins, draws, and losses for each heuristic which resulted from a pairwise comparison of all nine heuristics using the Shaffer post hoc test. The significance level was set at $\alpha = 0.05$ for all tests. Figure 4.12 illustrates the resulting wins, draws, and losses of every heuristic across all DOPs as a Sankey diagram. Since there were nine heuristics and 27 DOPs, there was a combined total of 216 pairwise comparisons across all 27 of the Shaffer post hoc tests, as can be seen in figure 4.12.

A number of notable observations, pertaining to the research question at hand, can be made from the figures and tables referred to above:

- The small p-values in table 4.3 strongly suggest that different heuristics displayed significantly varied performance in each environment type. A visual inspection of figures 4.6, 4.7, and 4.8 confirms this to be the case. A glance at the figures reveals that different environments yielded varied performance across the heuristics. For example, **APSO** yielded better P_r distributions than **Gauss10** for most Type I environments, but the two methods performed relatively on par with each other for a number of environments, such as A2C or C3R.

  Table 4.4 shows that no single heuristic was always the best performer across all environment types (i.e. had the most wins and fewest losses). This is corroborated by the ranks in table 4.3, where the best ranks and most wins were distributed across multiple heuristics across different environments.

- Each heuristic clearly had a different average diversity profile, as expressed by the spread of the $ADSC$ values in figures 4.9, 4.10, and 4.11. The shape of the diversity spread pattern across heuristics is strikingly similar across all 27 environment types.

- The heuristic **DEBest1Bin** had notably worse performance than any of the other heuristics in all environments. This makes sense and was expected, since DE is known to perform badly in dynamic environments. The extremely low value and low spread of $ADSC$ values associated with **DEBest1Bin** confirms that the heuris-

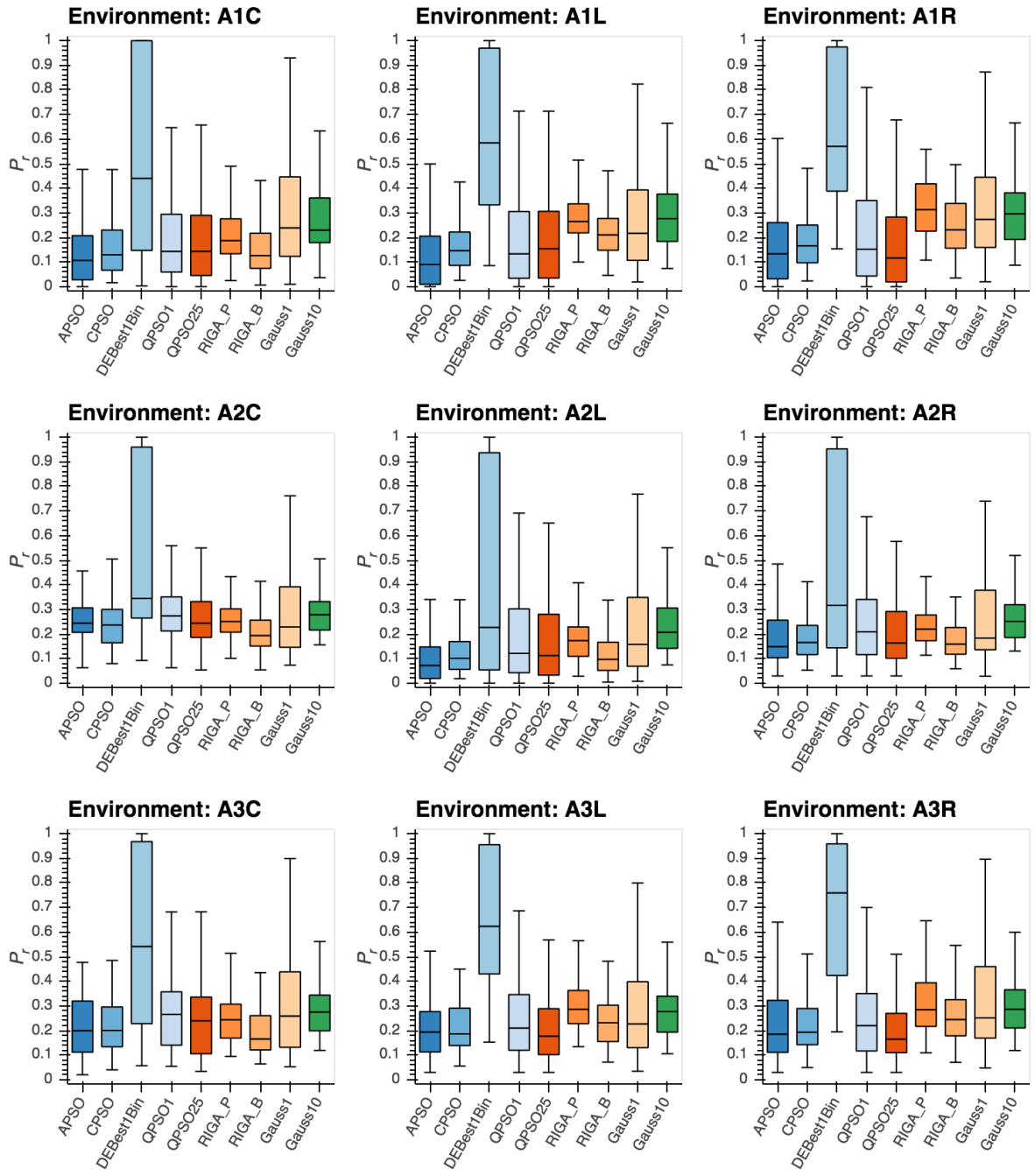


Figure 4.6: P_r values achieved by stand-alone heuristics in all abrupt environments over 71 repeat runs.

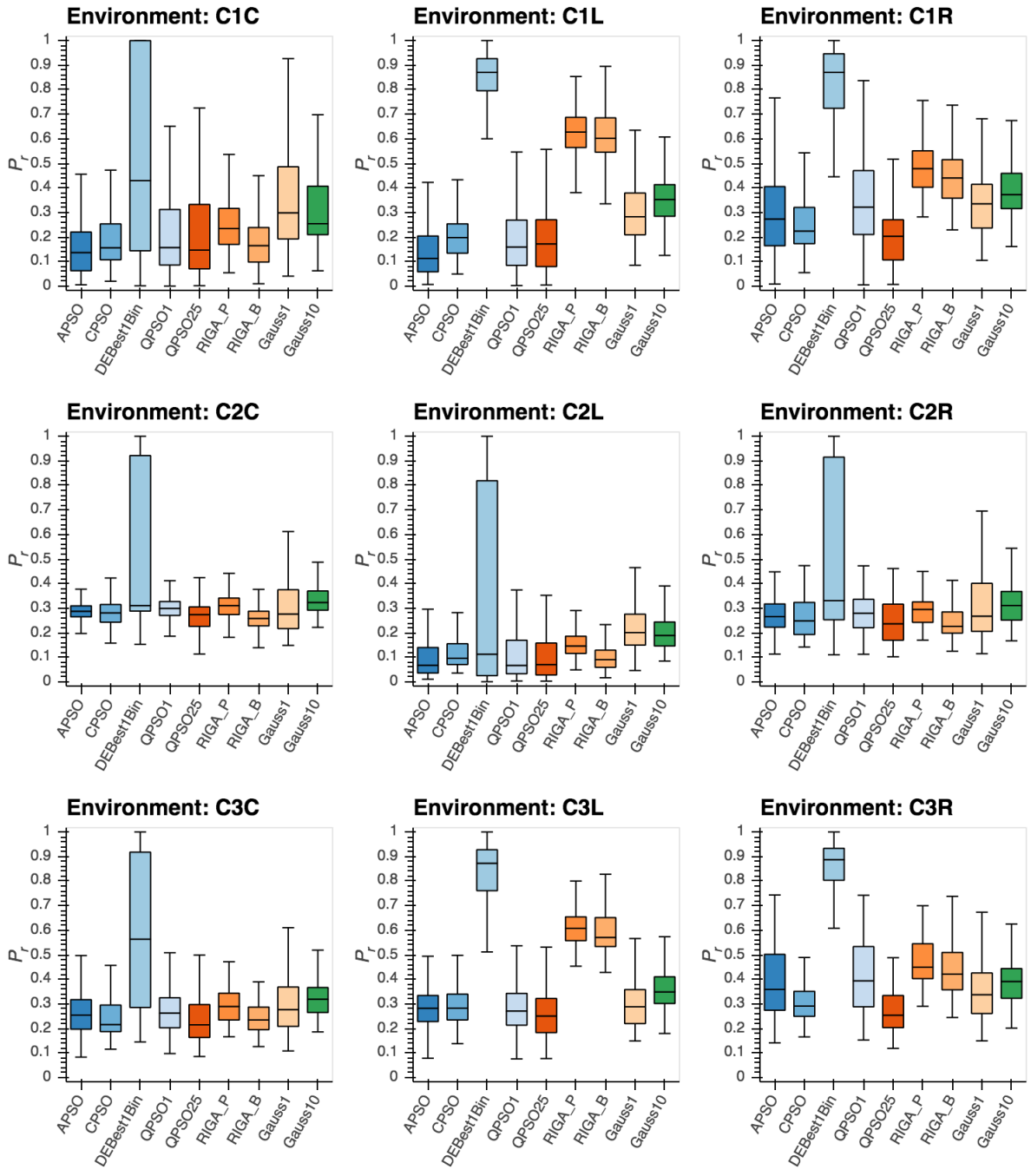


Figure 4.7: P_r values achieved by stand-alone heuristics in all chaotic environments over 71 repeat runs.

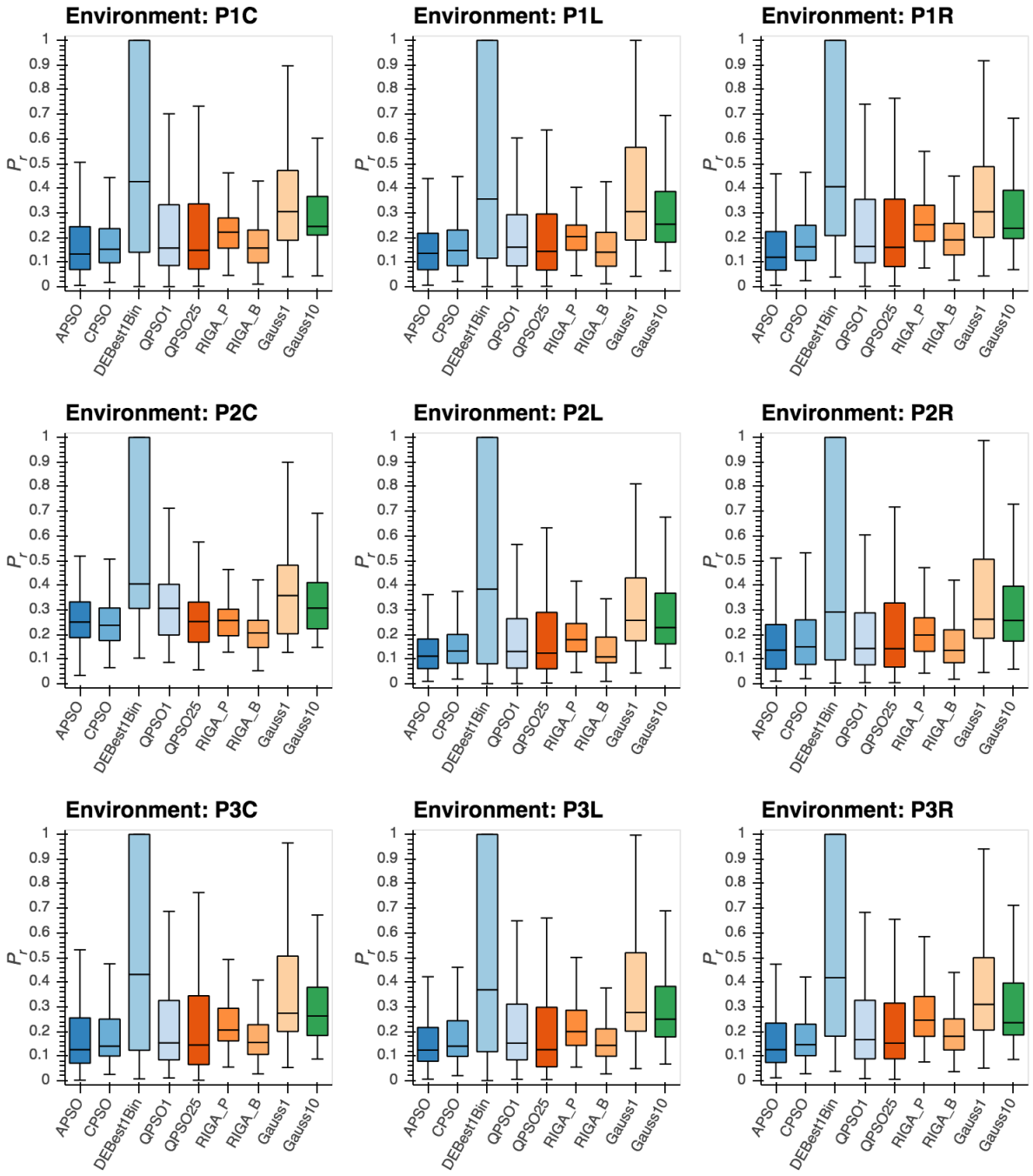


Figure 4.8: P_r values achieved by stand-alone heuristics in all progressive environments over 71 repeat runs.

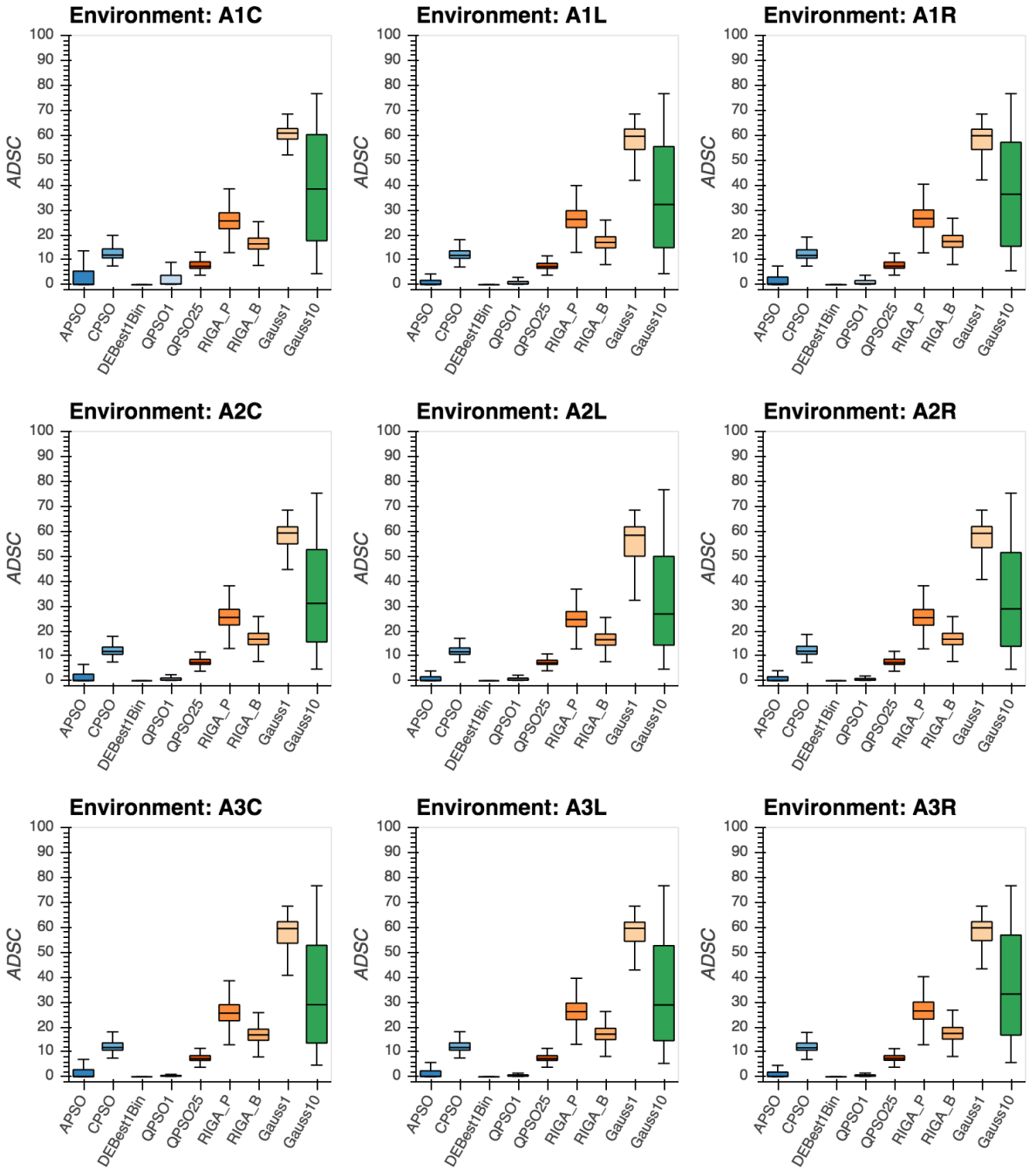


Figure 4.9: *ADSC* values of stand-alone heuristics in each abrupt environment.

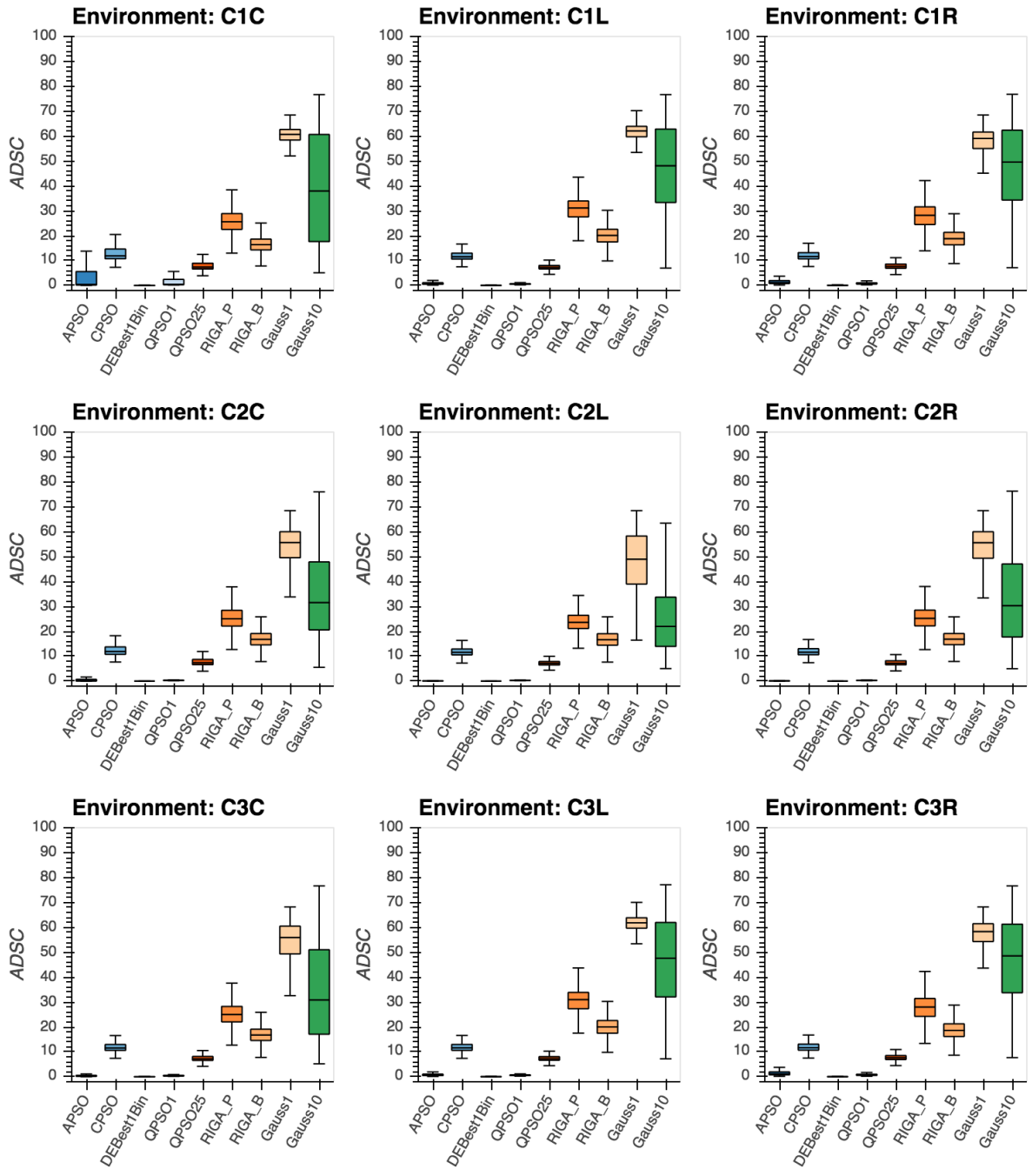


Figure 4.10: *ADSC* values of stand-alone heuristics in each chaotic environment.

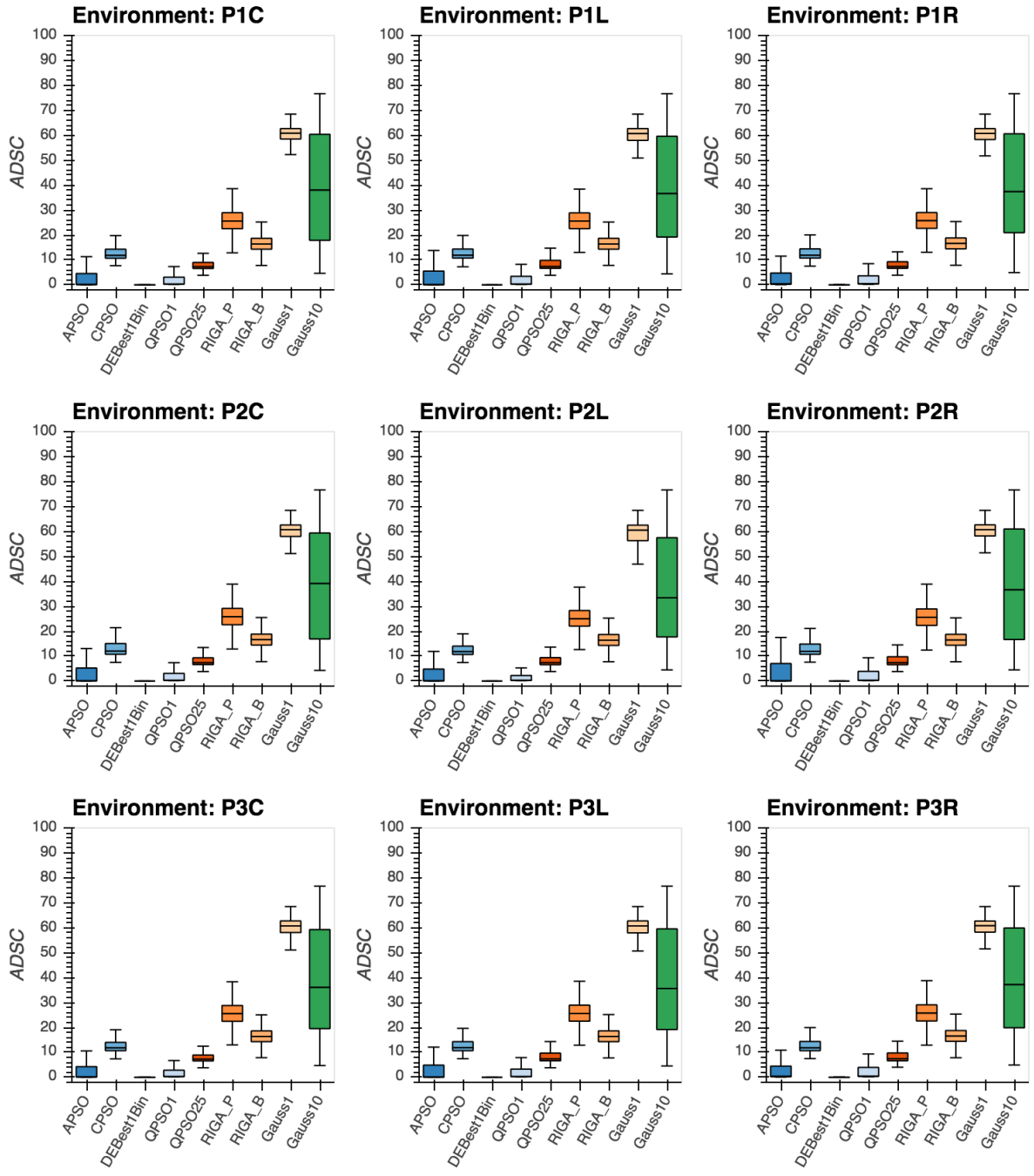


Figure 4.11: *ADSC* values of stand-alone heuristics in each progressive environment.

Table 4.3: Average ranks per environment for each heuristic using the Friedman test, as well as the resulting p-values. Bold values indicate the best performers.

Environment	APSO	CPSO	DEBest1Bin	QPSO1	QPSO25	RIGA_P	RIGA_B	Gauss1	Gauss10	p values
A1C	**2.80**	5.00	7.10	3.20	3.68	5.94	4.18	6.23	6.86	1.11e-16
A1L	**2.72**	4.14	8.46	3.04	3.44	6.55	4.86	5.25	6.55	1.11e-16
A1R	**2.89**	3.94	8.75	3.13	3.17	6.54	4.86	5.44	6.28	1.11e-16
A2C	5.01	4.96	6.77	5.27	4.62	5.18	**2.96**	4.45	5.77	3.44e-15
A2L	**3.39**	4.93	5.25	4.24	4.38	6.07	4.07	5.65	7.01	1.11e-16
A2R	4.11	4.73	5.95	4.68	4.13	5.93	**3.92**	4.78	6.77	7.32e-14
A3C	3.80	4.58	7.70	4.65	3.82	5.55	**3.45**	5.39	6.06	1.11e-16
A3L	3.87	4.17	8.77	3.79	**3.08**	6.46	4.34	4.76	5.75	1.11e-16
A3R	3.62	3.92	8.87	3.79	**3.06**	6.21	4.73	4.92	5.89	1.11e-16
C1C	**2.89**	4.63	6.54	3.86	3.27	5.93	4.15	6.92	6.80	1.11e-16
C1L	**2.06**	3.68	8.82	2.48	2.54	7.55	7.42	4.89	5.56	1.11e-16
C1R	3.59	2.87	8.94	4.41	**1.97**	7.18	6.01	4.61	5.41	1.11e-16
C2C	4.72	4.34	6.59	5.31	3.32	6.23	**3.07**	4.63	6.79	1.11e-16
C2L	**3.30**	4.79	4.68	3.87	3.48	6.14	3.82	7.42	7.51	1.11e-16
C2R	4.83	4.15	6.51	5.24	3.34	5.97	**3.23**	5.38	6.35	1.11e-16
C3C	4.25	3.72	7.73	4.55	**3.06**	6.10	3.51	5.42	6.66	1.11e-16
C3L	3.14	3.68	8.82	2.97	**2.42**	7.75	7.38	3.77	5.07	1.11e-16
C3R	4.48	2.93	8.99	5.11	**2.00**	6.86	5.79	3.92	4.93	1.11e-16
P1C	**3.23**	4.66	6.27	3.82	3.42	5.62	4.01	7.25	6.72	1.11e-16
P1L	**3.48**	4.48	5.63	4.11	3.68	5.59	3.94	7.35	6.75	1.11e-16
P1R	**2.51**	3.94	7.54	3.83	3.60	6.14	4.07	6.91	6.46	1.11e-16
P2C	4.46	4.44	6.62	5.27	4.05	4.86	**2.66**	6.39	6.24	1.11e-16
P2L	**3.30**	4.63	5.51	3.85	3.70	5.75	3.99	7.35	6.93	1.11e-16
P2R	**3.54**	4.56	5.52	4.05	3.80	5.63	3.90	7.11	6.89	1.11e-16
P3C	**3.14**	4.65	6.33	3.81	3.33	5.76	4.20	7.08	6.70	1.11e-16
P3L	**3.32**	4.56	5.95	4.20	3.41	5.70	3.86	7.12	6.87	1.11e-16
P3R	**2.55**	3.93	7.67	3.97	3.35	6.23	4.15	6.80	6.35	1.11e-16
Mean rank	3.52	4.26	7.12	4.09	**3.38**	6.13	4.32	5.82	6.37	–

Table 4.4: Wins, draws and losses per environment of each heuristic. The notation *W-D-L* indicates the number of wins, draws, and losses for each heuristic in each environment. Bold values indicate the best performers.

Environment	APSO	CPSO	DEBest1Bin	QPSO1	QPSO25	RIGA_P	RIGA_B	Gauss1	Gauss10
A1C	**6-2-0**	2-4-2	0-3-5	5-3-0	4-4-0	0-4-4	4-3-1	0-4-4	0-3-5
A1L	**6-2-0**	3-4-1	0-0-8	5-3-0	5-3-0	1-2-5	3-2-3	1-4-3	1-2-5
A1R	5-3-0	4-4-0	0-0-8	5-3-0	5-3-0	1-2-5	3-2-3	1-3-4	1-2-5
A2C	1-6-1	1-6-1	0-1-7	1-6-1	1-6-1	1-6-1	**8-0-0**	1-6-1	0-7-1
A2L	**5-3-0**	1-6-1	1-6-1	3-5-0	2-6-0	0-4-4	3-5-0	0-5-3	0-2-6
A2R	3-5-0	1-7-0	0-5-3	1-7-0	3-5-0	0-5-3	3-5-0	1-7-0	0-2-6
A3C	4-4-0	2-6-0	0-0-8	2-6-0	4-4-0	1-4-3	4-4-0	1-4-3	1-2-5
A3L	3-5-0	3-5-0	0-0-8	3-5-0	4-4-0	1-1-6	3-5-0	2-5-1	1-2-5
A3R	3-5-0	3-5-0	0-0-8	3-5-0	**5-3-0**	1-2-5	2-5-1	1-6-1	1-3-4
C1C	5-3-0	3-3-2	0-3-5	4-4-0	5-3-0	0-4-4	4-4-0	0-3-5	0-3-5
C1L	**6-2-0**	4-3-1	0-0-8	5-3-0	5-3-0	1-1-6	1-1-6	3-2-3	3-1-4
C1R	4-3-1	6-2-0	0-0-8	3-3-2	**7-1-0**	1-1-6	1-2-5	3-3-2	2-3-3
C2C	3-3-2	3-5-0	0-3-5	1-5-2	5-3-0	0-3-5	**6-2-0**	3-4-1	0-2-6
C2L	**5-3-0**	2-5-1	3-4-1	3-5-0	3-5-0	1-2-5	3-5-0	0-2-6	0-1-7
C2R	2-4-2	3-5-0	0-4-4	0-6-2	6-2-0	0-5-3	6-2-0	0-6-2	0-4-4
C3C	3-5-0	4-4-0	0-1-7	3-4-1	5-3-0	1-2-5	4-4-0	1-4-3	0-3-5
C3L	4-4-0	4-4-0	0-1-7	4-4-0	**5-3-0**	0-2-6	1-1-6	3-4-1	3-1-4
C3R	2-4-2	6-2-0	0-0-8	2-4-2	**7-1-0**	1-1-6	1-4-3	3-4-1	2-4-2
P1C	**5-3-0**	3-4-1	0-3-5	4-4-0	4-4-0	1-3-4	4-4-0	0-2-6	0-3-5
P1L	4-4-0	2-6-0	1-3-4	4-4-0	4-4-0	1-3-4	4-4-0	0-1-7	0-3-5
P1R	**6-2-0**	4-3-1	0-2-6	4-4-0	4-4-0	1-2-5	4-3-1	0-3-5	0-3-5
P2C	3-4-1	3-4-1	0-3-5	0-7-1	3-4-1	3-4-1	**8-0-0**	0-3-5	0-3-5
P2L	4-4-0	2-6-0	2-2-4	4-4-0	4-4-0	1-3-4	4-4-0	0-1-7	0-2-6
P2R	4-4-0	2-6-0	2-2-4	4-4-0	4-4-0	1-3-4	4-4-0	0-1-7	0-2-6
P3C	**5-3-0**	3-4-1	0-3-5	4-4-0	4-4-0	0-4-4	4-4-0	0-3-5	0-3-5
P3L	4-4-0	3-5-0	0-3-5	4-4-0	4-4-0	1-3-4	4-4-0	0-2-6	0-3-5
P3R	**7-1-0**	4-3-1	0-2-6	4-3-1	4-4-0	1-2-5	4-3-1	0-3-5	0-3-5
Mean wins	4.15	3	0.33	3.15	**4.3**	0.78	3.7	0.89	0.56
Mean draws	3.52	4.48	2	4.41	3.63	2.89	3.19	3.52	2.67
Mean losses	0.33	0.52	5.67	0.44	**0.07**	4.33	1.11	3.59	4.78

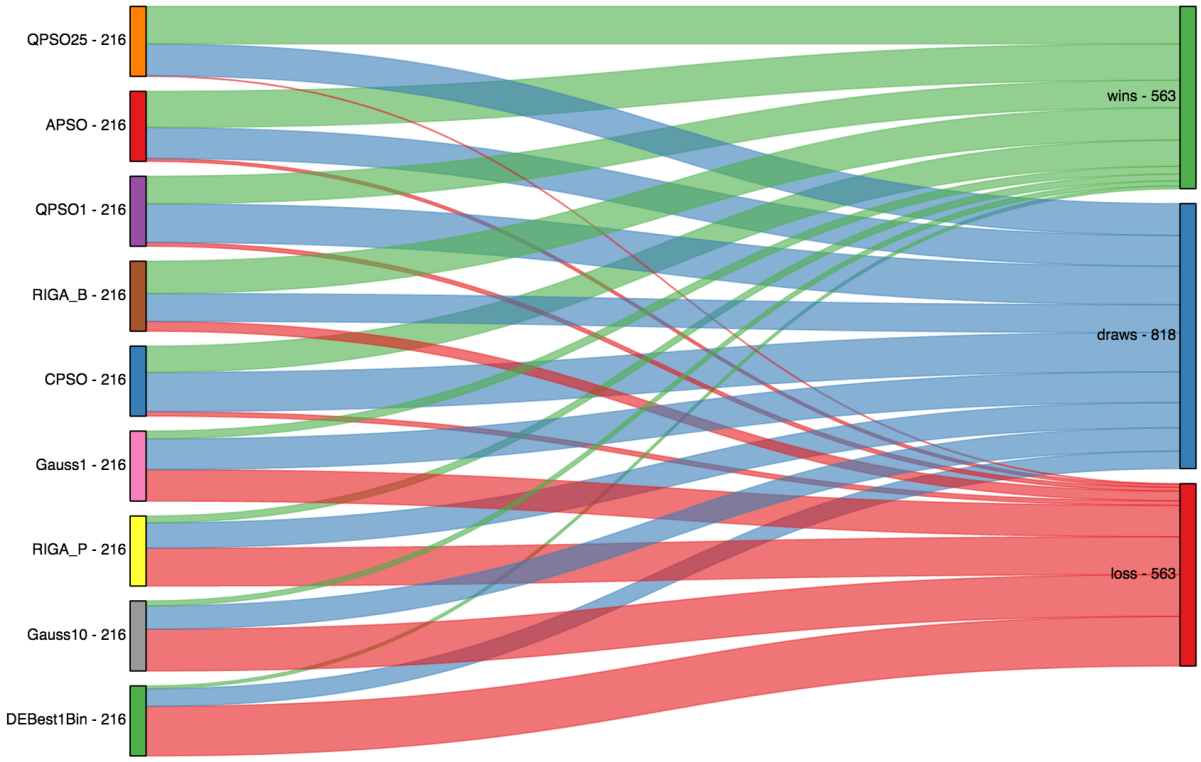


Figure 4.12: Sankey diagram [156] of the proportions of wins, draws, and losses for each standalone heuristic over all 27 environments resulting from all Friedman tests and all associated Shaffer post hoc tests (at $\alpha = 0.05$). The thickness of flows represent the number of hypothesis in the post hoc test where a heuristic either won, drew, or lost against another heuristic.

tic converged quickly and subsequently had no ability to diversify the population of entities.

- **APSO** had fewer charged particles than **CPSO**, but the two methods were identical otherwise. Section 4.3.2 expressed the hope that the higher concentration of charged particles in **CPSO** would result in higher diversity. The $ADSC$ values in figures 4.9, 4.10, and 4.11 confirm that **CPSO** generally maintained higher average levels of diversity in each environment than **APSO**, as would have been expected if more charged particles ended up repelling each other more often.

  APSO showed slightly better performance than **CPSO** in many of the 27 environments in figures 4.6, 4.7, and 4.8. Tables 4.3 and 4.4 show that **APSO** beat

CPSO in all progressive environments, and that **CPSO** performance was similar to or better than **APSO** in a number of abrupt and chaotic environments. This is consistent with the findings of [12][14] as discussed in section 4.3.2. The ranks in table 4.3 corroborate the visual plots by showing that **APSO** had better ranks for most (but not all) environment types.

- **QPSO1** and **QPSO25** had remarkably similar P_r values in almost every environment. Exceptions were C1R and C3R, where **QPSO25** performed much better. Generally, **QPSO25** had the same or tighter P_r distributions (with lower values) than **QPSO1** in all abrupt and chaotic environments. This trend was reversed in the majority of progressive environments. The ranks in table 4.3 corroborate the observations.

  Figures 4.9, 4.10, and 4.11 indicate that **QPSO25** had much greater average diversity levels than **QPSO1** in every environment. Note that the *ADSC* values of **QPSO25** were comparable to the *ADSC* values of **CPSO** in all environments. This shows that the Q, R_p, and R_c parameter values that were set for **CPSO** yielded diversity that was roughly the same as **QPSO25**, as hoped for in section 4.3.2.

- The Gaussian mutation heuristics performed noticeably worse than the meta-heuristic-based algorithms in most environments. Tables 4.3 and 4.4 show that **Gauss1** and **Gauss10** generally had the worst mean ranks and lowest mean wins across all environments (apart from **DEBest1Bin**). Both **Gauss1** and **Gauss10** performed relatively well in chaotic environments. A noteworthy example was C3R, where **Gauss1** was ranked well compared to the other heuristics.

  The *ADSC* value distributions of **Gauss1** and **Gauss10** were markedly different: **Gauss1** maintained a highly diverse population with a tight diversity spread, while **Gauss10** showed a wide range of diversity. **Gauss1** consistently had the highest diversity of any heuristic in every environment, which was probably due to entities being initialized uniformly across the entire search space (yielding high *ADSC* values) that subsequently only searched their local neighborhoods without moving much in the search space (yielding a tight spread in *ADSC* values).

- Figure 4.12 corroborates the mean wins, mean draws, and mean losses presented in table 4.4. **QPSO25** had the highest number of wins and the least losses of any of the stand-alone heuristics. **DEBest1Bin**, on the contrary, had the lowest wins and draws, and the largest losses score against other heuristics. The high number of wins versus draws indicate that the collection of heuristics was relatively well-balanced, with many environments where certain heuristics readily excelled over others.

The results confirm that the collection of heuristics show a wide variety of different diversity management strategies and different performance characteristics across different types of environments. The results serve as a baseline against which the performance of any hyper-heuristic can subsequently be compared.

4.4.4 Research Question 3

What is the performance of each speciated heuristic?

This research question performs the same analysis as used for research question 2 in section 4.4.3 above using the nine *speciated* versions of each heuristic and the two fixed heuristic allocation configurations, as explained in section 4.3.6. The corresponding results are presented in figures 4.13 through 4.19 and tables 4.5 and 4.6.

The following noteworthy observations pertaining to the current research question are drawn from the figures and tables listed above:

- Table 4.5 confirms that the speciated heuristics showed significant differences in performance. No single algorithm was consistently the top performer across all 27 environments, as indicated by the alternating wins, draws, and losses in table 4.6 and the ranks in table 4.5.

- The speciated heuristics showed more uniformity in diversity across environments, in contrast to the wide spread of $ADSC$ values across the stand-alone heuristics in figures 4.9, 4.10, and 4.11. The speciated heuristics showed a higher degree of overlapping $ADSC$ value ranges than the stand-alone heuristics.

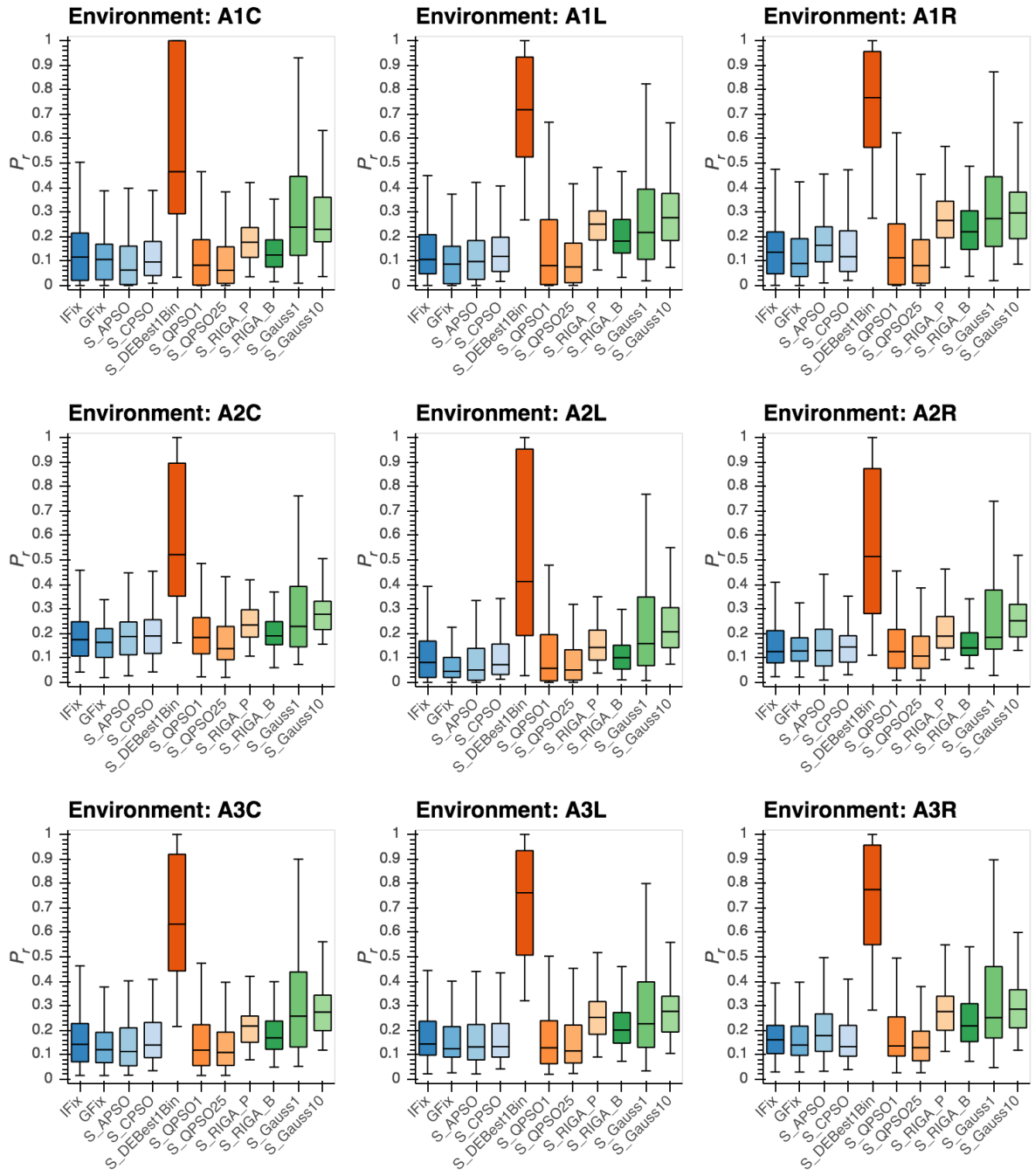


Figure 4.13: P_r values achieved by the speciated heuristics in all abrupt environments over 71 repeat runs.

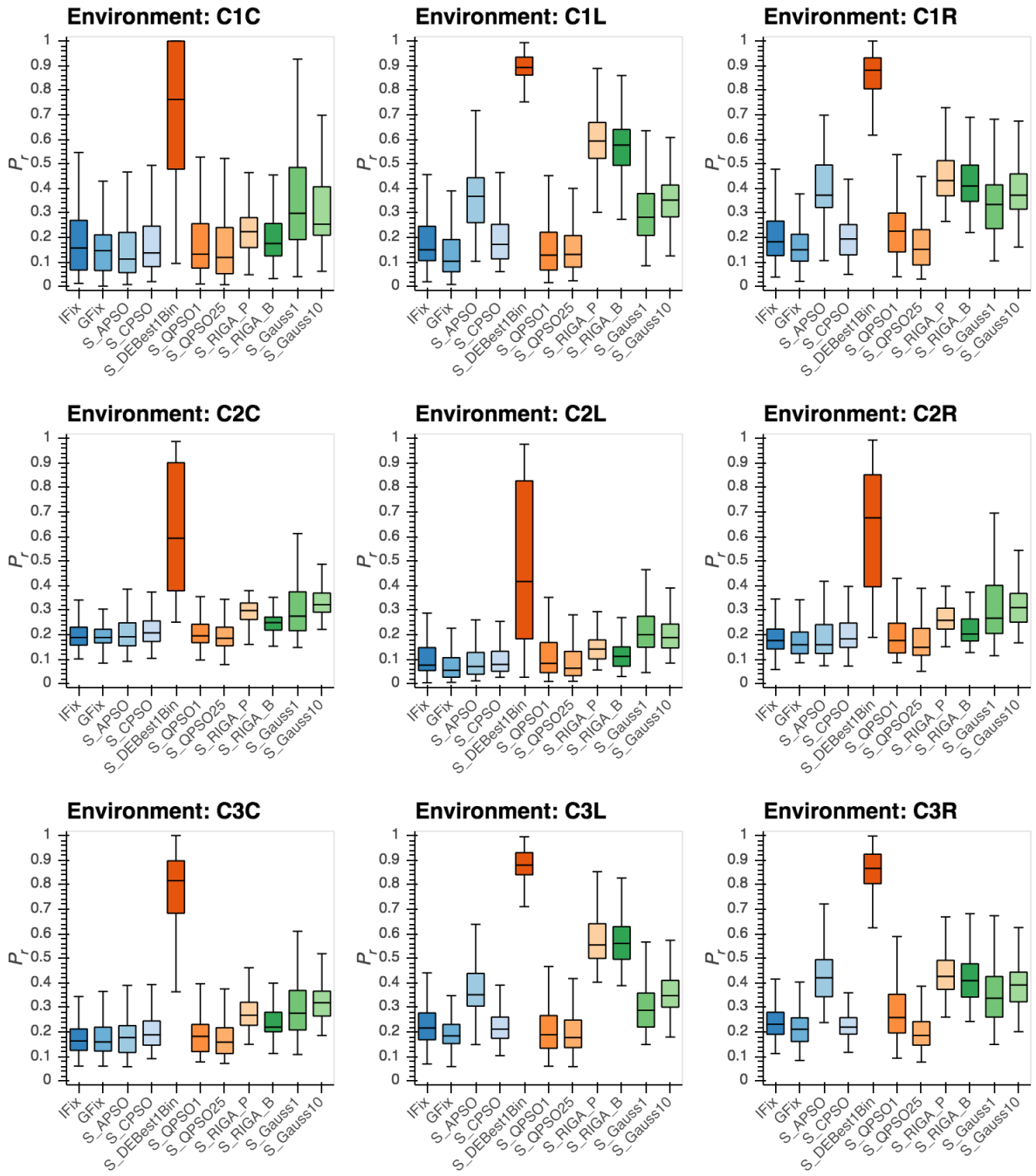


Figure 4.14: P_r values achieved by the speciated heuristics in all chaotic environments over 71 repeat runs.

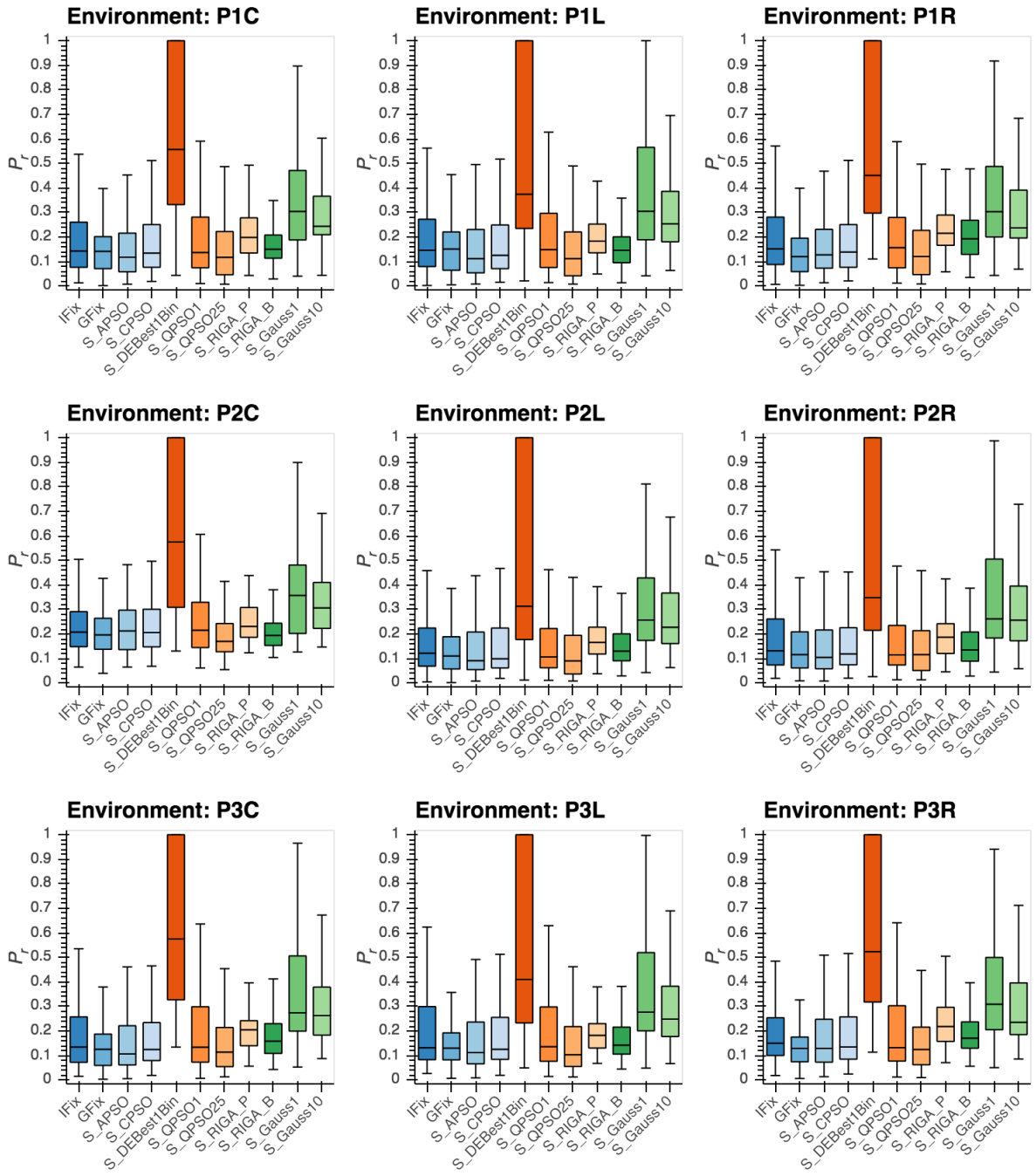


Figure 4.15: P_r values achieved by speciated heuristics in all progressive environments over 71 repeat runs.

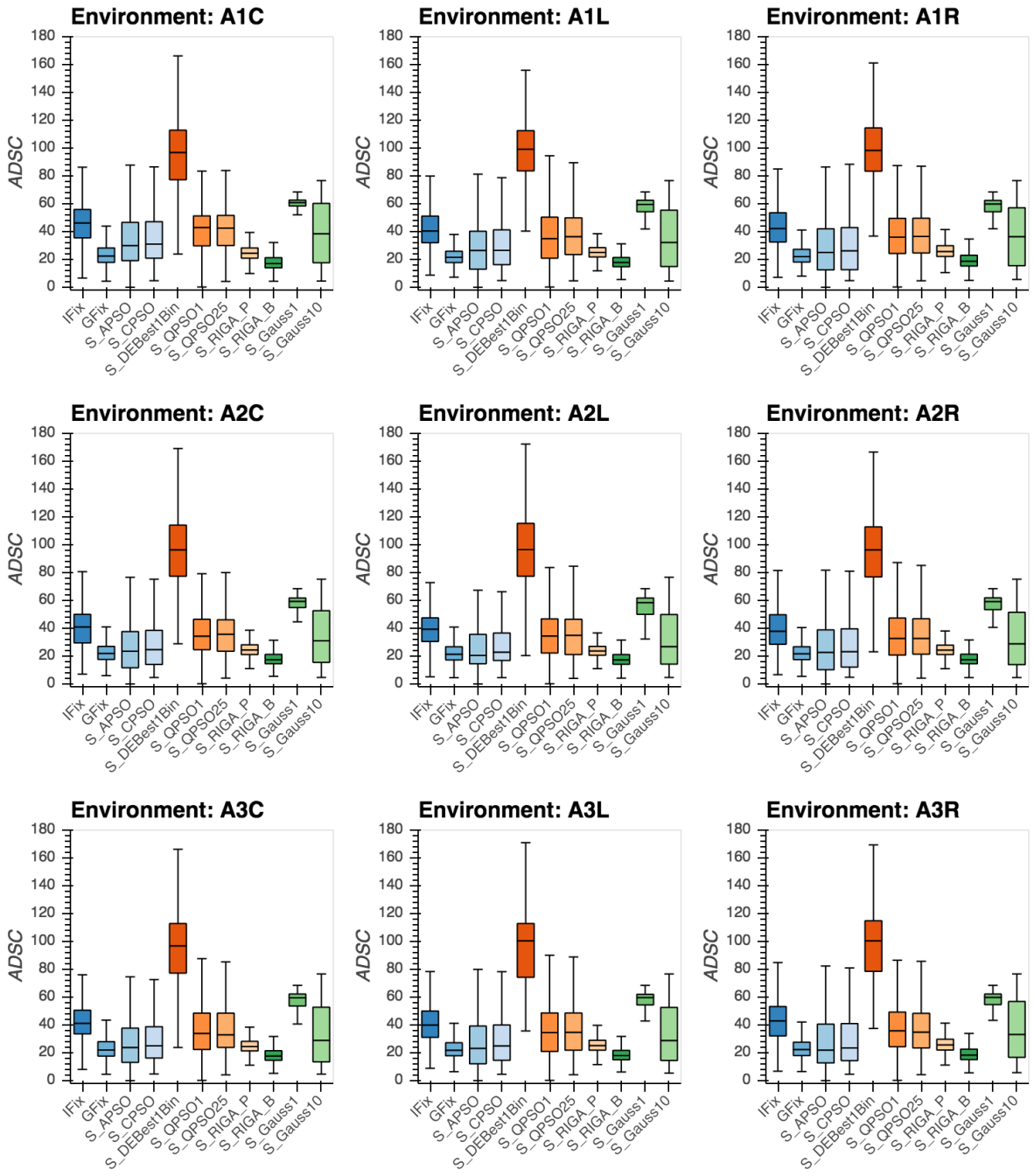


Figure 4.16: *ADSC* values of speciated heuristics in each abrupt environment.

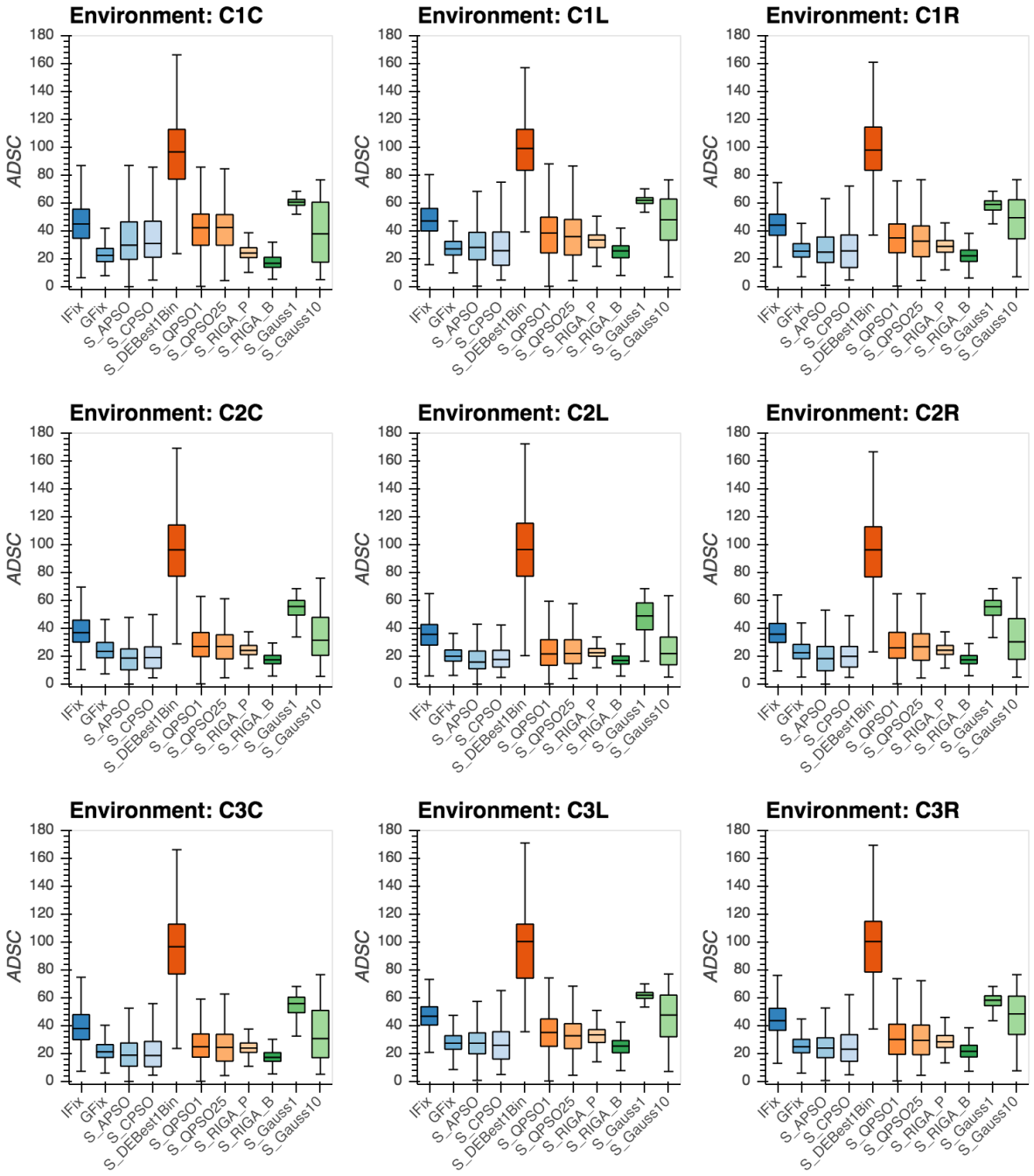


Figure 4.17: *ADSC* values of speciated heuristics in each chaotic environment.

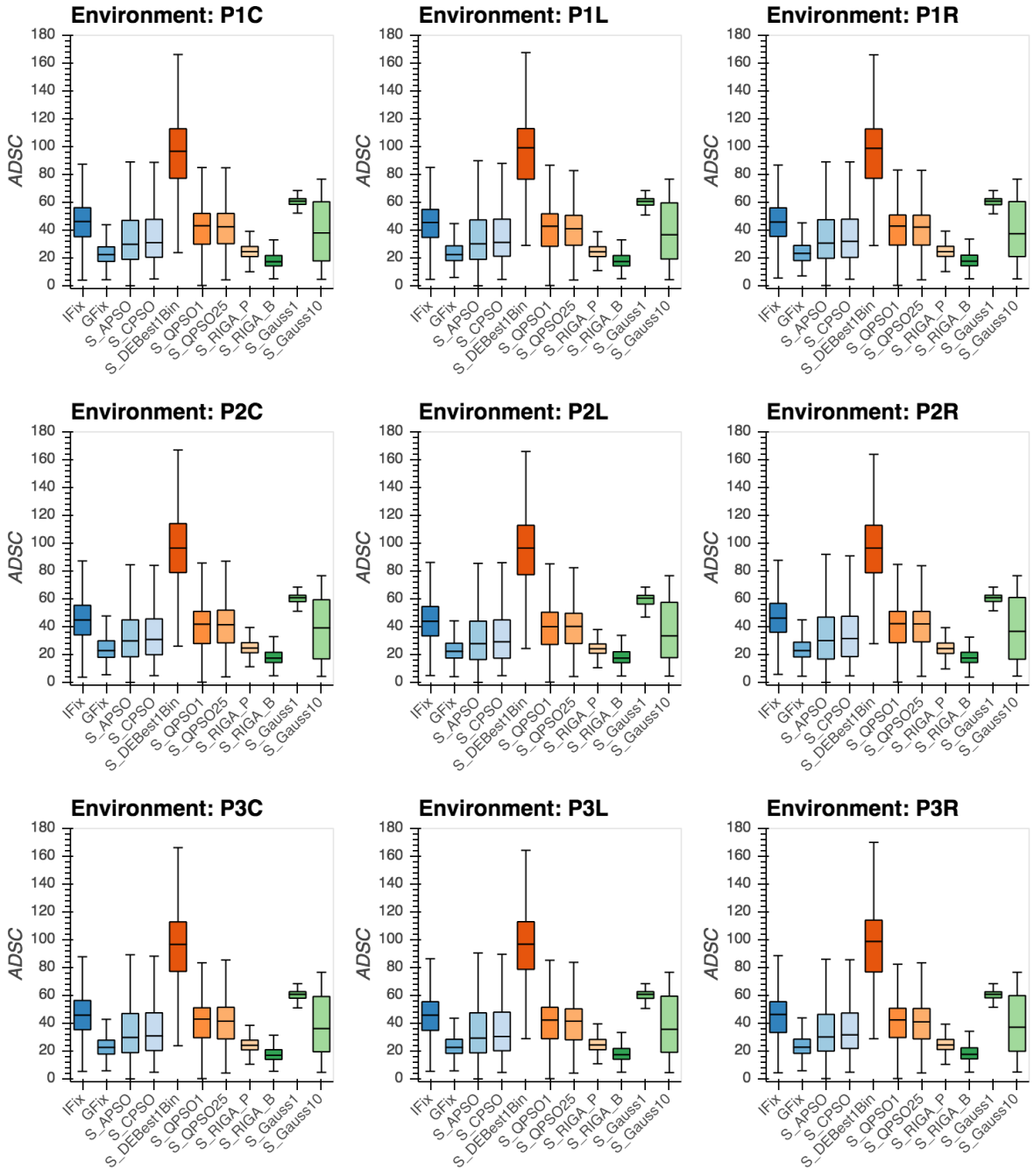


Figure 4.18: *ADSC* values of speciated heuristics in each progressive environment.

Table 4.5: Average ranks per environment for each speciated heuristic using the Friedman test, as well as the resulting p-value. Bold values indicate the best performers.

Environment	IFix	GFix	S_APSO	S_CPSO	S_DEBest1Bin	S_QPSO1	S_QPSO25	S_RIGA_P	S_RIGA_B	S_Gauss1	S_Gauss10	p values
A1C	4.76	3.34	**2.87**	5.52	10.70	3.72	4.23	7.49	5.85	8.45	9.07	1.11e-16
A1L	4.17	**2.80**	3.90	5.27	10.98	3.66	3.72	8.10	6.58	7.75	9.07	1.11e-16
A1R	3.89	**2.65**	5.61	4.76	10.99	3.21	3.30	8.27	6.69	7.81	8.83	1.11e-16
A2C	4.44	3.93	4.30	5.42	10.83	4.56	**3.85**	7.49	5.38	7.15	8.65	1.11e-16
A2L	4.58	3.34	**3.25**	5.46	10.36	3.83	3.87	7.58	6.10	8.35	9.28	1.11e-16
A2R	4.46	4.13	**3.86**	5.44	10.44	4.28	3.96	7.45	5.44	7.29	9.25	1.11e-16
A3C	4.08	3.96	**3.54**	5.45	10.91	3.93	3.90	7.45	6.03	7.91	8.85	1.11e-16
A3L	4.24	**3.46**	4.14	5.14	10.98	3.77	3.72	8.10	6.49	7.19	8.76	1.11e-16
A3R	3.66	3.55	5.58	4.32	10.99	3.75	**3.35**	8.08	6.75	7.42	8.55	1.11e-16
C1C	4.62	**2.94**	3.17	5.01	10.91	4.89	3.65	7.34	5.86	8.84	8.77	1.11e-16
C1L	3.54	**2.04**	6.94	4.08	10.99	2.83	3.07	9.41	9.23	6.61	7.27	1.11e-16
C1R	3.63	2.38	7.72	3.73	11.00	4.20	**2.30**	8.70	8.08	6.70	7.55	1.11e-16
C2C	3.55	**3.48**	3.66	4.70	10.75	4.42	3.54	8.32	6.34	7.72	9.52	1.11e-16
C2L	4.51	**2.28**	3.73	5.32	10.07	5.15	3.42	7.30	5.66	9.24	9.31	1.11e-16
C2R	3.93	3.37	3.80	5.10	10.72	4.61	**3.34**	8.01	5.76	8.13	9.24	1.11e-16
C3C	3.62	3.52	3.58	5.24	11.00	4.37	**3.30**	7.86	6.21	7.97	9.34	1.11e-16
C3L	3.59	2.69	7.01	3.68	10.99	3.24	**2.45**	9.41	9.37	6.37	7.21	1.11e-16
C3R	3.73	2.48	8.25	3.39	11.00	4.69	**2.13**	8.83	7.83	6.46	7.20	1.11e-16
P1C	4.86	3.31	**3.30**	4.99	10.70	5.34	3.46	6.86	5.25	9.15	8.77	1.11e-16
P1L	4.90	3.52	3.51	5.13	9.94	5.51	**3.34**	6.92	5.10	9.27	8.86	1.11e-16
P1R	4.86	**2.75**	3.80	4.70	10.75	4.72	3.36	7.38	6.00	8.96	8.72	1.11e-16
P2C	4.68	3.96	4.24	5.28	10.50	5.49	**3.32**	6.66	4.52	8.60	8.75	1.11e-16
P2L	4.92	3.63	3.51	5.01	9.92	5.02	**3.35**	6.87	5.56	9.34	8.87	1.11e-16
P2R	4.93	3.41	**3.25**	5.25	10.11	5.23	3.44	7.10	5.14	9.07	9.07	1.11e-16
P3C	4.86	**3.04**	3.34	5.07	10.79	5.10	3.56	7.03	5.42	9.05	8.75	1.11e-16
P3L	5.01	3.30	**3.28**	5.07	10.02	5.24	3.32	7.01	5.54	9.18	9.03	1.11e-16
P3R	5.04	**2.85**	3.51	4.68	10.70	4.74	3.50	7.46	5.87	9.01	8.65	1.11e-16
Mean rank	4.34	**3.19**	4.32	4.90	10.67	4.43	3.40	7.72	6.22	8.11	8.71	–

Table 4.6: Wins, draws and losses per environment for each speciated heuristic. The notation *W-D-L* indicates the number of wins, draws, and losses for each heuristic in each environment. Bold values indicate the best performers.

Environment	IFix	GFix	S_APSO	S_CPSO	S_DEBest1Bin	S_QPSO1	S_QPSO25	S_RIGA_P	S_RIGA_B	S_Gauss1	S_Gauss10
A1C	4-5-1	6-4-0	**7-3-0**	4-3-3	0-1-9	6-4-0	4-6-0	1-3-6	3-4-3	1-2-7	0-3-7
A1L	5-5-0	**6-4-0**	5-5-0	4-5-1	0-0-10	5-5-0	5-5-0	1-3-6	2-3-5	1-3-6	1-2-7
A1R	6-4-0	**7-3-0**	4-2-4	5-4-1	0-0-10	6-4-0	6-4-0	1-3-6	2-3-5	1-3-6	1-2-7
A2C	4-6-0	4-6-0	4-6-0	4-6-0	0-0-10	4-6-0	4-6-0	1-2-7	4-6-0	1-2-7	1-2-7
A2L	4-6-0	6-4-0	6-4-0	4-4-2	0-1-9	5-5-0	5-5-0	2-2-6	3-3-4	1-2-7	0-2-8
A2R	4-6-0	4-6-0	4-6-0	4-6-0	0-1-9	4-6-0	4-6-0	2-1-7	4-6-0	2-1-7	0-1-9
A3C	5-5-0	5-5-0	6-4-0	4-5-1	0-0-10	5-5-0	5-5-0	1-3-6	3-2-5	1-2-7	1-2-7
A3L	5-5-0	5-5-0	5-5-0	4-6-0	0-0-10	5-5-0	5-5-0	1-3-6	2-3-5	1-3-6	1-2-7
A3R	6-4-0	6-4-0	4-2-4	5-5-0	0-0-10	6-4-0	6-4-0	1-3-6	2-3-5	1-3-6	1-2-7
C1C	4-5-1	**8-2-0**	7-3-0	4-4-2	0-0-10	4-4-2	5-5-0	1-3-6	3-4-3	1-2-7	1-2-7
C1L	6-4-0	**7-3-0**	3-2-5	6-3-1	0-1-9	6-4-0	6-4-0	0-2-8	1-1-8	3-2-5	3-2-5
C1R	6-4-0	7-3-0	1-4-5	6-4-0	0-0-10	6-2-2	7-3-0	1-3-6	1-4-5	2-3-5	1-4-5
C2C	5-5-0	5-5-0	5-5-0	4-6-0	0-1-9	5-5-0	5-5-0	1-2-7	3-2-5	2-2-6	0-2-8
C2L	4-5-1	**8-2-0**	5-5-0	4-4-2	0-2-8	4-4-2	7-3-0	3-1-6	3-4-3	0-2-8	0-2-8
C2R	5-5-0	6-4-0	5-5-0	4-4-2	0-1-9	4-6-0	6-4-0	1-2-7	4-2-4	1-2-7	0-3-7
C3C	5-5-0	6-4-0	5-5-0	4-4-2	0-1-9	5-5-0	6-4-0	1-3-6	3-2-5	1-2-7	0-3-7
C3L	6-4-0	6-4-0	3-2-5	6-4-0	0-2-8	6-4-0	6-4-0	0-2-8	0-2-8	3-2-5	3-2-5
C3R	6-4-0	7-3-0	1-3-6	6-4-0	0-0-10	6-2-2	7-3-0	1-3-6	1-4-5	3-2-5	1-4-5
P1C	4-6-0	7-3-0	7-3-0	4-4-2	0-1-9	3-4-3	6-4-0	3-2-5	3-4-3	0-2-8	1-1-8
P1L	4-6-0	5-5-0	5-5-0	4-5-1	0-2-8	3-4-3	**7-3-0**	3-1-6	4-5-1	0-2-8	0-2-8
P1R	4-5-1	**8-2-0**	5-5-0	4-5-1	0-0-10	4-5-1	5-5-0	1-3-6	3-4-3	1-2-7	1-2-7
P2C	4-6-0	4-6-0	4-6-0	3-6-1	0-0-10	3-6-1	**6-4-0**	3-2-5	4-6-0	1-1-8	1-1-8
P2L	4-6-0	5-5-0	5-5-0	4-6-0	0-2-8	4-6-0	5-5-0	3-1-6	3-4-3	0-2-8	0-2-8
P2R	4-5-1	7-3-0	**8-2-0**	4-3-3	0-2-8	4-3-3	7-3-0	3-0-7	4-3-3	0-2-8	0-2-8
P3C	4-5-1	**8-2-0**	7-3-0	4-4-2	0-0-10	4-4-2	5-5-0	3-1-6	3-4-3	1-1-8	1-1-8
P3L	4-3-3	8-2-0	8-2-0	4-3-3	0-2-8	4-3-3	8-2-0	3-1-6	3-4-3	0-2-8	0-2-8
P3R	4-5-1	**8-2-0**	5-5-0	4-5-1	0-0-10	4-5-1	5-5-0	1-3-6	3-4-3	1-2-7	1-2-7
Mean wins	4.67	**6.26**	4.96	4.33	0	4.63	5.67	1.59	2.74	1.11	0.74
Mean draws	4.96	3.74	3.96	4.52	0.74	4.44	4.33	2.15	3.56	2.07	2.11
Mean losses	0.37	**0**	1.07	1.15	9.26	0.93	**0**	6.26	3.7	6.81	7.15

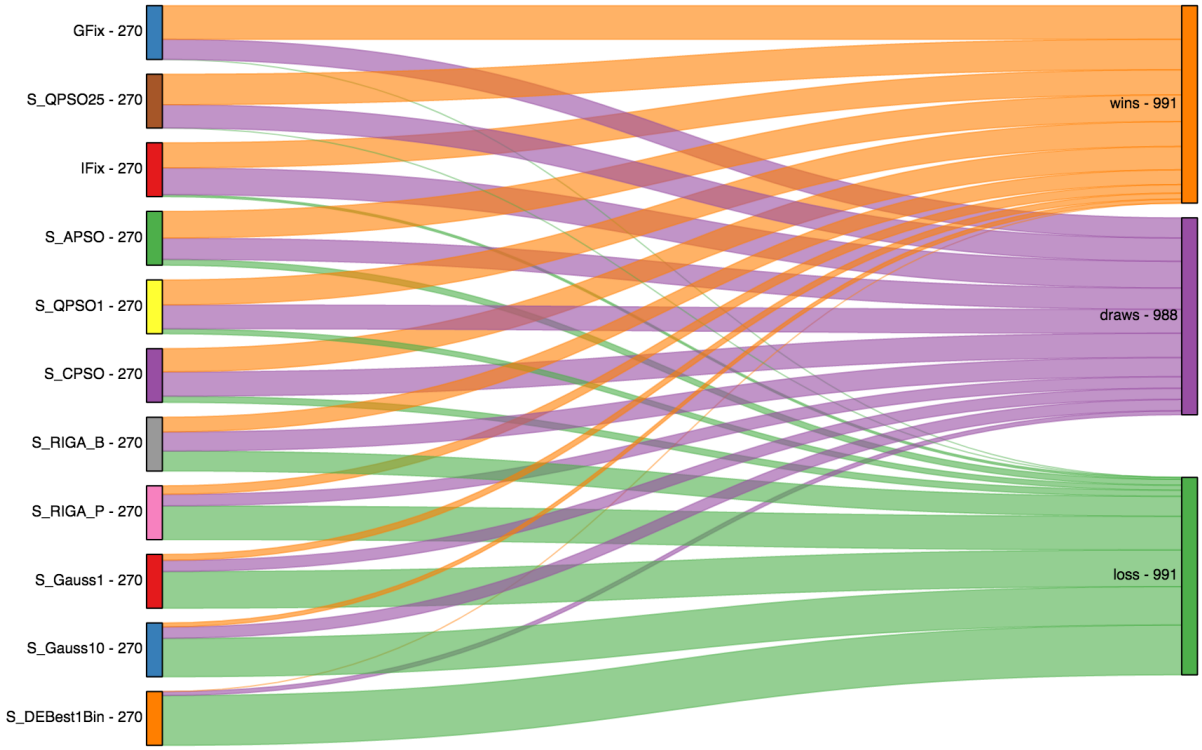


Figure 4.19: Sankey diagram [156] of the wins, draws and losses for each speciated heuristic over all 27 environments resulting from all Friedman tests with all associated Shaffer post hoc tests (at $\alpha = 0.05$). The thickness of flows represent the number of hypothesis in the post hoc test where a heuristic either won, drew, or lost against another heuristic.

- Similar to the results for the stand-alone heuristics, the speciated version of DE, namely **S_DEBest1Bin**, performed distinctly worse than the other heuristics. This pattern was noticeable in all 27 environments in figures 4.13, 4.14, and 4.15.

  A notable difference between **DEBest1Bin** and **S_DEBest1Bin** was the *ADSC* values of each heuristic. **DEBest1Bin** converged and maintained very low *ADSC* values in each of the 27 environments. In contrast, **S_DEBest1Bin** showed the highest diversity values of all the speciated methods, as can be seen in figures 4.16, 4.17, and 4.18. Closer inspection reveals that all entities in each **S_DEBest1Bin** sub-population converged to yield low *ADSC* values. The majority of *ADSC* values across all iterations for all algorithm runs were effectively zero for each environment, indicating that convergence occurred for that sub-population. Ev-

ery sub-population of **S_DEBest1Bin** showed exactly the same pattern across all types of environments. The high overall diversity of **S_DEBest1Bin** in figures 4.16, 4.17, and 4.18 suggests that each sub-population converged to separate points in the search space, yielding a high overall diversity for the **S_DEBest1Bin** population as a whole. Each DE sub-population was, however, unable to adapt to changing environments, as the poor P_r performance across all environments shows.

- **S_APSO** and **S_CPSO** showed almost identical diversity values in figures 4.16, 4.17, and 4.18, where **S_CPSO** generally had slightly elevated lower bounds than **S_APSO**. This is in stark contrast with **CPSO** that had noticeably larger diversity values than **APSO** in figures 4.9, 4.10, and 4.11. The use of speciation substantially increased the diversity of both **S_APSO** and **S_CPSO** compared to their stand-alone counterparts. Similar to **APSO** and **CPSO**, the performance values of **S_APSO** and **S_CPSO** varied across environments, as can be seen in table 4.5, and no single method dominated the other consistently across all 27 environments.

- Both **S_QPSO1** and **S_QPSO25** had similar P_r values in almost every environment, similar to their stand-alone counterparts **QPSO1** and **QPSO25**. Table 4.5 shows that **S_QPSO25** had a better overall mean rank than **S_QPSO1**. Figures 4.16, 4.17, and 4.18 show that **S_QPSO1** and **S_QPSO25** exhibited almost identical $ADSC$ behavior for each of the 27 environments, in contrast to their stand-alone counterparts that showed distinctly different $ADSC$ behavior between **QPSO1** and **QPSO25**.

- Figures 4.13, 4.14, and 4.15 show that **S_Gauss1** and **S_Gauss10** performed noticeably worse than the other heuristics (excluding **S_DEBest1Bin**) in most environments (the exceptions were C1L, C1R, C3L, and C3R). The ranks in table 4.5 corroborate the visual findings, where **S_Gauss1** and **S_Gauss10** were ranked the worst (after **S_DEBest1Bin**). **S_Gauss10** always had a much tighter spread of P_r values than **S_Gauss1** in every environment. **S_Gauss1** showed a higher and much tighter spread of $ADSC$ values than **S_Gauss10** in each environment, which was consistent with the stand-alone versions of the respective heuristics.

- **IFix** and **GFix** showed competitive performance across all 27 environments in

figures 4.13, 4.14, and 4.15, where **GFix** showed a tighter distribution of P_r values that was visually the same as or better than **IFix** in almost every environment. Table 4.5 confirms this visual finding where it can be observed that **GFix** always had a better Friedman rank than **IFix** in every environment. **GFix** had the best rank of all speciated heuristics overall. Table 4.6 shows that **GFix** had the same or more wins than **IFix** in every environment. **GFix** also showed noticeably lower *ADSC* values than **IFix** in each environment, as illustrated in figures 4.16, 4.17, and 4.18.

- Figure 4.19 illustrates the mean wins, mean draws, and mean losses shown in table 4.6. **GFix** had the highest number of wins and draws with no losses followed closely by **S_QPSO25**. **S_DEBest1Bin** was the worst performer. The overall order of the speciated heuristics was roughly similar to the stand-alone heuristics in figure 4.12, except for **S_RIGA_B** and **S_Gauss1** that both fell lower in the standings than their **RIGA_B** and **Gauss1** counterparts, respectively.

The results in this section confirm that HMHH variations utilizing homogeneous speciation (where multiple sub-populations are independently managed by the same heuristic) show a range of different performance and diversity characteristics. The **IFix** and **GFix** versions of HMHH configurations using heterogeneous speciation (where sub-populations are managed by different heuristics) show competitive behavior.

The stand-alone heuristics, the speciated heuristics, and the fixed allocation HMHH variations (**IFix** and **GFix**) form control groups that any intelligent HMHH selection operators need to outperform.

4.5 Summary

It is important that any heuristic pool consists of a varied collection of different types of heuristics. A key goal of a hyper-heuristic is to perform better than any of the heuristics in isolation. The experimental approach and control groups laid out in this chapter are used to conduct all experimentation in this thesis.

Control groups were proposed in section 4.3 that will be used in chapter 6 to objectively measure if any increase in performance by any hyper-heuristic is due to in-

telligent heuristic selection, or due to chance. Specifically, control groups were defined to determine if a hyper-heuristic performs better than any of the underlying individual heuristics, if any increased performance is due to the population of entities being split up into multiple sub-populations (speciation), and if intelligent heuristic allocation by any hyper-heuristic indeed performs better than simple random allocation.

A new performance measure, called the *relative error distance*, P_r, was proposed in section 4.2 that considers the sustained performance of any DOP-focused algorithm in a more holistic way. The P_r measure jointly compares the normalized performance of an algorithm in each search landscape over all change periods against the best possible performance score.

The performance and diversity of the stand-alone and speciated versions of each heuristic were systematically explored in section 4.4. The P_r and $ADSC$ values of the stand-alone and speciated versions of each heuristic were computed and analyzed. Rigorous nonparametric statistical tests were used to demonstrate how the performance of all algorithms in each environment differed significantly. It was shown that various heuristics excelled in different types of environments, which justifies each of their inclusion into the heuristic pool. The results of this chapter serve as a baseline against which the performance of any hyper-heuristic can subsequently be compared to in later chapters.

The next chapter investigates the effect that different combinations of heuristic change triggers and HMHH topologies have on the performance of any given hyper-heuristic.

Chapter 5

Performance Sensitivity of HMHH Under Varying Parameter Values

"Don't mistake activity with achievement."
– John Wooden

This chapter explores the effect that various mechanisms to trigger heuristic change combined with different neighborhood topologies have on the performance of selection operators in the HMHH framework. The frequency parameter of each trigger is systematically varied to ascertain the effect, if any, that each trigger has on performance. This process is repeated for 27 different types of DOPs across a wide variety of hyper-heuristics, as detailed in section 5.2.2. The entire procedure is repeated across two HMHH neighborhood topologies and contrasted against each other.

The term *hyper-heuristic* is used interchangeably with "HMHH selection operator" in this chapter, and the term *heuristic* refers to a meta-heuristic algorithm that has been configured with specific parameter values.

5.1 Introduction

The aim of a parameter sensitivity analysis is to determine how strong the effect of a parameter is on the outcomes of a system. The analysis is conducted by systematically varying a parameter's values in a controlled manner, and determining if the change in the

parameter's value yields a statistically significantly different outcome in performance. In this thesis the HMHH framework was extended with alternative heuristic change triggering mechanisms that each have a frequency parameter, namely k. Two different heuristic neighborhood topologies were also motivated. Different combinations of the three trigger mechanisms with different values for the parameter k, together with the two neighborhood topologies, may significantly alter the operation of any given hyper-heuristic. This chapter systematically explores the performance sensitivity of various hyper-heuristics with respect to the proposed trigger and neighborhood topology extensions.

Different heuristic change triggering mechanisms for HMHH were proposed in section 3.4.4, namely a *periodic* trigger as presented in the original HMHH algorithm, a *stagnation* trigger that signals a heuristic change for an entity if the entity's fitness does not steadily improve over a specific period, and a *random trigger* that probabilistically signals a heuristic change for an entity. Each type of trigger takes a frequency parameter, k, that defines the number of algorithm iterations between heuristic changes. Larger values of k increase the number of algorithm iterations before the hyper-heuristic is invoked to perform heuristic selection for the triggered entity.

Section 3.4.2 extended HMHH with alternative neighborhood topologies. Various topologies stipulate different ways in which entities are visible across heuristics. The *global* topology, the topology used in the original HMHH algorithm, allows each heuristic to view the information of all entities across the entire HMHH parent population, but each heuristic may only modify those entities that are assigned to the heuristic. The newly proposed *island* topology allows each heuristic to view and modify only those entities that are assigned to the heuristic.

Section 5.2 outlines the experimental procedure in detail, including an overview of the experimental method, the benchmark problem settings, and the parameterization strategy used to define heuristic and hyper-heuristic algorithm instances. Section 5.3 shows the results of experimentation as a series of three distinct research questions that, respectively, investigates the sensitivity of each type of trigger to different values for k, examines the impact that different types of triggering mechanisms have on the hyper-heuristics, and directly compares the performance of the *global* and *island* topologies when using all variations of triggers and values for k. Section 5.4 concludes the chapter.

5.2 Experimental Procedure

The experiments in this chapter only compare different versions of each hyper-heuristic against themselves. At no point is the effectiveness of one hyper-heuristic directly compared against the effectiveness of any other hyper-heuristic (this is left for chapter 6). Instead, a range of different hyper-heuristics are set up with different trigger and neighborhood topology configurations. The analysis is restricted to determining the following:

1. Whether the parameter k (i.e. the frequency with which heuristic selection is triggered) has a strong influence on the performance of each trigger. Are there any identified trends in performance sensitivity that are sustained across different hyper-heuristics, neighborhood topologies, and DOP types?

2. Whether different trigger choices result in noticeably different performance across hyper-heuristics, and whether different hyper-heuristics have strong tendencies to perform better with certain types of triggers. All hyper-heuristics share the same pool of heuristics, because testing the effectiveness of each trigger under varying heuristic pool compositions is out of scope for this thesis, and is left for future studies (as explained in section 4.3.5).

3. If either the *island* or *global* topology is always superior to the other, or if there are certain triggers, environments, or hyper-heuristics that each topology excels in.

The overall experimental approach is based on the benchmark problems, heuristic implementation choices, and overall design decisions outlined in section 4.3. The experimental method is reiterated in section 5.2.1, along with any additional design decisions used in this chapter. Section 5.2.2 discusses the applicable parameter values used for all HMHH selection operators, neighborhood topologies, and triggering mechanisms used in experiments.

5.2.1 Experimental Method

The same experimental method outlined in section 4.3 is used for the experimental analysis in this chapter, which is briefly summarized below along with the following minor modifications:

- **Benchmark problems**: The same MPB function generator is used, and the same 71 landscape samples are generated randomly for each of the 27 different DOP types.

- **Performance measures**: The P_r performance measure is used to capture the performance of each algorithm as a scalar value.

- **Control groups**: The control groups outlined in section 4.3 are not used in this chapter. Variations of the same hyper-heuristic are compared to themselves to determine if changes in trigger configurations yield statistically significant differences in performance. The goal is not to compare any performance improvements of any of the configurations against control groups.

- **Support for specific DOP-focused algorithm capabilities**: Section 2.5 discussed the different "building blocks" identified in the literature that modern DOP algorithms should cater for. Section 4.3.1 outlined how the experiment supports the incorporation of the various components. The experiments in this chapter use the same implementation decisions.

- **Heuristic algorithm and parameter value choices**: Section 4.3.2 outlined a selection of representative meta-heuristics from the SI and EC fields that make up a pool of nine heuristics. Suitable parameter choices were motivated for each heuristic. The same algorithms, parameter values, and implementation decisions for each heuristic are used in all experiments in this chapter.

- **Heuristic pool choices:** The same pool of nine heuristics is made available to all hyper-heuristics in every experiment run. The HMHH parent population consists of 50 entities that are assigned to specific heuristics across the pool. Initially, all 50 entities are distributed across all heuristics in the pool in a uniform random manner.

  Each heuristic maintains a minimum number of assigned entities to avoid the situation where a heuristic with zero assigned entities produces no feedback information. A lack of feedback could result in the heuristic never being chosen by certain hyper-heuristics, as discussed in section 3.4.6. Consequently, each heuristic always has a

minimum entity count of one in the *global* topology, and four in the *island* topology. Each heuristic can only view those entities that are assigned to it when the *island* topology is used. Since **DEBest1Bin** forms part of the heuristic pool, and DE requires a minimum of four individuals to be effective (as discussed in section 4.3.2), a minimum of four entities is used in *island* topology configurations.

The experiment design decisions above yield a controlled set of problem instances, a performance measure that allows for an apples-to-apples comparison of performance across algorithms and problem instances, and common algorithm components for use by all hyper-heuristics.

5.2.2 Hyper-heuristic Selection Operator Choices

Section 3.4 presented two HMHH neighborhood topologies, three heuristic change triggering mechanisms, and numerous selection operator algorithms for HMHH. Many selection operators and all of the heuristic change triggering mechanisms require parameter values. The passages below outline in more detail how each component is added to the experiment:

- **Selection operators**: Section 3.4.6 described the algorithmic layout and parameter requirements (if applicable) of each selection operator in detail. Table 5.1 summarizes the HMHH selection operator design and applicable parameter decisions that are used in the experiments.
- **HMHH neighborhood topologies**: Each selection operator is set up to use both the *island* and *global* neighborhood topologies, respectively.
- **Heuristic change triggers**: Each hyper-heuristic is configured to use each of the three heuristic change triggers outlined in section 3.4.4, namely a *periodic* trigger (PT), *stagnation* trigger (ST), and a *random* trigger (RT). The triggering mechanisms are used in combination with each of the selection operators and neighborhood topologies above.
- **Heuristic change frequency (k)**: Each trigger requires a frequency parameter, k, that controls the number of algorithm iterations before triggering a heuristic

Table 5.1: Selection operator algorithms labels and parameters.

Selection Operator	**Label**	**Parameter Notes**
Fixed	**IFix GFix**	Parameterless
Simple Random	**Rand**	Parameterless
Permutation	**Perm**	Parameterless
Roulette Wheel	**RoulM**	Using equation (3.3)
	RoulX	Using equation (3.4)
Heuristic Tournament	**HTour2M**	Minimum tournament size is 2, uses equation (3.3)
	HTour2X	Minimum tournament size is 2, uses equation (3.4)
	HTour5M	Tournament size is 50% of n_h, uses equation (3.3)
	HTour5X	Tournament size is 50% of n_h, uses equation (3.4)
Entity Tournament	**ETour2**	Minimum tournament size is 2
	ETour25	Tournament size is 50% of n_s
Ant-inspired Rank-based	**ARank**	Parameterless
Normalized Ant-inspired Rank-based	**NARank**	Parameterless
Ant-inspired Fitness Proportional	**AProp**	Parameterless
Normalized Ant-inspired Fitness Proportional	**NAProp**	Parameterless
Frequency Improvement ($k = 10$)	**Freq2**	Minimum tournament size is 2
	Freq5	Tournament size is 50% of n_h
Frequency Improvement Reinforcement Learning	**RLFreq**	Parameterless, see rank ranges
Fitness Proportional Reinforcement Learning	**RLProp**	Parameterless, see rank ranges
Difference Proportional	**DProp**	$\beta = 5$
Competitive Population	**Comp**	Parameterless
Softmax	**Soft**	$\tau = 1$

change for an entity. The parameter k is interpreted by each triggering mechanism to yield roughly the same frequency of heuristic changes per trigger for corresponding values of k. The value of k is set to different percentages of the landscape change frequency of every environment type, namely $k \in \{20\%, 40\%, 60\%, 80\%, 100\%\}$. For example, if the length of a change period of an environment such as C3R is 20 iterations (refer to table 2.2), then a choice of $k = 40\%$ results in $20 * 0.4 = 8$ iterations before each triggering event.

The combination of three triggering mechanisms with five parameters values for k results in 15 variations per hyper-heuristic. The same setup is repeated across the two neighborhood topologies to yield a total of 30 variations for each hyper-heuristic. All variations are executed for 1000 algorithm iterations on 71 random (but reproducible) instances of each of the 27 problem types.

5.3 Experimental Results

All variations of each hyper-heuristic were configured and executed as explained in section 5.2 above. The results of all experiments are organized into distinct research questions which are listed in section 5.3.1 and answered in the subsequent sections below.

5.3.1 Research Questions

The following research questions drive the experimental analysis:

1. *Does the parameter k make a significant difference to the performance of each type of trigger?* How sensitive is each type of trigger to different values for k across different hyper-heuristics? Are similar trends in sensitivity to changes in k visible across hyper-heuristics? How does the choice between an *island* or *global* neighborhood topology affect performance sensitivity with respect to k?

2. *Does the type of trigger significantly alter performance of the same hyper-heuristic?* How does the performance of each hyper-heuristic vary when different types of triggers are used? Is any trigger consistently better across different hyper-heuristics? How does the performance of the same combination of hyper-heuristic and trigger differ between the *global* and *island* topology?

3. *When is the global or island neighborhood topology significantly better?* Is either topology consistently (and significantly) superior across different hyper-heuristics, different environment types, and different trigger configurations?

5.3.2 Research Question 1

Does the parameter k make a significant difference to the performance of each type of trigger?

A trigger can be considered sensitive to the value of the parameter k if gradually differentiated values of k (based on a percentage of the period of time between environment changes) yield significantly different results when keeping all other variables constant. Conversely, if various values of k (all other variables being equal) yield statistically insignificant differences in performance, then the trigger type is not sensitive to the parameter k. Different hyper-heuristics might show different behaviors in this regard across different environments and trigger types.

For this research question, each hyper-heuristic is configured to use each of the three heuristic change triggers outlined in section 3.4.4. For each trigger, the values for k were set to $k \in \{20\%, 40\%, 60\%, 80\%, 100\%\}$ of the length of the change period of each environment, as outlined in section 5.2.2. A separate Friedman test was used for each hyper-heuristic and each environment to compare the P_r values of the five k-value configurations. The test was repeated independently for each trigger. The analysis was repeated using both the *island* and *global* topologies. Figure 5.1 illustrates the boundaries of each Friedman test and all associated Shaffer post hoc tests in this research question.

The following outcomes were possible per environment:

1. If a Friedman test did not show a significant difference (i.e. p-values were greater than $\alpha = 0.05$), then all five configurations of k immediately received zero wins and zero losses each, because their performances were statistically indistinguishable from each other.

2. If a Friedman test showed that there were significant differences in performance between the five configurations of k, then a full pairwise comparative analysis was conducted between all ten possible comparisons of the five configurations using a Shaffer post hoc test. The post hoc test determined which configurations were superior, inferior, or similar to each other. Wins and losses were assigned to each configuration using the method described in section 2.7.4.

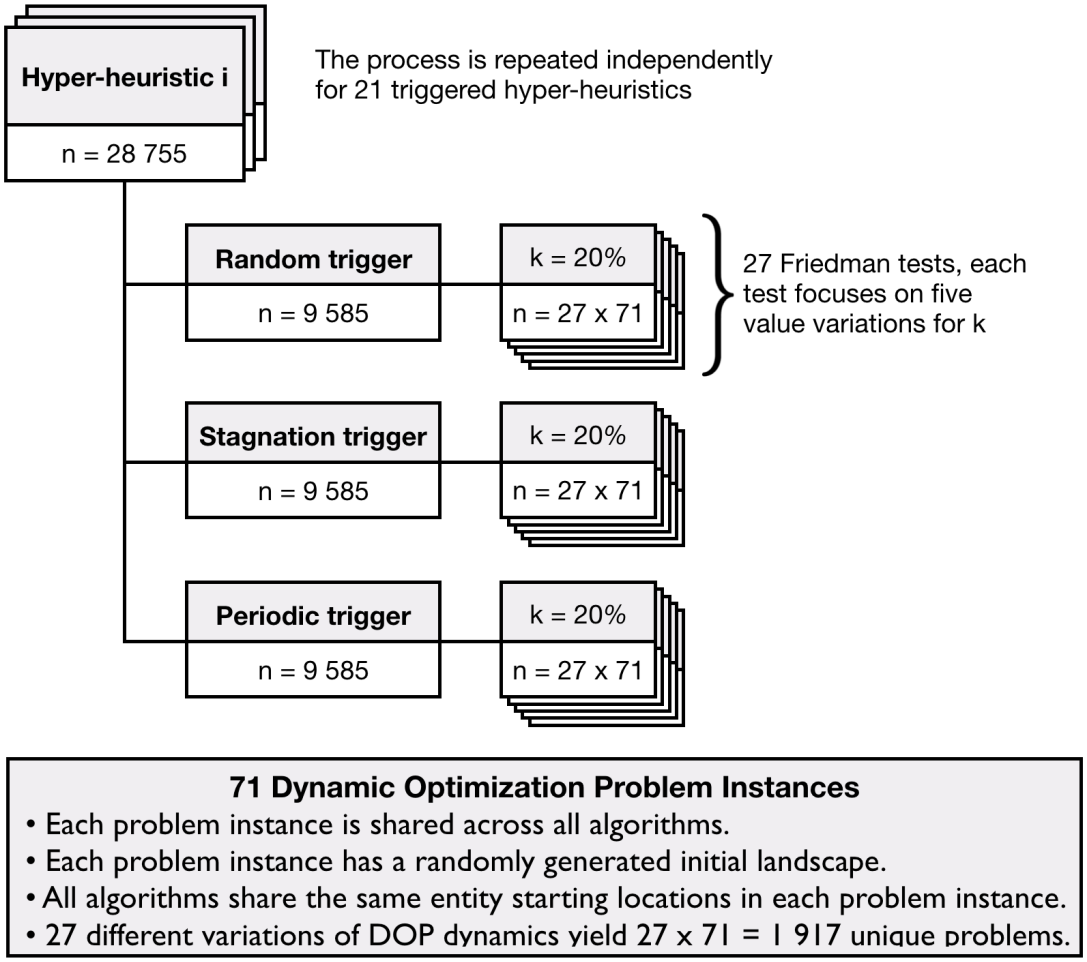


Figure 5.1: Focus of the Friedman test and associated Shaffer post hoc tests for research question 1. The process was repeated for each triggered hyper-heuristic, and the entire experiment was repeated using the *global* and the *island* topologies, respectively. The value n represents the number of algorithm runs underpinning each level.

Contrasting the number of wins and losses achieved by a hyper-heuristic and trigger configuration at different values for k yields insight into how sensitive that configuration was to changes in values for k. The net number of wins minus losses is a scalar value that expresses how well a configuration ranked against all other configurations. A positive wins minus losses value shows that the configuration recorded more wins than losses against other configurations. A negative wins minus losses value indicates that the configuration lost against more configurations than what the configuration won against.

The process was repeated for every combination of trigger type and neighborhood topology. For each combination of hyper-heuristic, trigger type, and neighborhood topology, the above procedure resulted in 27 sets of wins, draws, and losses for the five different values of k. The mean number of wins minus losses (across all 27 environments) for every value of k was then computed for each hyper-heuristic. The highest possible mean wins minus losses value was four, which indicates total dominance by one configuration of k across every environment. The lowest possible value was minus four, indicating that all other configurations of k performed better across every environment. A mean wins minus losses value of zero for any value of k indicates that the performance of the hyper-heuristic was not significantly different from configurations of the same hyper-heuristic using other values of k.

Table 5.2 shows the Pearson correlation between values for k and wins minus losses per hyper-heuristic, individually per trigger and topology variation (across all environments). Figures 5.2, 5.3, and 5.4, respectively show the output of the analysis for the RT, ST, and PT triggers for individual hyper-heuristics using the *global* neighborhood topology. Figures 5.5, 5.6, and 5.7, respectively show the same information for the same experiment carried out using the *island* topology.

The correlation analysis in table 5.2 confirms the presence of predominantly negative correlations: For the majority of hyper-heuristics, regardless of the type of trigger or topology that was used, lower values for k generally yielded higher wins minus losses values. Four noteworthy positive correlations where found that were larger than 0.1, namely **Perm** when using the *Global_PT*, **DProp** and **RLFreq** when using the *Global_RT* configuration, and **ETour2** when using the *Island_ST* configuration. Isolated exceptions exist in the table where correlations were found to be in the range of $[-0.1, 0.1]$, i.e.:

1. **Comp**, **NARank**, **Rand** and **Soft** when using *Global_PT* configurations,
2. **NARank** and **RLProp** when using the *Global_RT* trigger,
3. **DProp**, **ETour2**, **Freq2**, **HTour5M**, **NAProp**, and **RoulX** when using *Island_RT* configurations.

The negative correlations for the majority of hyper-heuristics, topology, and trigger combinations confirm that better performance resulted at lower values for k. Each hyper-

Table 5.2: Pearson correlation between values for k and wins minus losses per hyper-heuristic, individually per trigger and topology variation.

	Global topology			Island topology		
HH	**Global_PT**	**Global_ST**	**Global_RT**	**Island_PT**	**Island_ST**	**Island_RT**
AProp	-0.36	-0.59	-0.33	-0.53	-0.23	-
ARank	-0.05	-0.58	-0.46	-0.54	-0.40	-0.32
Comp	0.08	-0.51	-0.20	-0.60	-0.61	-0.17
DProp	-0.69	-0.59	0.22	-0.53	-0.29	-0.09
ETour2	-0.69	-0.53	-0.47	-0.17	0.13	-0.04
ETour25	-0.72	-0.64	-0.12	-0.33	-0.32	-
Freq2	-0.56	-0.62	-0.31	-0.62	-0.33	-0.09
Freq5	-0.75	-0.64	-0.30	-0.66	-0.55	-0.34
HTour2M	-0.63	-0.61	-0.42	-0.48	-0.35	-0.29
HTour2X	-0.64	-0.57	-0.43	-0.58	-0.49	-0.22
HTour5M	-0.78	-0.67	-0.29	-0.66	-0.50	-0.04
HTour5X	-0.81	-0.66	-0.37	-0.65	-0.57	-0.24
NAProp	-0.45	-0.51	-0.28	-0.51	-0.20	-0.04
NARank	-0.08	-0.59	-0.06	-0.51	-0.44	-0.33
Perm	0.19	-0.47	-0.41	-0.49	-0.43	-0.3
RLFreq	-0.69	-0.67	0.13	-0.63	-0.54	-0.17
RLProp	-0.66	-0.60	0.06	-0.48	-0.51	-
Rand	0.03	-0.48	-0.31	-0.57	-0.36	-0.21
RoulM	-0.53	-0.60	-0.26	-0.48	-0.46	-0.13
RoulX	-0.48	-0.58	-0.18	-0.33	-0.44	0.02
Soft	-0.06	-0.56	-0.22	-0.63	-0.46	-0.22

heuristic, however, showed different sensitivities to k for different topology and trigger combinations.

The following observations, with regard to the stated research question, can be made for the configurations that use the *global* topology:

- **Random trigger under the *global* topology**: Figure 5.2 sheds more light on the relatively weaker correlation coefficients in table 5.2 for the *Global_RT* configuration. The low number of wins minus losses in figure 5.2 imply that the RT trigger overall was relatively insensitive to different values of k. Those hyper-heuristics with strong negative correlation in the table (namely **ARank**, **ETour2**, **HTour2M**, **HTour2X**, and **Perm**) all show positive wins minus losses values for

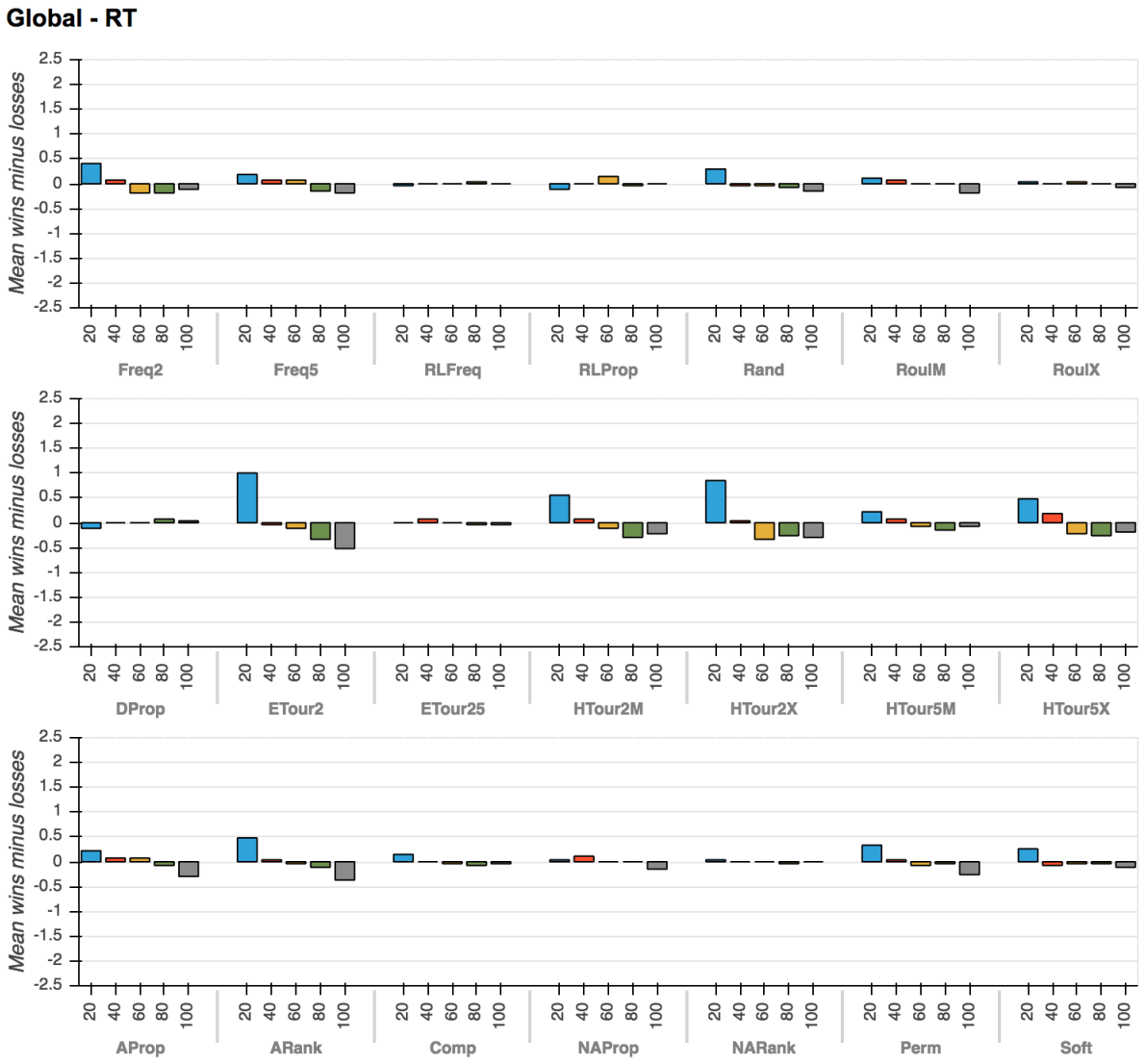


Figure 5.2: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the RT trigger using the *global* topology.

$k = 20\%$ and decreasing (mostly negative) values at higher values for k in figure 5.2.

The other hyper-heuristics that had correlation values that were closer to zero followed the same pattern, where larger values of k yielded lower performance. For these hyper-heuristics, however, the difference in performance gain was almost negligible, i.e., these hyper-heuristics were relatively insensitive to the value of k

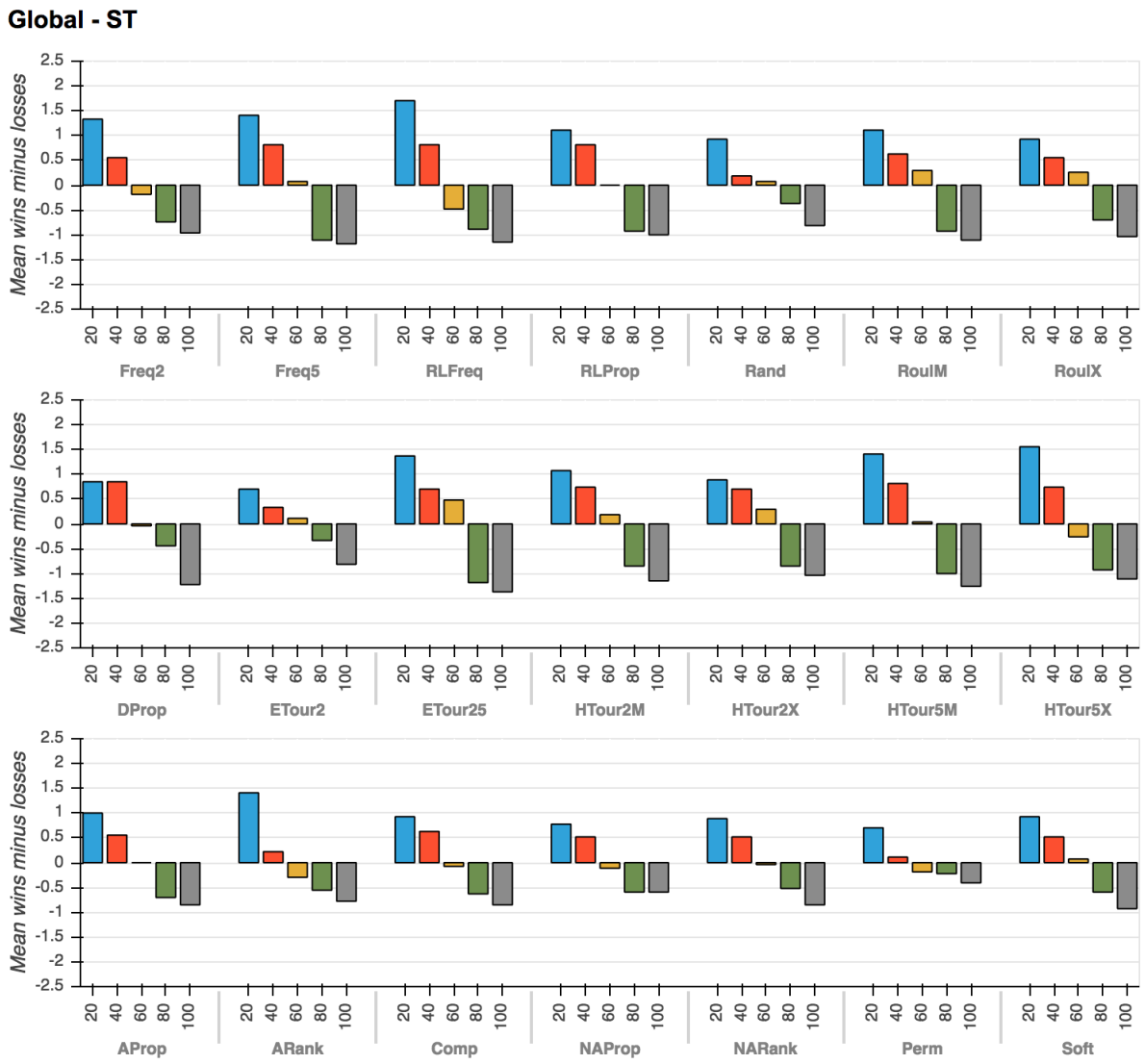


Figure 5.3: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the ST trigger using the *global* topology.

when using the *Global_RT* configuration.

There is evidence that there was a correlation between the tournament size parameter and the value of k in those hyper-heuristics that employ tournament selection. There were large discrepancies visible in the performances of **ETour2** versus **ETour25**, **Freq2** versus **Freq5**, **HTour2M** versus **HTour5M**, and **HTour2X** versus **HTour5X**. For each instance, the variation of the hyper-heuristic with the

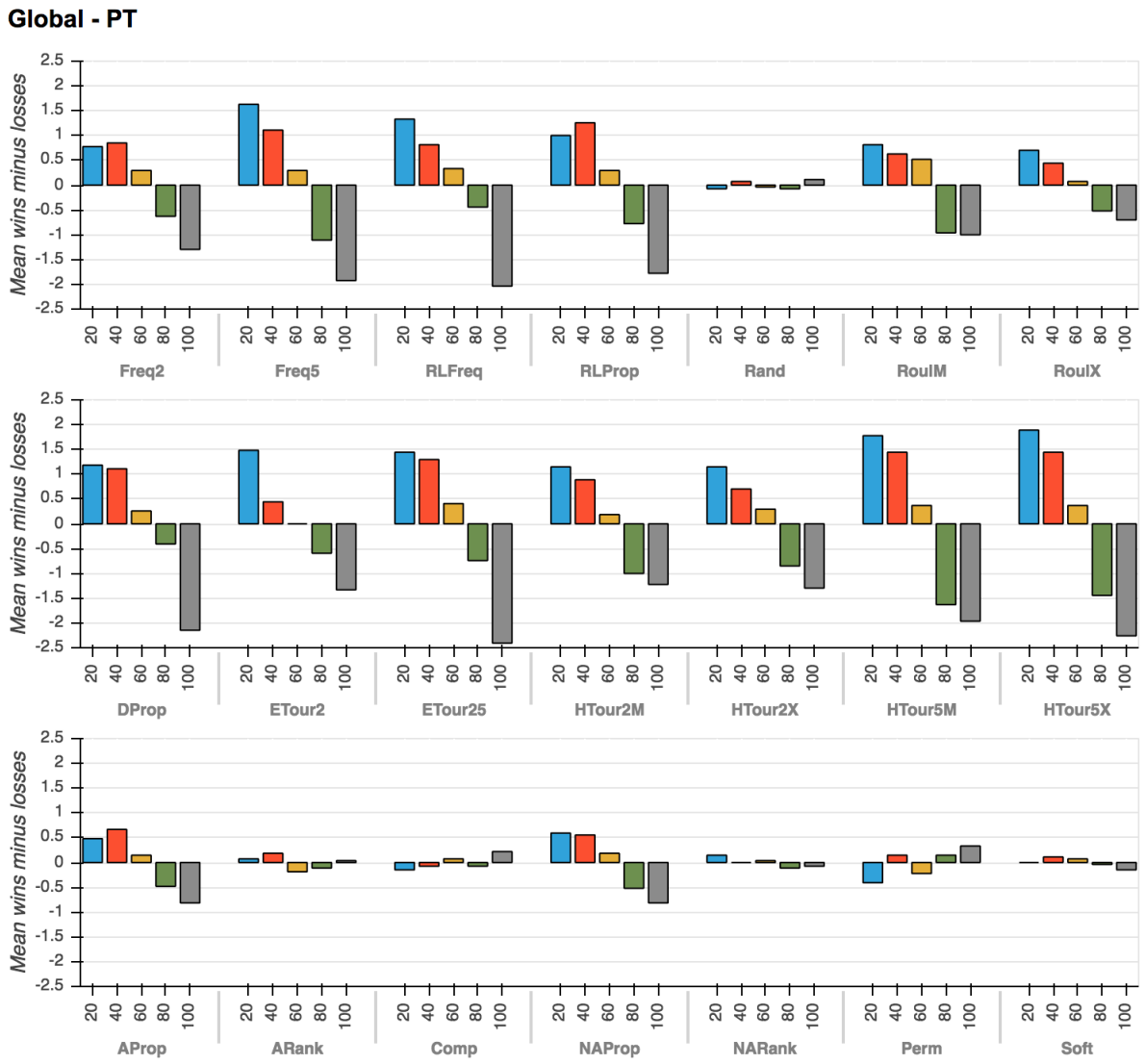


Figure 5.4: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the PT trigger using the *global* topology.

greatest sensitivity to k had a tournament size parameter value of two.

- **Stagnation trigger under the *global* topology**: Visually, the wins minus losses distribution patterns in figure 5.3 appear nearly identical for every hyper-heuristic. Configurations with $k = 20\%$ dominated in every hyper-heuristic and had noticeably better mean win minus losses values than other configurations. The visual

findings reflect the strong negative correlations in table 5.2.

The ST trigger performed better for all of the investigated hyper-heuristics when lower values of k were used. This implies that lower tolerances for what was deemed to be stagnating behavior of entities generally had a positive effect on any hyper-heuristic compared to higher (more lenient) tolerances. It is clear that $k = 20\%$ was the most beneficial setting for k for all investigated hyper-heuristics that used a *stagnation* trigger combined with a *global* topology. In this thesis, the value of k was systematically varied across the range of $k \in \{20\%, 40\%, 60\%, 80\%, 100\%\}$ relative to the length of the change period of each environment, as outlined in section 5.2.2. Future studies should focus on even shorter stagnation periods such as $k \in \{5\%, 10\%\}$ to determine if performance can be improved even more, and whether or not performance degrades if stagnation tolerances are too short.

Tournament size had an influence on performance for those hyper-heuristics that employ tournament selection: **Freq2** versus **Freq5**, **ETour2** versus **ETour25**, **HTour2M** versus **HTour5M**, and **HTour2X** versus **HTour5X** all show that larger tournament sizes performed better. The general shape of the mean wins value distribution remained the same across different tournament sizes for each hyper-heuristic. This behavior is the opposite of what was observed for the RT results, where lower tournament sizes yielded better performance.

- **Periodic trigger under the *global* topology**: Table 5.2 shows large discrepancies where some hyper-heuristics had very strong negative correlation values, but a sizable group of hyper-heuristics showed weak negative (and even positive) correlation values. Figure 5.4 echos this finding and shows that hyper-heuristics configured with PT triggers had more diversified mean wins minus losses histogram shapes than the RT and ST triggers. Most hyper-heuristics had histograms shapes similar to the ST variants in figure 5.3, where the highest number of net wins occurred at $k = 20\%$ and win rates gradually declined as the value of k increased. A number of hyper-heuristics deviated from this trend, namely **Rand**, **ARank**, **Comp**, **NARank**, **Perm**, and **Soft**. These hyper-heuristics showed low sensitivity to different values for k. **Perm** showed a positive correlation between k and wins minus losses values in table 5.2, which is also visible in figure 5.4.

Similar to the ST results, tournament size had an influence on performance for those hyper-heuristics that employed tournament selection: **Freq2** versus **Freq5**, **ETour2** versus **ETour25**, **HTour2M** versus **HTour5M**, and **HTour2X** versus **HTour5X** all show that larger tournament sizes performed better.

The same analysis was performed on the same set of hyper-heuristic algorithms and values of k on identical problem instances, but using the *island* neighborhood topology. The following observations, with regard to the stated research question, can be made about the results of the *island* topology:

- **Random trigger under the *island* topology**: Figure 5.5 shows that most hyper-heuristics had very low value counts across the different values for k. A noteworthy departure from the *global* topology was **ETour2**, which showed no significant differences between configurations in figure 5.5, but showed strong performance at $k = 20\%$ in figure 5.2. Similar trends were observed for **HTour2X** and (to a lesser degree) for **HTour5X**, **Freq2**, and **Rand**.

  Generally, the same conclusions can be made for the RT trigger for the *island* topology as for the *global* topology: lower values of $k \approx 20\%$ tended to perform the same or better than higher values of k, and the performance difference of the majority of the investigated hyper-heuristics were mostly unaffected by the value of k.

- **Stagnation trigger under the *island* topology**: Figure 5.6 shows that the strongest performance occurred at $k = 20\%$ across half of the hyper-heuristics. Notable exceptions were **Freq2**, **HTour2M**, and **HTour2X**, where $k = 20\%$ still yielded the best performance, but faced stronger competition from configurations where $k = 40\%$. A noteworthy difference for the *island* topology compared to the *global* topology was the lack of sensitivity to changing values of k shown by the following hyper-heuristics: **ETour2** and **ETour25**, **DProp**, and **AProp**.

  Generally, low values of $k = 20\%$ yielded the same or better performance than higher values of k across the majority of investigated hyper-heuristics when the ST trigger and an *island* topology were used. Similar to the *global* topology cases, future studies should focus on even shorter stagnation periods such as $k \in \{5\%, 10\%\}$

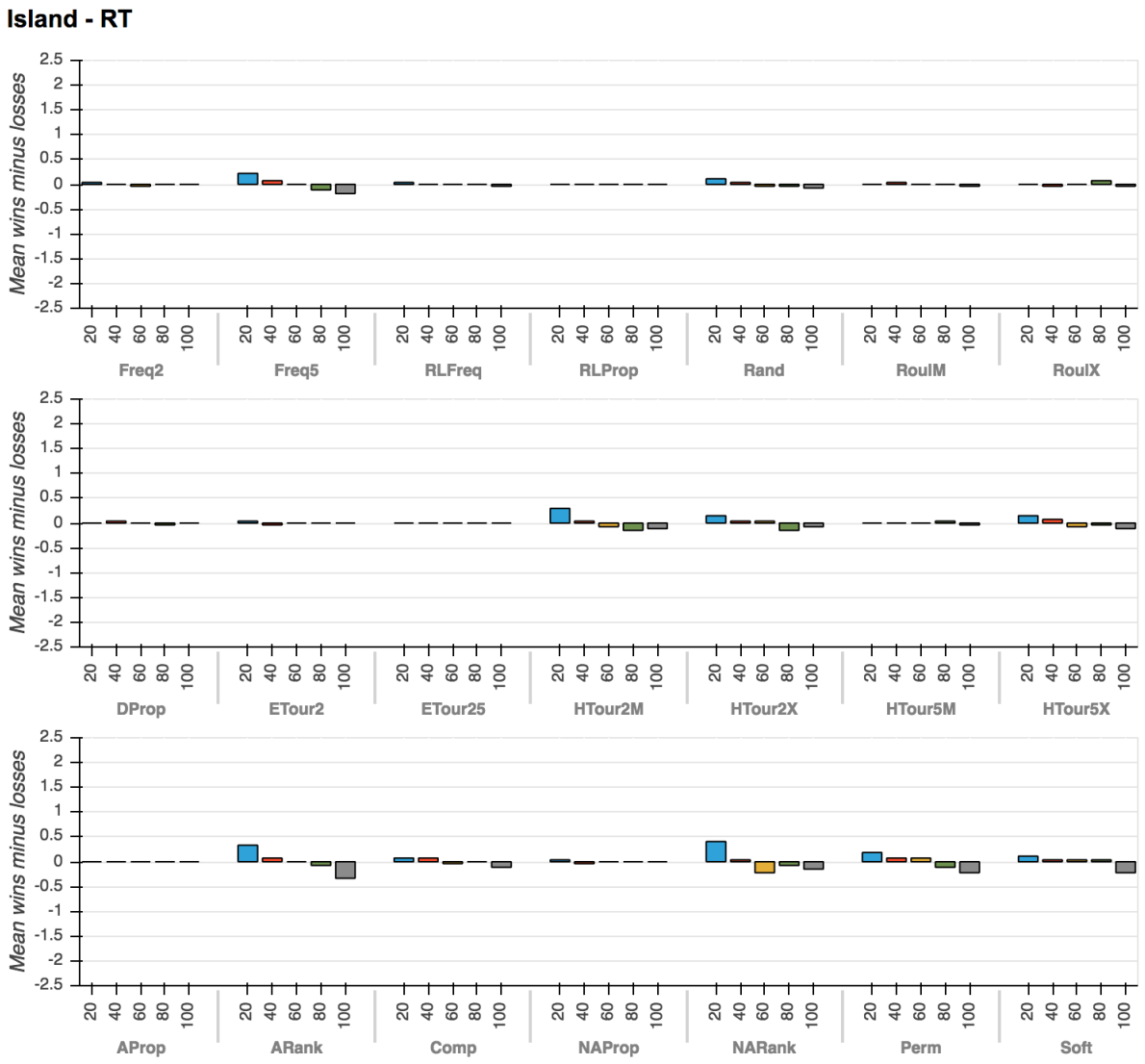


Figure 5.5: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the RT trigger using the *island* topology.

to determine if performance can be improved even more, and whether or not a point exists where performance degrades if stagnation tolerances are too short.

- **Periodic trigger under the *island* topology**: Figure 5.7 shows distinctly different distributions of values for nearly every hyper-heuristic compared to their *global* topology counterparts in figure 5.4. This observation is echoed in table 5.2

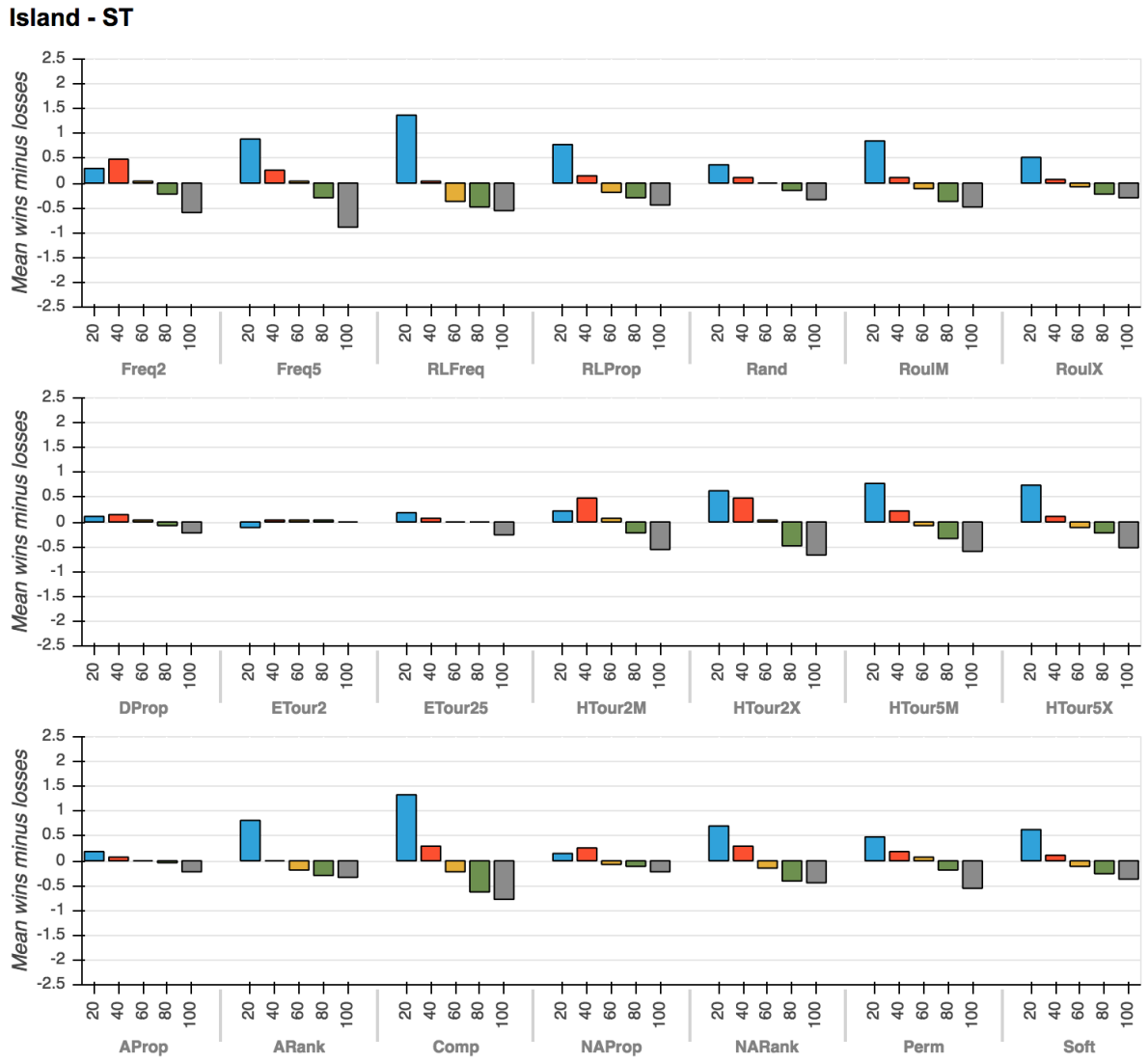


Figure 5.6: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the ST trigger using the *island* topology.

where the difference in correlation coefficients were large for nearly every hyper-heuristic. However, for the majority of hyper-heuristics, those configurations that used $k = 20\%$ yielded noticeably better performance. Both **ETour2** and **ETour25** were almost totally invariant to the value of k for the *island* topology, which was not the case for the *global* topology.

The addition of multiple types of trigger mechanisms yielded noticeably varied results

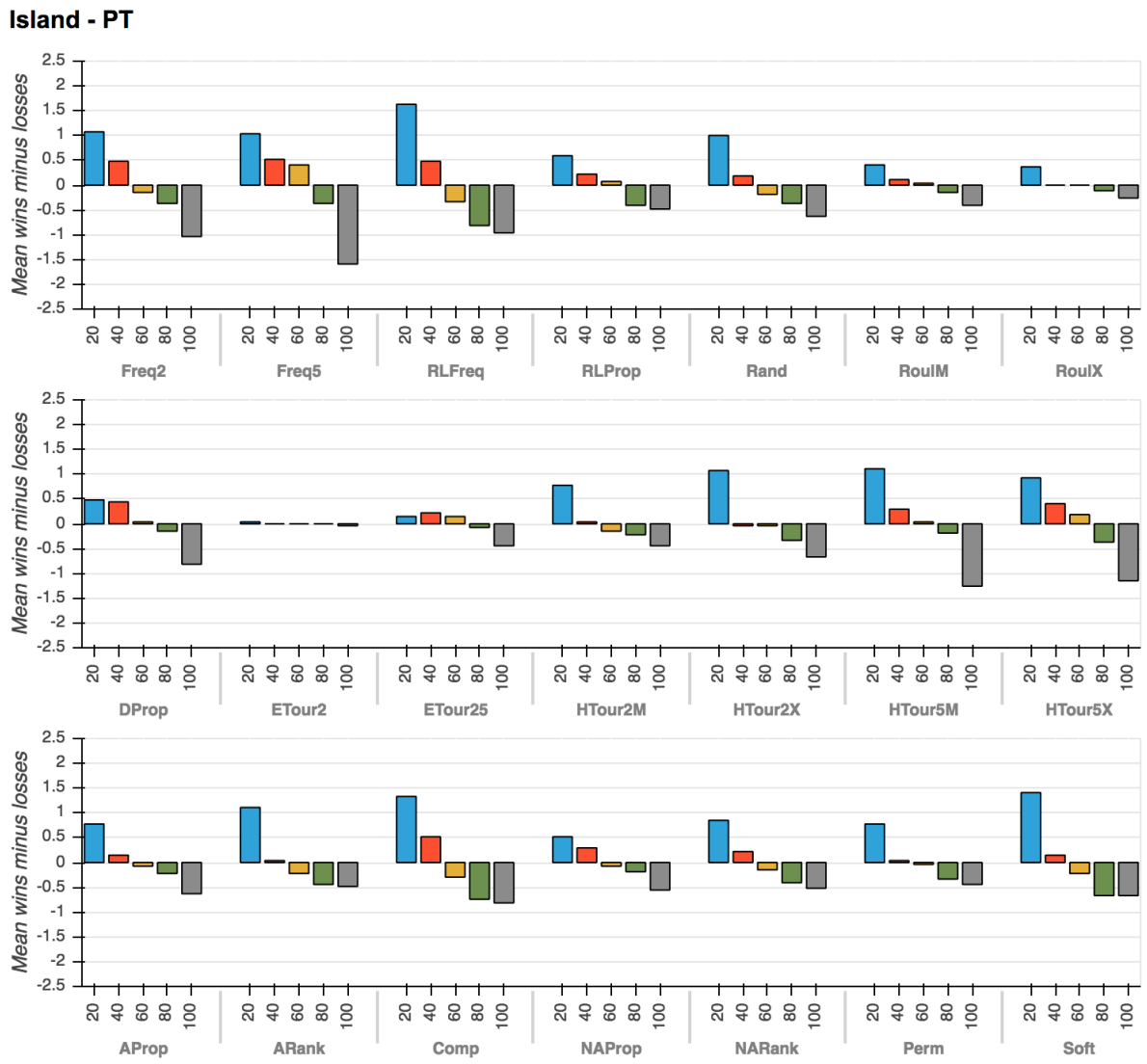


Figure 5.7: Mean wins minus losses across all 27 environments achieved by each hyper-heuristic for five possible values of k for the PT trigger using the *island* topology.

pertaining to the frequency with which heuristic selection was performed by each hyper-heuristic. Various triggers showed different sensitivities to the value of the parameter k. The same trigger also performed differently with alternate sensitivities to values of k when a *global* versus an *island* neighborhood topology was used.

Every trigger type, for the *global* or *island* topology alike, generally showed better performance at lower values of $k = 20\%$ for nearly all the investigated hyper-heuristics.

Table 5.2 support the visual observations in the respective figures by showing mostly negative correlations between values for k and good performance. More frequent heuristic selection for entities was more beneficial than less frequent heuristic changes. Even under a random trigger, more frequent heuristic changes are beneficial, implying that simple algorithmic diversity is effective at improving performance.

However, these insights resulted from an aggregate study across all 27 environments. Future studies need to determine if performance sensitivity to the value of k is strongly problem dependent or not.

5.3.3 Research Question 2

> *Does the type of trigger significantly alter performance of the same hyper-heuristic?*

The original HMHH algorithm relies on the *periodic* trigger mechanism to determine when heuristic changes are required for entities. Two additional HMHH triggers, namely *random* and *stagnation* triggers, are adapted from heterogeneous PSO behavior schedules for use as heuristic selection triggers (as explained in section 3.4.4). This research question expands the analysis of the previous research question by comparing the performance of each type of trigger that each use different values of k (independently for each hyper-heuristic). Key questions come to mind when considering the effect of different triggers on HMHH:

1. Were there any broadly observable visually identifiable trends across triggers?
2. Was the performance of any hyper-heuristics sensitive or insensitive to trigger changes?
3. Did ST or RT configurations ever outperform PT configurations?
4. Did any hyper-heuristics perform noticeably better with specific trigger choices?
5. Were any trigger configurations sensitive to different DOP types?
6. Was any one trigger always the best performer across all hyper-heuristics?

These questions above were answered as follows. A Friedman test and Shaffer post hoc test was performed to compare the P_r measurements of all 15 trigger variations (i.e., three triggers, each with five different values for k) of each hyper-heuristic individually. This test was performed independently for each environment. The *island* and *global* topology configurations were compared and ranked independently as well. Figure 5.8 illustrates the boundaries of each Friedman test and all associated Shaffer post hoc tests in this research question. Similar to the previous research question, the resulting ranks and wins, draws, and losses were only compared to the variations of the same hyper-heuristic. Individual Friedman and Shaffer post hoc tests for each environment yielded the number of pairwise wins, draws, and losses for each of the 15 trigger configurations against the 14 other trigger configurations.

The aim in this research question is to determine, for each hyper-heuristic, which trigger configurations yielded more wins than losses against other trigger configurations more often across environments. The net number of wins minus losses is a scalar value that expresses how well a configuration ranked against all other configurations in an environment. Higher wins minus losses values reveal trigger configurations that outperformed most other configurations while simultaneously not being outperformed by many others configurations. The lower the wins minus losses value for a configuration, the greater the number of other configurations that performed better than the configuration. Wins minus loss values are symmetrical around zero, since the sum of all wins, draws, and losses is zero. In a test with a large number of draws between configurations, the wins minus losses value of each configuration will be close to zero (since most methods will, mostly, record zero wins and zero losses). For a test where most configurations had significantly different performance value distributions, the wins minus loss value distribution spanned the nearly entire range from $[-14, 14]$.

Aggregated across all 27 environments, the wins minus losses values of each trigger configuration forms a distribution of integer values. The shape and location of the distribution expresses the tendency each configuration had to perform better or worse than other configurations. Narrow distributions indicate that a configuration tended to be ranked the same way against other configurations, regardless of the environment at hand. Wider distributions show that the configuration was prone to perform well against other configurations in certain environments, while being outperformed by the

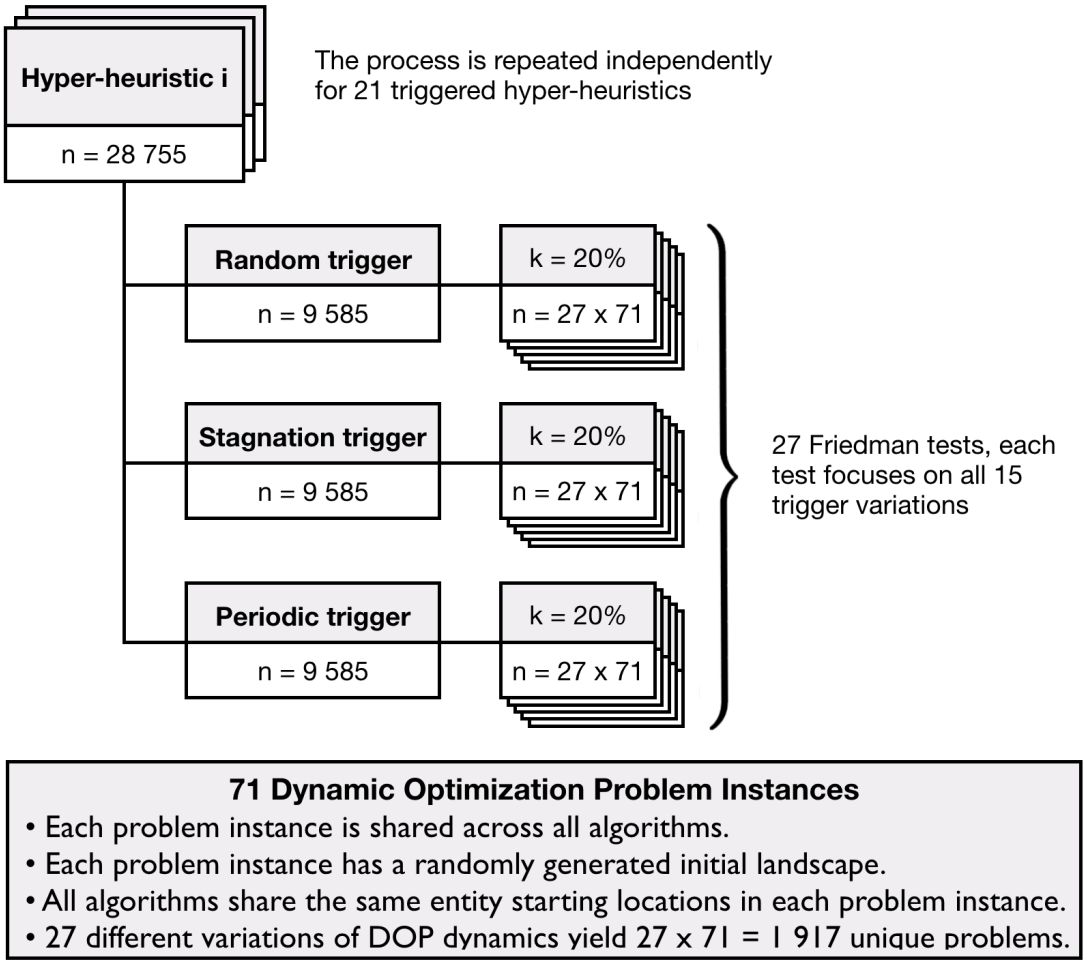


Figure 5.8: Focus of the Friedman test and associated Shaffer post hoc tests for research question 2. The process was repeated for each triggered hyper-heuristic, and the entire experiment was repeated using the *global* and the *island* topologies, respectively. The value n represents the number of algorithm runs underpinning each level.

other methods in certain environments. Distributions with predominantly positive values reveal configurations that always tend to outperform other configurations in most environments, while distributions with predominantly negative values highlight configurations that perform worse than other configurations, regardless of the environment at hand.

Figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, and 5.15 show the distributions of Friedman ranks and net wins minus losses achieved by different trigger configurations for each

individual hyper-heuristic. The *global* and *island* topology results are shown separately. The ranks of each trigger configuration were aggregated across the 27 environments as box whisker plots.

The wins minus losses values for each configuration are shown as a heat map, where the y-axis indicates the net number of wins achieved, as discussed in the paragraphs above. The darkness of each cell indicates the number of environments in which specific wins minus losses values were achieved. For example, in figure 5.9, the **ARank (Island)** heat map shows that RT configurations yielded a high proportion of net wins for roughly half of all environments, and approximately zero values for the remainder of the environments. Similarly, **AProp (Island)** shows a dark band across the zero value cells for the RT configuration, indicating that most configurations yielded zero wins minus losses values against the other configurations for approximately 25 of the 27 environments. The sum of each column in each heat map is equal to the total number of environments, namely 27.

Table 5.3 summarizes the wins minus losses values for each hyper-heuristic by calculating the average values across all 27 environments, for each of the 15 trigger and topology configurations. The average and standard deviation of the wins-minus-losses across all hyper-heuristics (i.e., columns) are also shown for each configuration. Table 5.4 shows the Pearson correlation [162] between the Friedman ranks and k, as well as the correlation between the wins minus losses values and k for each hyper-heuristic, for all trigger configurations and topologies.

The following observations, with regard to the stated research question, can be made when considering the results in figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, and 5.15 and tables 5.3 and 5.4:

Were there any broadly observable visually identifiable trends across triggers?

At a glance, nearly every hyper-heuristic showed a wide array of both positive and negative wins minus losses values in their respective heat maps, for the *island* and *global* topologies alike. Isolated exceptions resulted when the *island* topology was used, namely **ETour2**, **ETour25**, and **DProp**, which yielded mostly zero or approximately zero wins minus losses values across environments. Such a large proportion of environments show-

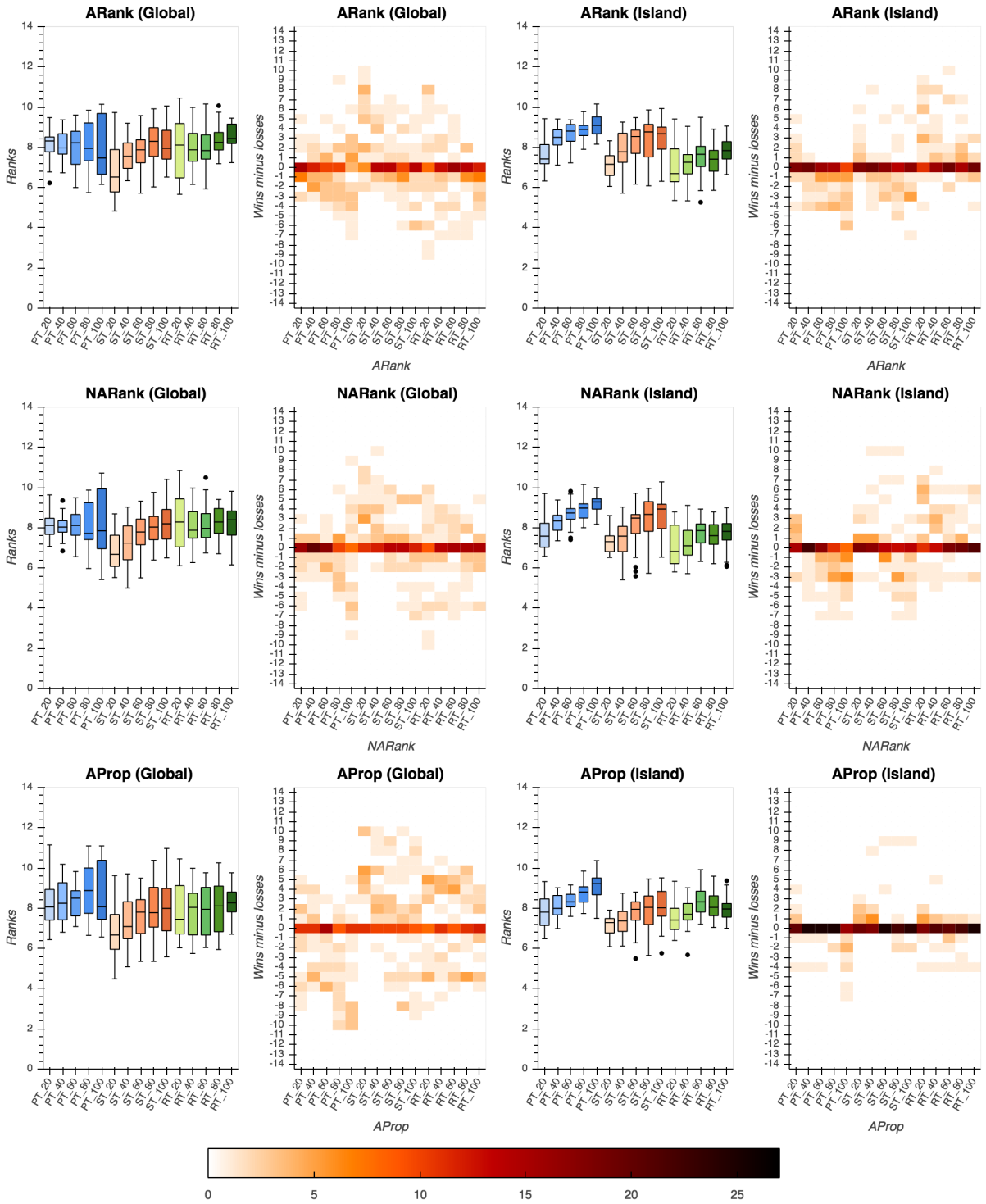


Figure 5.9: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 1 of 7).

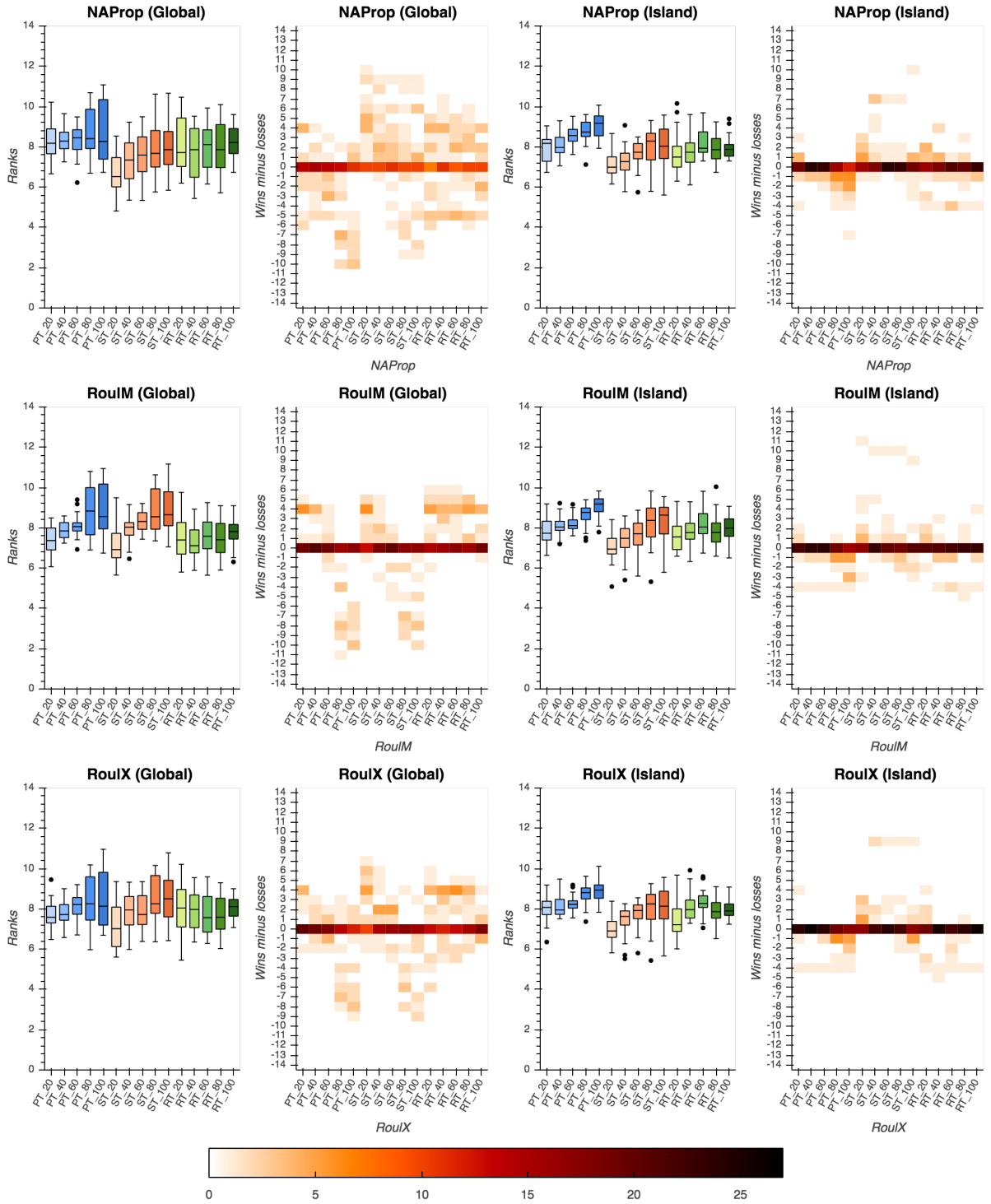


Figure 5.10: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 2 of 7).

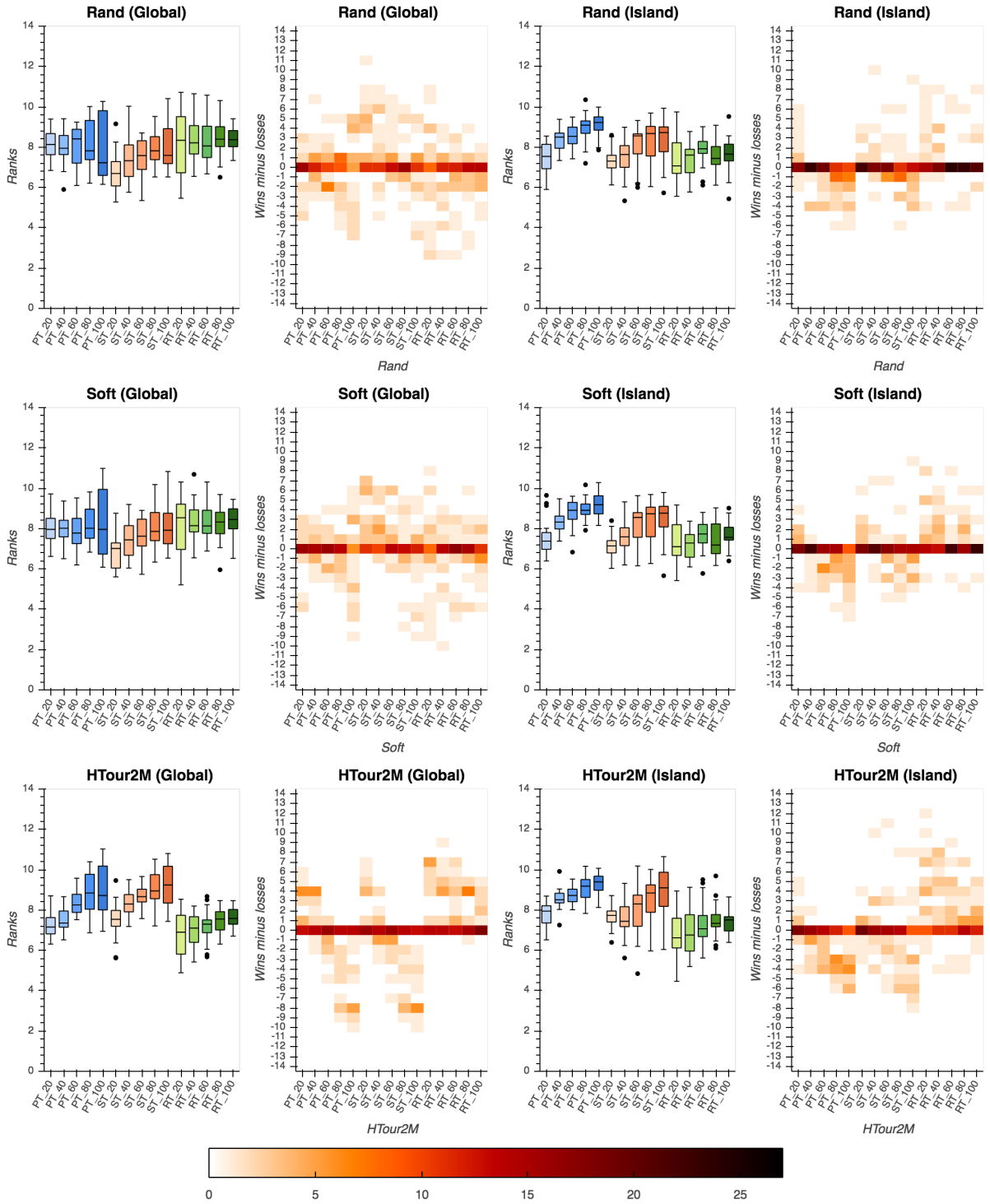


Figure 5.11: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 3 of 7).

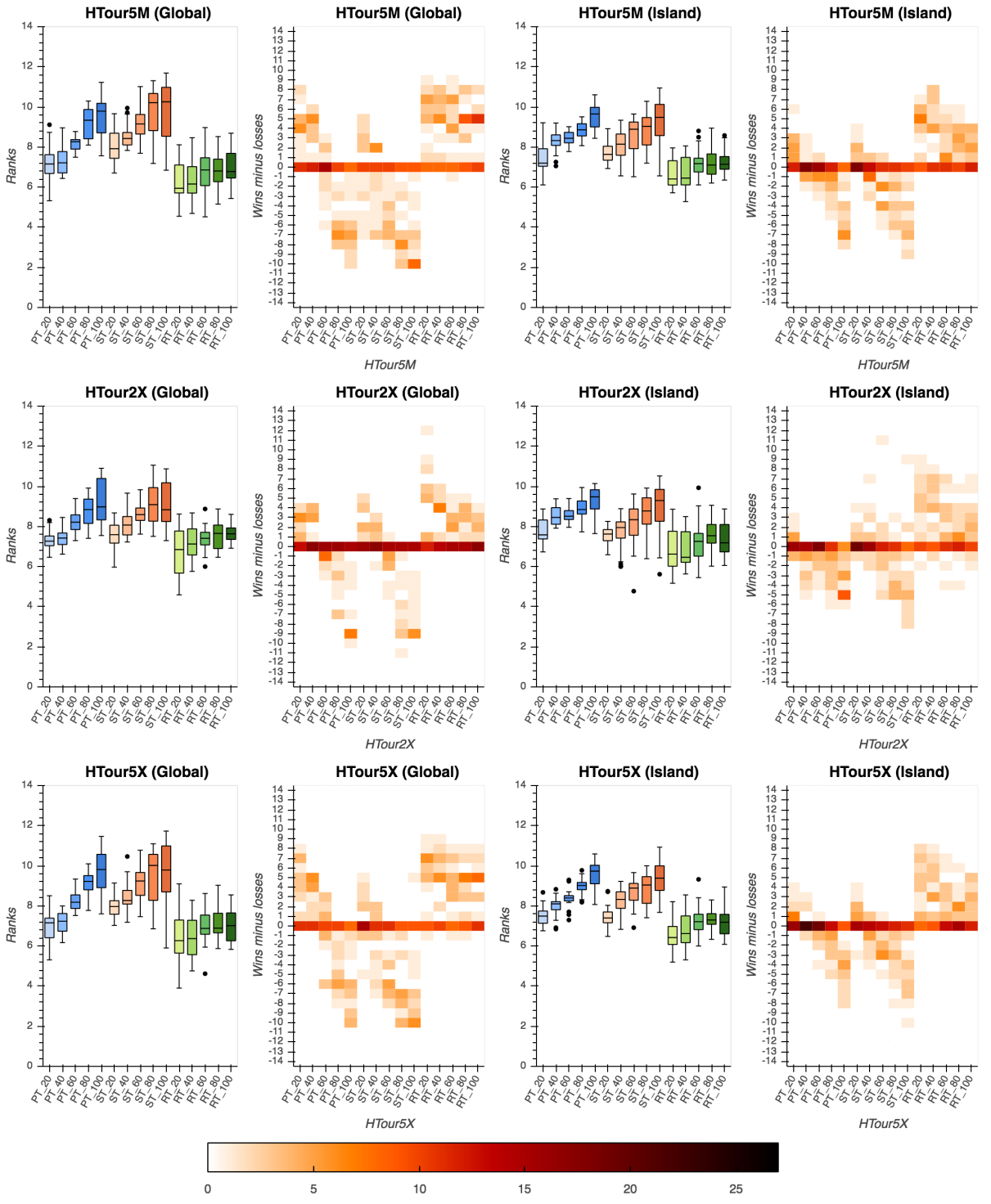


Figure 5.12: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 4 of 7).

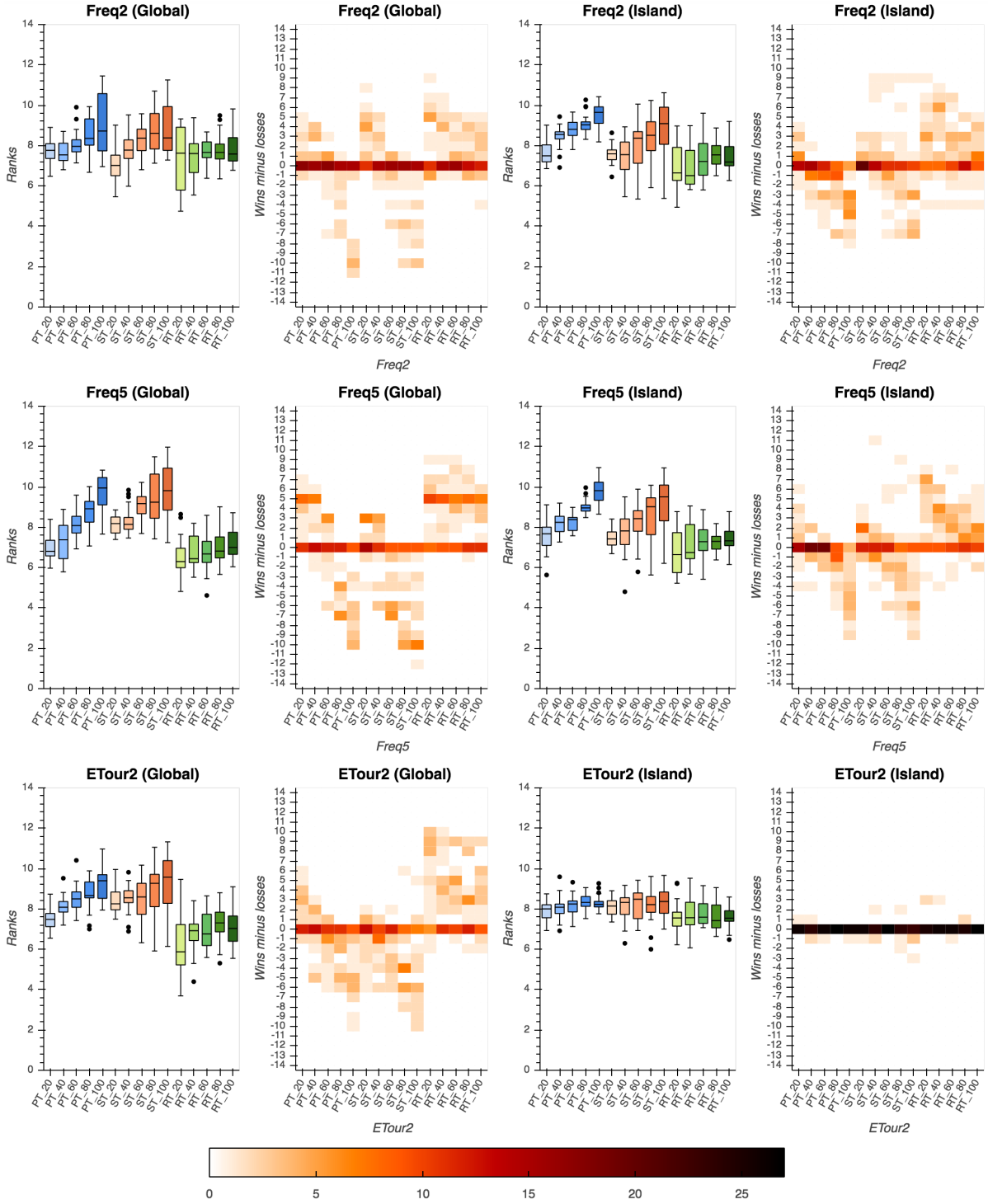


Figure 5.13: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 5 of 7).

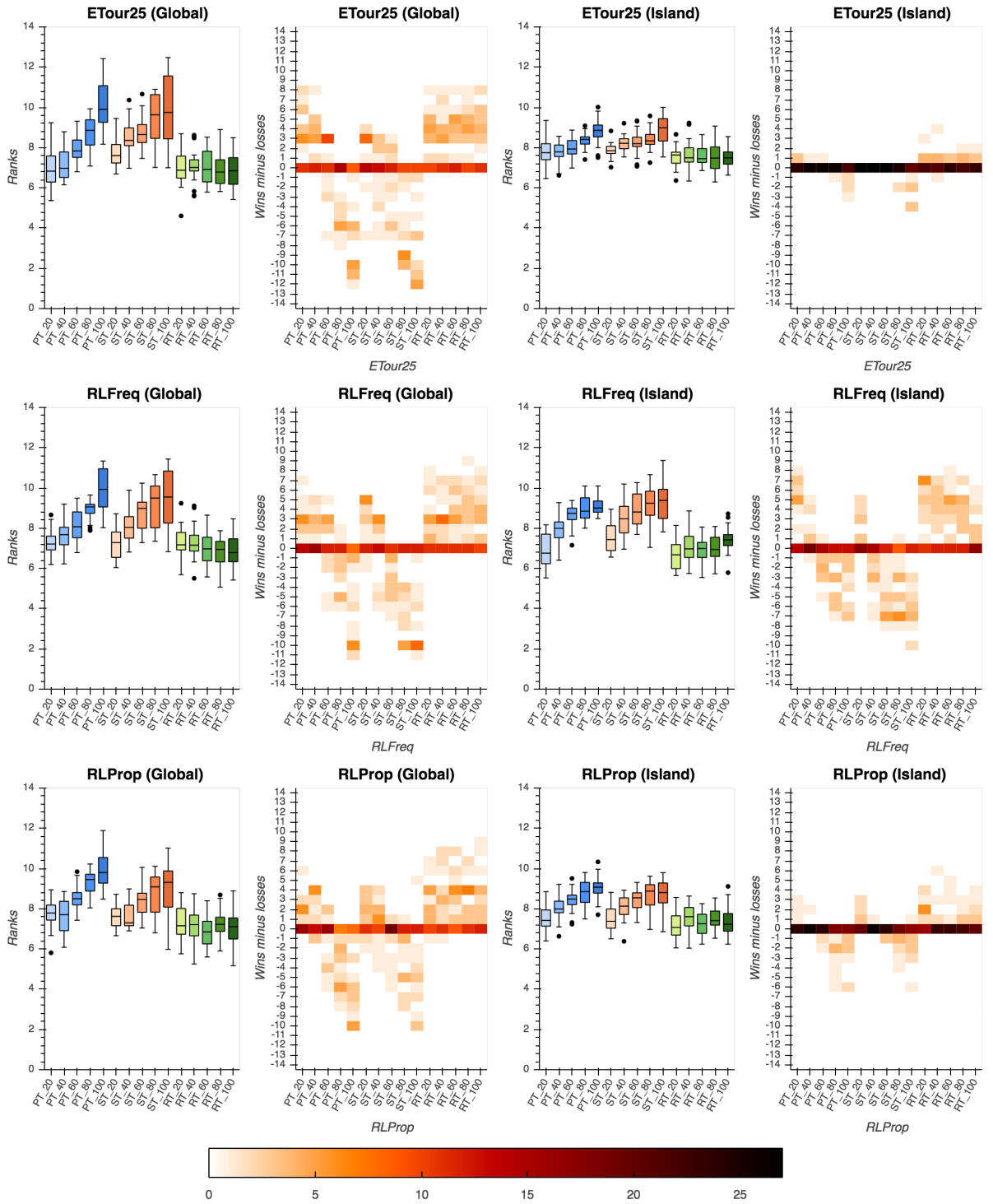


Figure 5.14: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 6 of 7).

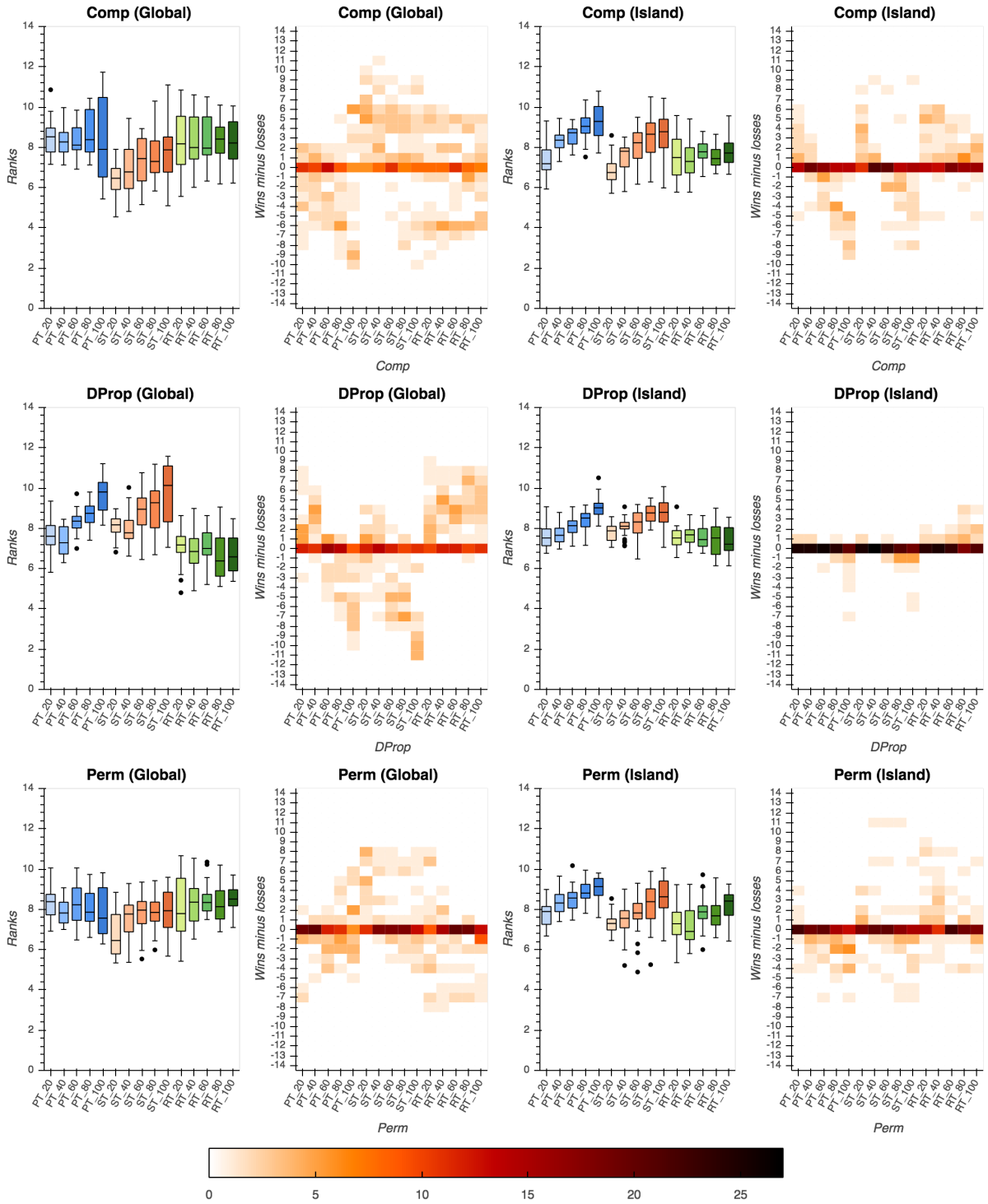


Figure 5.15: Distribution of Friedman ranks and wins minus losses across all DOPs for each hyper-heuristic (part 7 of 7).

Table 5.3: Average wins minus losses per hyper-heuristic for each trigger and value of k for both topologies. Averages were computed across the 27 environments. Bold values indicate the best configurations.

HH	PT_20	PT_40	PT_60	PT_80	PT_100	ST_20	ST_40	ST_60	ST_80	ST_100	RT_20	RT_40	RT_60	RT_80	RT_100
AProp (Global)	**-0.67**	-0.78	-1.52	-3.22	-3.11	**3.52**	2.44	1.30	-0.37	0.41	0.41	**0.81**	0.59	0.44	-0.26
AProp (Island)	**0.15**	-0.19	-0.19	-0.15	-1.26	0.48	**0.67**	0.48	0.19	0.19	**0.15**	0.07	-0.26	-0.22	-0.11
ARank (Global)	**-0.11**	-0.30	-0.44	-0.37	-0.63	**3.56**	1.04	0.59	-0.19	-0.74	**0.44**	-0.41	-0.48	-0.74	-1.22
ARank (Island)	**0.30**	-0.59	-0.89	-1.07	-1.96	**0.70**	-0.11	-0.37	-0.63	-0.81	**2.37**	1.30	0.89	0.85	0.04
Comp (Global)	-1.26	**-0.93**	-1.37	-2.52	-1.67	**4.15**	3.22	2.11	0.85	0.81	**-0.52**	-0.78	-0.81	-0.59	-0.70
Comp (Island)	**1.04**	0.11	-0.78	-1.96	-3.04	**2.11**	0.56	-0.26	-1.15	-1.48	**1.52**	1.44	0.85	0.63	0.41
DProp (Global)	1.33	**1.93**	-0.52	-1.41	-3.78	**0.15**	-0.07	-2.59	-2.89	-5.07	2.15	2.63	2.22	**3.11**	2.81
DProp (Island)	0.07	0.07	-0.07	-0.04	-0.70	**0.07**	0	-0.04	-0.26	-0.67	0.15	0.07	0.19	**0.70**	0.44
ETour2 (Global)	**1.44**	-0.37	-1.04	-1.93	-3.30	**-0.78**	-1.22	-1.70	-2.96	-3.74	**4.74**	3.11	2.89	2.41	2.44
ETour2 (Island)	**0.04**	-0.04	-0.04	0	-0.04	**-0.04**	0	0	-0.04	-0.22	**0.22**	0.11	-0.04	0.07	0
ETour25 (Global)	**2.70**	2.07	0.56	-2.44	-5.52	**0.89**	-0.41	-1.00	-4.56	-5.33	2.52	2.44	2.41	**2.85**	2.81
ETour25 (Island)	**0.11**	0.04	0.04	-0.04	-0.33	0	0	0	-0.15	-0.70	0.26	0.26	0.11	0.22	0.19
Freq2 (Global)	0.67	**0.96**	-0.19	-1.56	-3.63	**2.22**	0.89	-0.63	-2.19	-2.81	**1.96**	1.63	0.93	0.89	0.85
Freq2 (Island)	**0.67**	-0.63	-1.30	-1.89	-3.59	0.44	**1.22**	-0.04	-0.67	-2.19	2.07	**2.74**	1.33	0.81	1.00
Freq5 (Global)	**2.63**	1.81	0.15	-2.96	-4.89	**0.63**	-0.15	-3.30	-4.59	-5.07	**3.63**	3.19	3.48	2.89	2.56
Freq5 (Island)	**0.63**	-0.07	-0.26	-1.44	-4.26	**0.81**	0.26	-0.52	-1.44	-2.89	3.11	**2.22**	1.44	1.26	1.15
HTour2M (Global)	**1.67**	1.15	-0.59	-2.52	-3.11	**1.48**	0.11	-1.00	-2.78	-3.37	**2.48**	2.19	1.89	1.44	0.96
HTour2M (Island)	**0.30**	-0.70	-1.26	-2.04	-2.70	0.07	**0.33**	-0.04	-1.19	-1.89	**3.00**	2.56	1.63	0.93	1.00
HTour2X (Global)	**1.26**	1.19	-0.56	-1.59	-3.30	**1.19**	0.56	-0.78	-3.00	-3.22	**2.93**	1.93	1.19	1.37	0.85
HTour2X (Island)	**0.22**	-1.07	-0.63	-1.59	-2.85	0.33	**0.52**	-0.33	-1.59	-2.41	2.78	**2.81**	1.52	0.93	1.37
HTour5M (Global)	**2.70**	2.37	-0.33	-3.81	-4.67	**0.22**	-1.22	-3.26	-4.81	-5.19	**3.93**	3.89	3.59	3.33	3.26
HTour5M (Island)	**1.22**	-0.04	-0.41	-1.22	-3.81	**0.67**	-0.48	-1.56	-2.22	-3.48	3.07	**3.30**	1.63	1.93	1.41
HTour5X (Global)	**2.85**	2.19	-0.26	-3.04	-4.70	**0.63**	-1.04	-3.33	-4.59	-4.85	**3.74**	3.67	3.07	2.78	2.89
HTour5X (Island)	**0.67**	0.19	-0.41	-1.26	-3.26	**0.63**	-0.59	-1.19	-1.89	-3.11	**3.15**	2.85	1.56	1.04	1.63
NAProp (Global)	**-0.19**	-0.52	-0.93	-2.81	-3.04	**3.30**	2.26	1.19	-0.22	0.04	-0.04	**0.59**	0.22	0.33	-0.19
NAProp (Island)	**0.04**	-0.11	-0.19	-0.70	-1.22	0.59	**1.22**	0.37	0.33	0.30	0	**0.22**	-0.48	-0.11	-0.26
NARank (Global)	-0.52	**-0.15**	-0.26	-0.78	-1.00	**2.59**	1.85	0.78	0.22	-0.70	-0.67	**-0.19**	-0.30	-0.41	-0.48
NARank (Island)	**0.52**	-0.56	-0.96	-1.74	-1.96	0.44	**0.93**	0.44	-0.74	-1.26	**2.11**	1.15	0.63	0.52	0.48
Perm (Global)	-0.81	-0.33	-0.93	0.04	**0.30**	**3.07**	1.26	0.70	0.41	0.37	**-0.22**	-0.70	-1.04	-0.96	-1.15
Perm (Island)	**0.07**	-0.56	-0.67	-1.00	-1.48	0.48	**0.63**	0.41	-0.30	-0.81	**1.41**	1.37	0.33	0.11	0
RLFreq (Global)	**1.52**	1.22	0.33	-1.41	-4.67	**1.74**	0.19	-2.04	-4.00	-4.48	2.00	2.04	2.48	2.52	**2.56**
RLFreq (Island)	**2.67**	0.63	-0.89	-2.04	-1.96	**1.11**	-1.33	-2.96	-3.15	-3.48	**3.11**	2.56	2.44	2.26	1.04
RLProp (Global)	0.96	**1.15**	-0.74	-3.15	-4.70	**1.04**	0.56	-0.67	-2.15	-3.04	1.74	2.26	**2.63**	2.11	2.00
RLProp (Island)	**0.22**	0.07	-0.15	-1.00	-0.89	**0.48**	0.04	-0.11	-0.56	-0.74	**0.93**	0.41	0.48	0.30	0.52
Rand (Global)	-0.41	**0.22**	-0.07	-0.67	-0.41	**2.74**	1.30	1.07	0.37	-0.41	-0.63	-1.00	-0.74	-0.74	-0.63
Rand (Island)	**0.74**	-0.48	-0.78	-1.26	-1.52	0.44	**0.56**	0.19	-0.41	-0.67	**1.30**	0.96	0.33	0.07	0.52
RoulM (Global)	**1.11**	0.74	0.26	-2.74	-3.00	**1.56**	0.63	-0.33	-2.37	-2.41	1.41	**1.59**	1.41	1.41	0.74
RoulM (Island)	**-0.04**	-0.22	-0.22	-0.37	-0.85	**0.93**	0.56	0.56	0.19	0.11	**0.19**	-0.11	-0.41	-0.22	-0.07
RoulX (Global)	**0.70**	0.19	-0.04	-1.85	-2.30	**2.22**	0.89	0.30	-1.67	-1.74	0.37	0.85	**0.96**	0.93	0.19
RoulX (Island)	**-0.04**	-0.15	-0.22	-0.48	-0.67	0.67	**0.85**	0.52	0.37	0.22	**0**	-0.33	-0.44	-0.15	-0.15
Soft (Global)	-0.26	**0.04**	-0.07	-0.44	-1.07	**2.41**	1.56	0.93	-0.11	-0.78	-0.37	-0.52	-0.63	**-0.19**	-0.48
Soft (Island)	**0.37**	-0.33	-1.00	-0.93	-2.04	**0.78**	0.26	-0.37	-0.67	-0.52	1.44	**1.48**	0.44	0.81	0.26
μ (Global)	**0.83**	0.66	-0.41	-1.96	-2.96	**1.83**	0.70	-0.56	-1.98	-2.44	**1.52**	1.39	1.24	1.20	0.94
μ (Island)	**0.47**	-0.22	-0.54	-1.06	-1.92	**0.58**	0.29	-0.23	-0.76	-1.26	**1.54**	1.31	0.68	0.61	0.52
σ (Global)	1.24	1.02	**0.53**	1.06	1.63	1.28	**1.16**	1.61	1.86	2.06	1.64	1.51	1.49	**1.41**	1.49
σ (Island)	0.60	**0.37**	0.40	0.67	1.18	**0.45**	0.60	0.80	0.88	1.20	1.18	1.14	0.80	0.65	**0.56**

Table 5.4: Pearson correlation between Friedman ranks and k, as well as between wins minus losses and k per hyper-heuristic, across all trigger and topology variations.

	Correlation between ranks and k								**Correlation between wins minus losses and k**							
HH	**Global_PT**	**Global_RT**	**Global_ST**	**Island_PT**	**Island_RT**	**Island_ST**	μ	σ	**Global_PT**	**Global_RT**	**Global_ST**	**Island_PT**	**Island_RT**	**Island_ST**	μ	σ
AProp	0.18	0.11	0.34	0.61	0.25	0.42	0.32	0.16	-0.29	-0.07	-0.31	-0.31	-0.11	-0.09	-0.20	0.11
ARank	0.00	0.21	0.43	0.62	0.29	0.46	0.34	0.20	-0.07	-0.18	-0.46	-0.41	-0.29	-0.26	-0.28	0.13
Comp	0.04	0.02	0.38	0.69	0.15	0.57	0.31	0.26	-0.10	-0.01	-0.33	-0.54	-0.21	-0.46	-0.27	0.19
DProp	0.69	-0.17	0.46	0.68	-0.11	0.54	0.35	0.35	-0.61	0.10	-0.51	-0.29	0.22	-0.32	-0.24	0.30
ETour2	0.66	0.29	0.26	0.29	-0.02	0.09	0.26	0.21	-0.57	-0.22	-0.36	-0.07	-0.15	-0.13	-0.25	0.17
ETour25	0.78	-0.09	0.54	0.58	-0.06	0.54	0.38	0.33	-0.68	0.06	-0.55	-0.32	-0.04	-0.32	-0.31	0.26
Freq2	0.48	0.18	0.56	0.74	0.23	0.43	0.44	0.19	-0.50	-0.19	-0.56	-0.61	-0.22	-0.33	-0.40	0.16
Freq5	0.79	0.27	0.57	0.78	0.20	0.51	0.52	0.23	-0.69	-0.13	-0.56	-0.64	-0.29	-0.42	-0.45	0.20
HTour2M	0.63	0.38	0.59	0.69	0.26	0.42	0.50	0.15	-0.59	-0.22	-0.56	-0.51	-0.27	-0.27	-0.40	0.15
HTour2X	0.71	0.40	0.59	0.66	0.26	0.48	0.52	0.15	-0.58	-0.29	-0.54	-0.52	-0.25	-0.39	-0.43	0.13
HTour5M	0.77	0.26	0.53	0.78	0.32	0.56	0.54	0.20	-0.71	-0.09	-0.51	-0.66	-0.28	-0.56	-0.47	0.22
HTour5X	0.81	0.23	0.54	0.83	0.33	0.65	0.56	0.22	-0.70	-0.12	-0.52	-0.63	-0.29	-0.52	-0.46	0.20
NAProp	0.20	0.08	0.37	0.59	0.13	0.42	0.30	0.18	-0.35	-0.03	-0.33	-0.40	-0.10	-0.12	-0.22	0.14
NARank	0.07	0.01	0.44	0.65	0.26	0.48	0.32	0.23	-0.09	0.01	-0.41	-0.46	-0.24	-0.27	-0.24	0.16
Perm	-0.11	0.15	0.36	0.59	0.38	0.46	0.30	0.23	0.18	-0.11	-0.31	-0.33	-0.25	-0.20	-0.17	0.17
RLFreq	0.77	-0.21	0.61	0.77	0.30	0.58	0.47	0.34	-0.61	0.09	-0.58	-0.59	-0.25	-0.49	-0.40	0.25
RLProp	0.75	-0.09	0.50	0.67	0.06	0.52	0.40	0.31	-0.65	0.02	-0.52	-0.38	-0.11	-0.43	-0.34	0.23
Rand	0.02	0.07	0.40	0.65	0.13	0.42	0.28	0.22	-0.05	0.01	-0.36	-0.43	-0.17	-0.23	-0.20	0.16
RoulM	0.53	0.13	0.56	0.61	0.16	0.46	0.41	0.19	-0.51	-0.11	-0.51	-0.26	-0.08	-0.13	-0.27	0.18
RoulX	0.28	0.02	0.44	0.55	0.14	0.45	0.31	0.18	-0.44	-0.02	-0.51	-0.23	-0.01	-0.11	-0.22	0.20
Soft	0.11	0.06	0.40	0.67	0.18	0.49	0.32	0.22	-0.13	0.01	-0.41	-0.46	-0.22	-0.26	-0.25	0.16
μ	0.44	0.11	0.47	0.65	0.18	0.47	0.39	0.18	-0.42	-0.07	-0.46	-0.43	-0.17	-0.30	-0.31	0.14
σ	0.32	0.16	0.10	0.11	0.13	0.11	0.15	0.08	0.26	0.11	0.09	0.15	0.12	0.14	0.15	0.06

ing zero and near-zero values indicate that different trigger configurations yielded little to no significant changes in performance. The narrow distributions indicate that this behavior was consistent across environments. Table 5.3 corroborates these visual findings by showing how **ETour2**, **ETour25**, and **DProp** consistently had near-zero average wins minus losses across most *island* topology configurations.

There was a strong visual trend among the trigger configurations for most hyper-

heuristics where rank values tended to increase as values of k increased. The trend is more visibly prevalent in the *global* topology results than in the *island* topology results. The implication is that higher values of k yielded worse Friedman rank values for a trigger configuration (i.e., worse performance). This observation is validated by the correlation coefficients in table 5.4, which shows that the overwhelming majority of hyper-heuristic configurations had a positive correlation between Friedman ranks and k. Negative correlations were found in isolated cases only, and were mostly close to zero (the exception was **DProp** using *Global_RT* which had a stronger negative correlation coefficient of -0.17). These results were consistent with the findings of the previous research question in section 5.3.2 that examined each type of trigger separately.

Similarly, there was a strong negative association between increasing values for k and lower wins minus losses values for each hyper-heuristic configuration. This inverse relationship shows that higher wins minus losses values occurred at lower values for k, which is visible in the plots as positive values at low values for k, and progressively smaller (even negative) values at larger values for k. Table 5.4 corroborates these visual findings with strong negative correlation coefficients for nearly every hyper-heuristic configuration. Isolated exceptions exist, such as **DProp** using the *Island_RT* configuration with a 0.22 correlation value, and **Perm** using the *Global_PT* configuration at 0.18. These results show that statistically significant increases in performance resulted for most hyper-heuristics by using lower values for k.

Was the performance of any hyper-heuristics sensitive or insensitive to trigger changes?

Generally, a hyper-heuristic was insensitive to a change in the type of trigger if, for each commensurate value of k, the three triggers had wins minus losses distributions (across all environments) that mostly overlapped each other. In other words, if a change in trigger yielded comparable levels of performance for the hyper-heuristic, then that hyper-heuristic was not sensitive to the type of trigger. On the other hand, a hyper-heuristic was sensitive to the type of trigger if, for each commensurate value of k, the wins minus losses distributions were non-overlapping. Differences in performance occurred depending on which trigger was used in conjunction with such a hyper-heuristic.

A visual inspection of figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, and 5.15 reveals that not all hyper-heuristics showed the same sensitivity to perform differently with alternate trigger configurations. Hyper-heuristics that were insensitive to the type of trigger tended to show similar-looking rank plots for each trigger type, where the blue, orange, and green groups of ranks had similar distribution sizes and value ranges (across comparable values of k). In these cases, the wins minus losses sub-plots also tended to show relatively uniform patterns across triggers (at comparable values for k). On the other hand, many sensitive hyper-heuristics showed larger differences in rank value distributions across triggers at comparable values of k, with the respective wins minus losses sub-plots displaying large discrepancies between triggers.

A Friedman test was performed on the three wins minus loss distributions across the 27 environments for each trigger (at each specific value for k) to provide insight into whether the performance of the three triggers overlapped, or if the three distributions were, in fact, different. Three-way comparisons were conducted between the PT, ST and RT trigger configurations for each value of k. Independent tests were conducted for the *global* and *island* topologies. Table 5.5 shows the Friedman test p-values for each hyper-heuristic at each value for k and each topology. At a significance level of $\alpha = 0.05$, over half of the table entries show significant differences were present between the wins minus losses value distributions of the three triggers. For the *global* topology, 15 of the 21 hyper-heuristics showed significant differences between triggers at $k = 20\%$ (the *island* topology only showed 9). A number of hyper-heuristics never showed any significant differences in performance between triggers, most notably **Freq2** and **RoulX** when the *global* topology was used, and **AProp**, **ETour2**, **ETour25**, **RoulM**, **RoulX** and (to a degree) **DProp** when the *island* topology was used.

By and large, however, the choice of trigger significantly affected the performance of most hyper-heuristics.

Did ST or RT configurations ever outperform PT configurations?

Visually, the wins minus losses plots for a large proportion of hyper-heuristics show that the ST and RT trigger configurations tended to have higher wins minus loss values than the PT trigger configurations (at comparable values for k). The RT configurations also

Table 5.5: Friedman test p-values of wins minus losses per hyper-heuristic, where comparisons were conducted between PT, ST and RT triggers for each value of k. Independent tests were conducted for the *global* and *island* topologies. Bold values indicate p-values less than $\alpha = 0.05$. The notation X-Y-Z stipulates the best-to-worst ranking of the triggers per hyper-heuristic and value for k using the p-values and $\alpha = 0.05$.

	Global topology					**Island topology**				
HH	k=**20%**	k=**40%**	k=**60%**	k=**80%**	k=**100%**	k=**20%**	k=**40%**	k=**60%**	k=**80%**	k=**100%**
AProp	**0.004**	**0.0356**	0.1001	0.1751	**0.0386**	0.6264	0.1592	0.7781	0.9394	0.0594
ARank	**0.0016**	0.4385	0.6436	0.9394	0.3646	**0.0415**	**0.0427**	**0.0312**	**0.0163**	**0.0057**
Comp	**0.0**	**0.0076**	**0.0153**	**0.0322**	0.0776	0.268	0.0815	**0.0247**	**0.0047**	**0.0086**
DProp	**0.0119**	**0.0032**	**0.0004**	**0.0002**	**0.0**	0.9736	0.8983	0.5771	**0.0415**	0.0589
ETour2	**0.0**	**0.0**	**0.0001**	**0.0001**	**0.001**	0.7994	0.9649	0.9736	0.8903	0.7046
ETour25	**0.0312**	**0.0006**	**0.0**	**0.0**	**0.0**	0.5985	0.7046	0.8287	0.4896	0.1561
Freq2	0.0589	0.3926	0.2241	0.2606	0.0732	**0.0315**	**0.0001**	**0.0112**	**0.001**	**0.0**
Freq5	**0.0**	**0.0001**	**0.0**	**0.0**	**0.0**	**0.0006**	**0.0014**	**0.0103**	**0.0**	**0.0**
HTour2M	0.1264	**0.0322**	**0.0026**	**0.0067**	**0.0042**	**0.0029**	**0.0007**	**0.0076**	**0.0047**	**0.0032**
HTour2X	**0.0063**	**0.0086**	**0.0059**	**0.004**	**0.0155**	**0.0005**	**0.0001**	**0.0436**	**0.0011**	**0.0004**
HTour5M	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0002**	**0.0008**	**0.0004**	**0.0**	**0.0**
HTour5X	**0.0006**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0001**	**0.0011**	**0.0003**	**0.0**
NAProp	**0.0024**	0.0891	0.2606	0.2394	0.1592	0.1686	0.3231	0.4305	0.0746	**0.0053**
NARank	**0.0015**	0.0954	0.3926	0.2372	0.8437	**0.0315**	**0.0356**	0.0644	**0.0078**	**0.0014**
Perm	**0.0009**	0.3612	0.3926	0.7781	0.7046	0.1104	**0.0124**	0.5033	0.1339	0.1751
RLFreq	0.5771	0.2372	**0.0002**	**0.0**	**0.0**	**0.0119**	**0.0002**	**0.0**	**0.0001**	**0.0004**
RLProp	0.3231	0.5033	**0.0002**	**0.0**	**0.0**	0.4036	0.8437	0.3714	0.0891	**0.0303**
Rand	**0.0063**	0.3748	0.1313	0.5667	0.8287	0.7574	0.1193	0.3646	**0.0216**	**0.0014**
RoulM	0.6264	0.1686	**0.041**	**0.0268**	0.0589	0.1592	0.8667	0.5464	0.4807	0.0792
RoulX	0.3084	0.7994	0.5173	0.3646	0.4385	0.146	0.5033	0.5464	0.4385	0.1264
Soft	**0.0312**	**0.0436**	0.3748	0.9394	0.4677	0.3926	**0.0015**	0.0589	**0.0067**	**0.0008**
HH	k=**20%**	k=**40%**	k=**60%**	k=**80%**	k=**100%**	k=**20%**	k=**40%**	k=**60%**	k=**80%**	k=**100%**
AProp	S-R-P	S-R-P	-	-	S-R-P	-	-	-	-	-
ARank	S-R-P	-	-	-	-	R-S-P	R-S-P	R-S-P	R-S-P	R-S-P
Comp	S-R-P	S-R-P	S-R-P	S-R-P	-	-	-	R-S-P	R-S-P	R-S-P
DProp	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	-	-	-	R-P-S	-
ETour2	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	-	-	-	-	-
ETour25	P-R-S	R-P-S	R-P-S	R-P-S	R-S-P	-	-	-	-	-
Freq2	-	-	-	-	-	R-P-S	R-S-P	R-S-P	R-S-P	R-S-P
Freq5	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-S-P	R-S-P	R-P-S	R-S-P	R-S-P
HTour2M	-	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-S-P	R-S-P	R-S-P	R-S-P
HTour2X	R-P-S	R-P-S	R-P-S	R-P-S	R-S-P	R-S-P	R-S-P	R-S-P	R-S-P	R-S-P
HTour5M	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-S-P
HTour5X	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S	R-S-P
NAProp	S-R-P	-	-	-	-	-	-	-	-	S-R-P
NARank	S-P-R	-	-	-	-	R-P-S	R-S-P	-	R-S-P	R-S-P
Perm	S-R-P	-	-	-	-	-	R-S-P	-	-	-
RLFreq	-	-	R-P-S	R-P-S	R-S-P	R-P-S	R-P-S	R-P-S	R-P-S	R-P-S
RLProp	-	-	R-S-P	R-S-P	R-S-P	-	-	-	-	R-S-P
Rand	S-P-R	-	-	-	-	-	-	-	R-S-P	R-S-P
RoulM	-	-	R-P-S	R-S-P	-	-	-	-	-	-
RoulX	-	-	-	-	-	-	-	-	-	-
Soft	S-P-R	S-P-R	-	-	-	-	R-S-P	-	R-S-P	R-S-P

tended to outperform both the PT and ST configurations, which is generally visible as a larger proportion of orange entries above the zero line for the RT configurations than the PT or ST configurations. A general pattern was also visible across many hyper-heuristics where the ST configurations tended to outperform their PT counterparts at low values of k, but had worse performance at higher values of k.

The lower section of table 5.5 confirms the visual findings above. The notation *X-Y-Z* stipulates the best-to-worst ranking of the triggers where significant differences were found in wins minus losses value distributions. The best-to-worst ranking (where significant differences were found to exist) were performed using the mean wins minus losses from table 5.3. For example, the tuple "*R-S-P*" indicates that the RT trigger outperformed both the ST and PT triggers, the ST trigger only outperformed the PT trigger, and the PT trigger did not perform better than any of the other triggers. For the *global* topology at $k = 20\%$, the RT always ranked higher than the PT configurations for all hyper-heuristics except **ETour25**, **NARank**, **Rand**, and **Soft** (when significant differences were found – six hyper-heuristics showed no differences between the results for the triggers). For larger values of k, the RT trigger always outperformed the PT trigger when significant differences were present. The sole exception was **Soft** when using $k = 40\%$. The RT trigger was always the best trigger when the *island* topology was used and significant differences were present.

For the *global* topology, the PT configurations outperformed the ST configurations for those hyper-heuristics that made use of tournament selection (when significant differences were present), the only exceptions being **ETour25** and **HTour2X** at $k = 100\%$. This observation holds for all values for k. All configurations of **DProp** also saw PT triggers outperform ST triggers, for all versions of k. All other hyper-heuristics saw the ST trigger perform on par with or better than the PT trigger at all values of k, except for **RLFreq** at $k = 60\%$ and $k = 80\%$, and **RoulM** at $k = 60\%$. For the *island* topology, the various trigger configurations for hyper-heuristics showed different proclivities to perform better with either ST or PT.

Figure 5.16 graphically shows the information from table 5.3. The plots corroborate the visual findings above by showing how the RT trigger consistently had higher mean wins minus losses values than the PT trigger for nearly every hyper-heuristic. Isolated exceptions are visible where the RT curve drops below the PT curve, most notable **Perm**

Table 5.6: Significant wins per trigger, by hyper-heuristic, using the best-to-worst ranking of triggers in table 5.5.

Trigger	AProp	ARank	Comp	DProp	ETour2	ETour25	Freq2	Freq5	HTour2M	HTour2X	HTour5M	HTour5X	NAProp	NARank	Perm	RLFreq	RLProp	Rand	RoulM	RoulX	Soft
PT_wins (Global)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PT_wins (Island)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RT_wins (Global)	0	0	0	5	5	4	0	5	4	5	5	5	0	0	0	3	3	0	2	0	0
RT_wins (Island)	0	5	3	1	0	0	5	5	5	5	5	5	0	4	1	5	1	2	0	0	3
ST_wins (Global)	3	1	4	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	2
ST_wins (Island)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

when using the *Global_80* or *Global_100* configurations, and **Rand** using the *Global_40* or *Global_60* configurations. The performance of all RT configurations was always the same or better than corresponding PT configurations when the *island* topology was used. These visuals need to be considered in conjunction with the results in table 5.5 to account for significant differences, since the curves only show the averages of each distribution for each hyper-heuristic.

Did any hyper-heuristics perform noticeably better with specific trigger choices?

Table 5.6 summarizes the best-to-worst ranking of triggers in table 5.5 by listing how many configurations each trigger was ranked as being the best, broken down by each hyper-heuristic and each topology. Notable patterns that are visible in tables 5.5 and 5.6 include:

- **Freq2** and **RoulX** showed no significant differences in performance between triggers when using the *global* topology, at any value of k. These two hyper-heuristics were relatively insensitive to changes in triggering mechanisms. When using the *island* topology, **AProp**, **ETour2**, **ETour25**, **RoulM**, and **RoulX** showed no significant differences between triggers.
- Under the *global* topology, **AProp**, **ARank**, **Comp**, **NAProp**, **NARank**, **Perm**, **Rand**, and **Soft** significantly outperformed the other triggers when using the ST trigger. All these hyper-heuristics had the roulette wheel selection strategy in

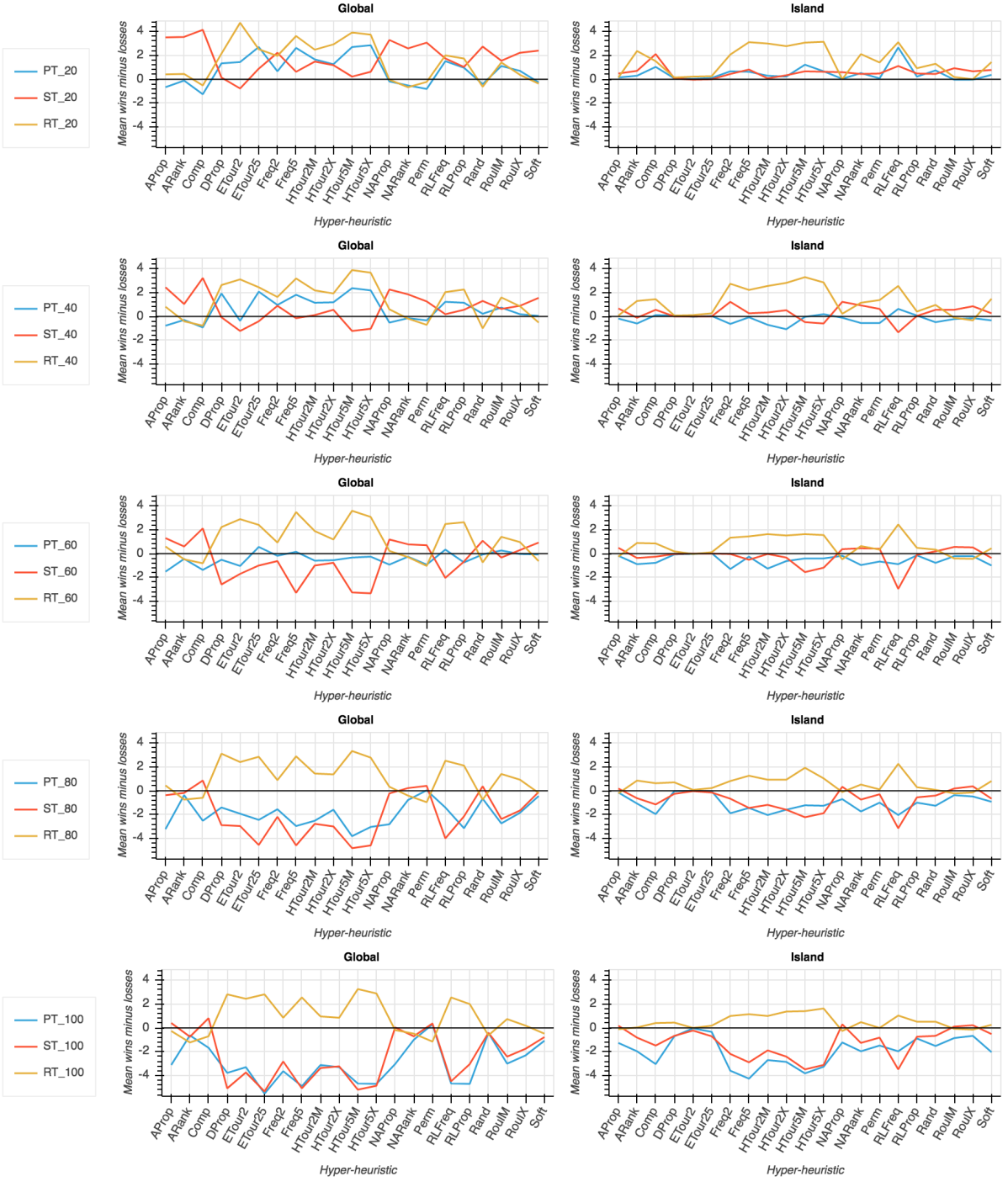


Figure 5.16: Average wins minus losses per hyper-heuristic for each trigger and value of k for both topologies, as shown in table 5.3.

common. Table 5.5 confirms that at least one win of each of the total listed wins for each hyper-heuristic always occurred when k was set to $k = 20\%$. It is clear that the ST trigger is the most viable choice for a number of hyper-heuristics under the *global* topology.

The ST trigger did not perform as well against the other triggers when the *island* topology was used, and the only hyper-heuristic that performed best using the ST trigger was **NAProp**.

- The RT trigger was a strong performer across a large proportion of hyper-heuristics, both when the *island* or *global* topology was used. Under the *global* topology, the RT trigger performed on par with or better than other triggers across those hyper-heuristics that relied on tournament selection. Exceptions were **ETour25**, which lost against PT at $k = 20\%$, and **Freq2**, which never showed any significant differences between any triggers. The remaining tournament selection-based hyper-heuristics, namely **Freq5**, **HTour2M**, **HTour2X**, **HTour5M**, and **HTour5X** performed best with the RT trigger, at all values for k. In contrast, **RLFreq** and **RLProp** performed best using the RT trigger only at $k = 60\%$, $k = 80\%$, and $k = 100\%$ (with no significant differences between triggers at lower values for k). **DProp** also always performed best when using an RT trigger, while **RoulM** showed significant improvements using RT only when $k = 60\%$ and $k = 80\%$.

  When using the *island* topology, the RT trigger was always the best trigger choice across all values for k for **Freq2**, **Freq5**, **HTour2M**, **HTour2X**, **HTour5M**, and **HTour5X** (all tournament selection-based methods), as well as for **ARank** and **RLFreq**. **Perm** only performed best using the RT trigger at $k = 40\%$. **Comp**, **Dprop**, **Rand**, and **Soft** showed no significant differences at low values for k, but performed best using the RT trigger at higher values for $k = 80\%$ and/or $k = 100\%$.

- Under both the *island* or *global* topology, the PT trigger configurations were always matched or outperformed by configurations that used either the RT trigger or the ST trigger for all hyper-heuristics except **ETour25** (when using the *global* topology). For the remaining hyper-heuristics, one of the other triggers was always a better choice, regardless of k or topology.

Were any trigger configurations sensitive to different DOP types?

For any given trigger and value of k, a wide distribution of wins minus losses values (across the 27 environments) indicates that the performance of a hyper-heuristic was influenced by which type of DOP was being solved. The vertical range of the cells in the heat maps combined with cell intensity, as visible in figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 and 5.15, indicate the sensitivity of performance for each hyper-heuristic to different environments. Narrower distributions indicate configurations where performance was invariant across DOPs, while wider spreads indicate volatility based on changing DOPs. Statistically, this sensitivity of a hyper-heuristic to different DOPs can be expressed using the standard deviation across the distribution of wins minus losses values (per trigger and value for k).

Figures 5.17 and 5.18 show the range of wins minus losses values across every trigger configuration for every hyper-heuristic, for both for the *island* and *global* topology, respectively. The red curve in each plot reports the mean wins minus losses value for each hyper-heuristic, as reported in table 5.3. The shaded part of each plot shows the area covering twice the standard deviation of the wins minus losses values. The shaded area therefore represents approximately 95% of the distribution of values [162]. The various hyper-heuristics show different likelihood ranges for wins minus losses values, visible as expansions and contractions of the shaded area across the x-axis. Different topology, trigger, and values for k also impact the size of the distributions, which are shown as different plot panels in figures 5.17 and 5.18.

Under the *global* topology, the PT trigger configurations had narrower distributions than the RT or ST trigger configurations when $k = 20\%$, $k = 40\%$, and $k = 60\%$. The distributions became wider as the value of k increased to $k = 80\%$ and $k = 100\%$. The bulk of the shaded area lies above the zero line when $k = 20\%$ and $k = 40\%$, but as the value of k increases the majority of the area falls below the zero line. The wider distributions and low-valued shaded area indicate that higher values of k were detrimental to the performance of PT triggers. The same trends are visible when the *island* topology was used, except that the distributions for each value of k were much narrower compared to their *global* topology counterparts. Another notable deviation from the *global* topology plots is the fact that **DProp**, **ETour2**, and **ETour25** always

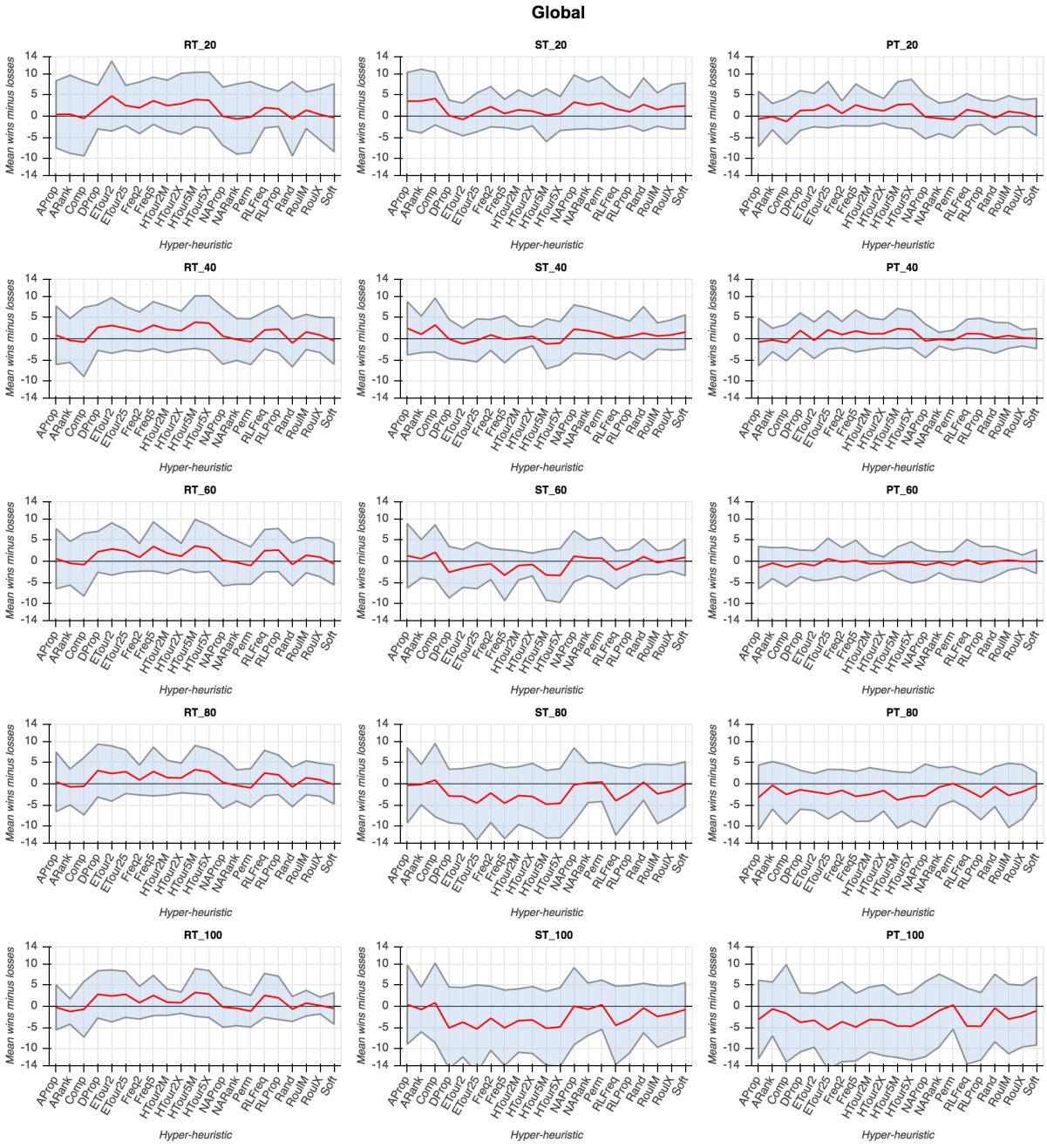


Figure 5.17: Range of the wins minus losses per hyper-heuristic for each trigger and value of k for the *global* topology. The middle red curve represents the mean wins minus losses value for each hyper-heuristic across all 27 environments. The upper and lower curves, respectively, represent double the standard deviation above and below the mean.

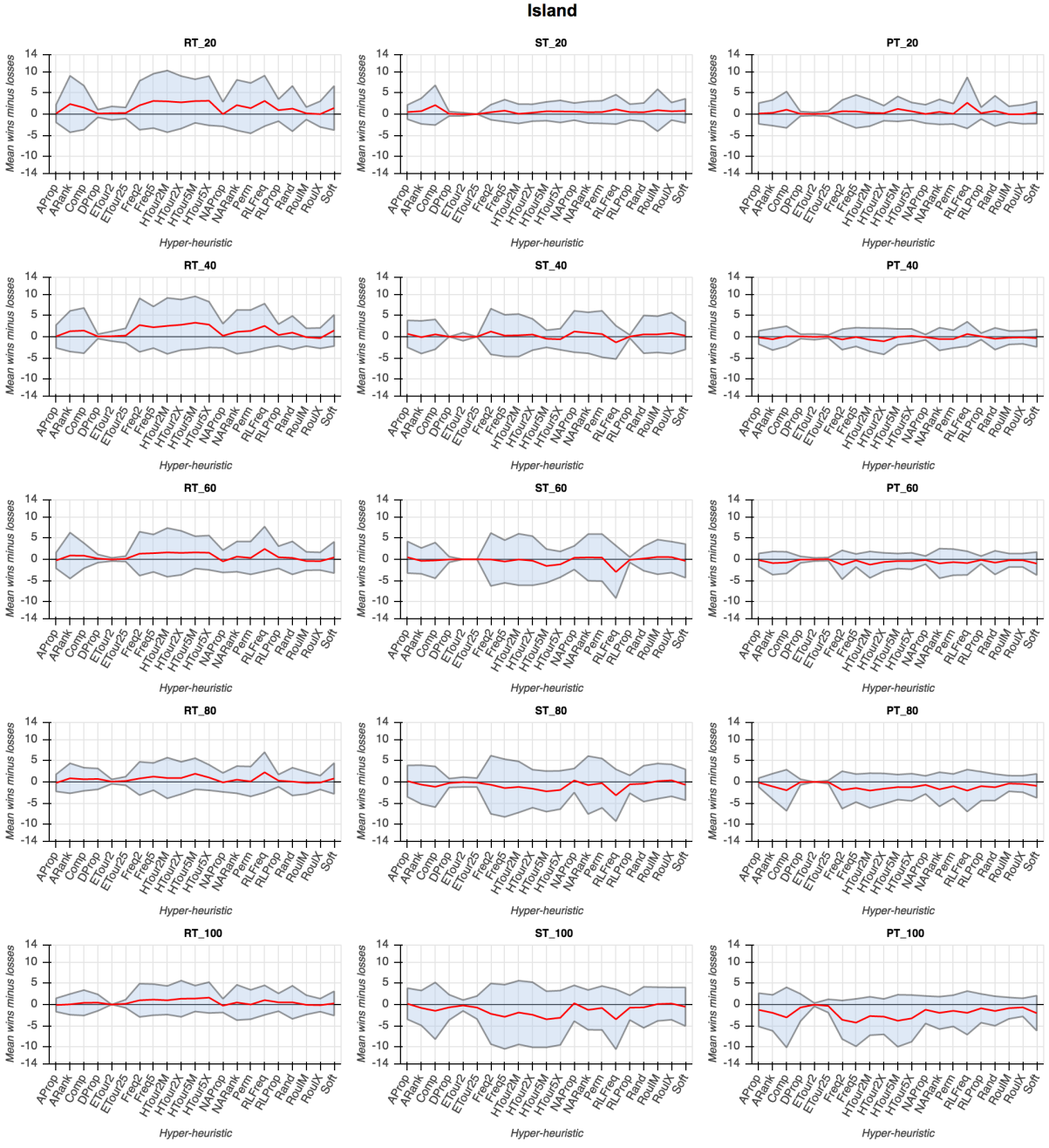


Figure 5.18: Range of the wins minus losses per hyper-heuristic for each trigger and value of k for the *island* topology. The middle red curve represents the mean wins minus losses value for each hyper-heuristic across all 27 environments. The upper and lower curves, respectively, represent double the standard deviation above and below the mean.

showed a net wins minus loss value of roughly zero when using the *island* topology at nearly every value for k, regardless of the environment.

The ST trigger showed the same tendencies as the PT trigger for both topologies, where progressively more of the shaded area fell below the zero line and the width of the distribution increased as values of k increased. Beyond $k = 40\%$ the red mean curve for the majority of hyper-heuristics indicates that net losses occurred. **DProp**, **ETour2**, and **ETour25** showed practically the same behavior as for the PT trigger when the *island* topology was used.

The RT trigger showed the opposite behavior to the ST and PT triggers, in that the width of distributions got smaller as values for k increased. The red mean curve remained relatively stable and on the positive side of the zero line as k varied. Overall, the bulk of the shaded area remained above the zero line as well, which indicates that the RT trigger yielded positive wins minus loss values more frequently than net losses.

Under the *global* topology and for $k = 20\%$, the PT trigger yielded the most predictable range of performance, regardless of the environment type or hyper-heuristic. This is visible in the plot as the shaded area being the thinnest and least jittery across the x-axis. Table 5.3 corroborates this finding by showing how the overall deviation of the averages across all hyper-heuristics for the PT trigger was the lowest of all three triggers. The ST trigger showed relatively similar behavior, but with slightly wider distributions than the PT trigger. The RT trigger had the highest mean wins minus loss values of the three triggers, but the widest shaded area, which indicates high volatility in performance across different environments. However, most of the shaded area for the RT trigger lies above the zero line for those hyper-heuristics located between **DProp** and **HTour5X** in the plot, as well as **RLFreq** and **RLProp**. On the other hand, the negative portion of the shaded area for the RT trigger configurations of these hyper-heuristics were at similar levels as for the PT or ST triggers configurations. This illustrates how the highlighted hyper-heuristics were more likely to perform the same as or better than the PT or ST triggers.

Under the *island* topology and for $k = 20\%$, similar observations hold in that the PT and ST had the lowest deviation in performance across environments and hyper-heuristics (the exceptions being **DProp**, **ETour2**, and **ETour25**). The RT trigger also had the widest distributions of the three triggers, and the majority of the shaded area

was in the positive part of the plot. However, the negative shaded area of the ST and PT triggers were smaller than the RT trigger equivalents, so the likelihood that an ST or PT configuration could perform better than the corresponding RT configuration was higher than under the *global* topology.

Overall, good performance across multiple types of environments was influenced by the choice of trigger and value of k.

Was any one trigger always the best performer across all hyper-heuristics?

The trigger mechanisms proposed in this thesis (i.e., ST and RT) performed well compared to the original HMHH trigger (i.e., PT). Table 5.5 shows that either S or R was ranked higher than P in nearly every cell except **ETour25** at $k = 20\%$ using the *global* topology. Table 5.6 confirms that the PT trigger recorded significantly better wins against the RT or ST triggers in just that one highlighted configuration.

The RT trigger was the outright best choice in six hyper-heuristics under the *global* topology, with good performance in another five hyper-heuristics. Under the *island topology*, the RT trigger held complete dominance over other trigger for eight hyper-heuristics, with another seven hyper-heuristics showing configurations with good performance relative to the other triggers.

The ST trigger tended to be superior to both the PT and RT triggers for eight hyper-heuristics when $k = 20\%$ under the *global* topology. At higher values for k the ST trigger performed worse than PT for nearly every hyper-heuristic, the only exceptions being **AProp**, **Comp**, and **RLProp**, and isolated configurations for **ETour25**, **RLFreq**, and **RoulM**. Under the *island* topology when $k = 20\%$, the ST trigger only outperformed the PT trigger when using **ARank**, **Freq5**, or **HTour2X**. The PT trigger outperformed the ST trigger in six of the hyper-heuristics. At higher values for k the ST trigger outperformed the corresponding PT trigger configurations for most hyper-heuristics, with isolated cases where the PT trigger performed better (notably **RLFreq**, **HTour5X**, **HTour5M**, and **DProp**).

Overall, table 5.6 clearly shows that the RT trigger performed either on par with or better than the other triggers for the majority of hyper-heuristic configurations (apart from **AProp**, **ARank**, **Comp**, **NAProp**, **NARank**, **Perm**, **Rand**, and **Soft** where

ST was better), and especially when $k = 20\%$.

It is clear that the type of trigger had a profound impact on the performance of most of the investigated hyper-heuristics. Almost every hyper-heuristic, for both the *island* and *global* topologies alike, showed a pattern where either all three or two of the three trigger types displayed significantly different performance. Most triggers showed their best performance at lower values for k, and the RT trigger frequently performed on par with or better than the other triggers.

5.3.4 Research Question 3

When was the global or island neighborhood topology significantly better?

Consider two sets of samples of runs of two versions of the same hyper-heuristic, where both algorithms have been configured with the exact same context. Specifically, the same triggers are configured with the same values for k, the same heuristic pool is used by hyper-heuristics, the same set of DOP problem instances are solved, the same entity starting positions are used, and even the same stream of pseudo-randomness is employed. Both algorithms are identical other than that they differ in only one design choice: the one algorithm uses a *global* neighborhood topology while the other uses an *island* topology. The question is, all other factors being equal, when (and if) either one of the *global* or *island* topologies is ever significantly better than the other.

A comparison of the (significant) differences in performance between the sets of results of the two algorithms indicates if there are noteworthy differences between the algorithms as a result of swapping the neighborhood topology configurations. If a non-parametric test shows no significant difference between the results, then both topologies yield similar performance. Alternatively, the choice of topology has a direct impact on the performance of the hyper-heuristic.

The experiment outlined in section 5.2.1 described all possible combinations of 21 triggered hyper-heuristics and 15 different trigger configurations using either the *island* or *global* topology. Each configuration produced 71 run samples for each of the 27 different DOP types. The result was $27 \times 15 \times 21 = 8505$ individual run sample sets for the *island* or *global* topology, respectively to yield a total result set of 17010 sample sets.

The total number of algorithm runs for each topology was $8505 \times 71 = 603855$, which aligns with the experiment totals described in figure 4.3.

A Wilcoxon signed ranks test, as explained in 2.7.4, was used to individually compare each of the 8505 matched pairs of *island* and *global* samples. The goal was to determine if there was a significant difference between each pair of samples at a significance level of $\alpha = 0.05$. If a significant difference was detected between the *island* and *global* results in an individual test of one pair of samples, the topology with the highest number of pairwise wins was awarded a win. If no significant difference was detected then both methods received a draw. Figure 5.19 illustrates how the Wilcoxon tests were conducted.

The 8505 pairwise comparisons yielded an overall count of draws, wins for *global*, and wins for *island*. The aggregated results across all environments, hyper-heuristics, and trigger configurations were as follows:

	Count	**Percent**
Draws	5766	67.8%
Global wins	1839	21.6%
Island wins	900	10.6%

The table above shows that roughly two out of every three overall comparisons did not yield statistically significant differences between the *island* and *global* topologies. In the cases where significant differences did exist, the *global* topology performed better twice as often. The numbers in the table can, however, be broken down further as follows: figure 5.20 shows the draws and wins aggregated per hyper-heuristic, figure 5.21 shows the results aggregated per trigger type (and k value), and figure 5.22 shows the results broken down by individual environment. Figure 5.23 shows a heat map of the difference between the wins of the *global* and *island* topologies per hyper-heuristic, DOP type, and trigger type for the cases where significant differences exist (draws are not represented in the heat map). Positive values in the heat map indicate that the *global* topology was superior, zero values indicate the same number of wins for both topologies, and negative values indicate that the *island* topology was superior.

The following observations, with regard to the stated research question, can be made about the *island* and *global* topologies:

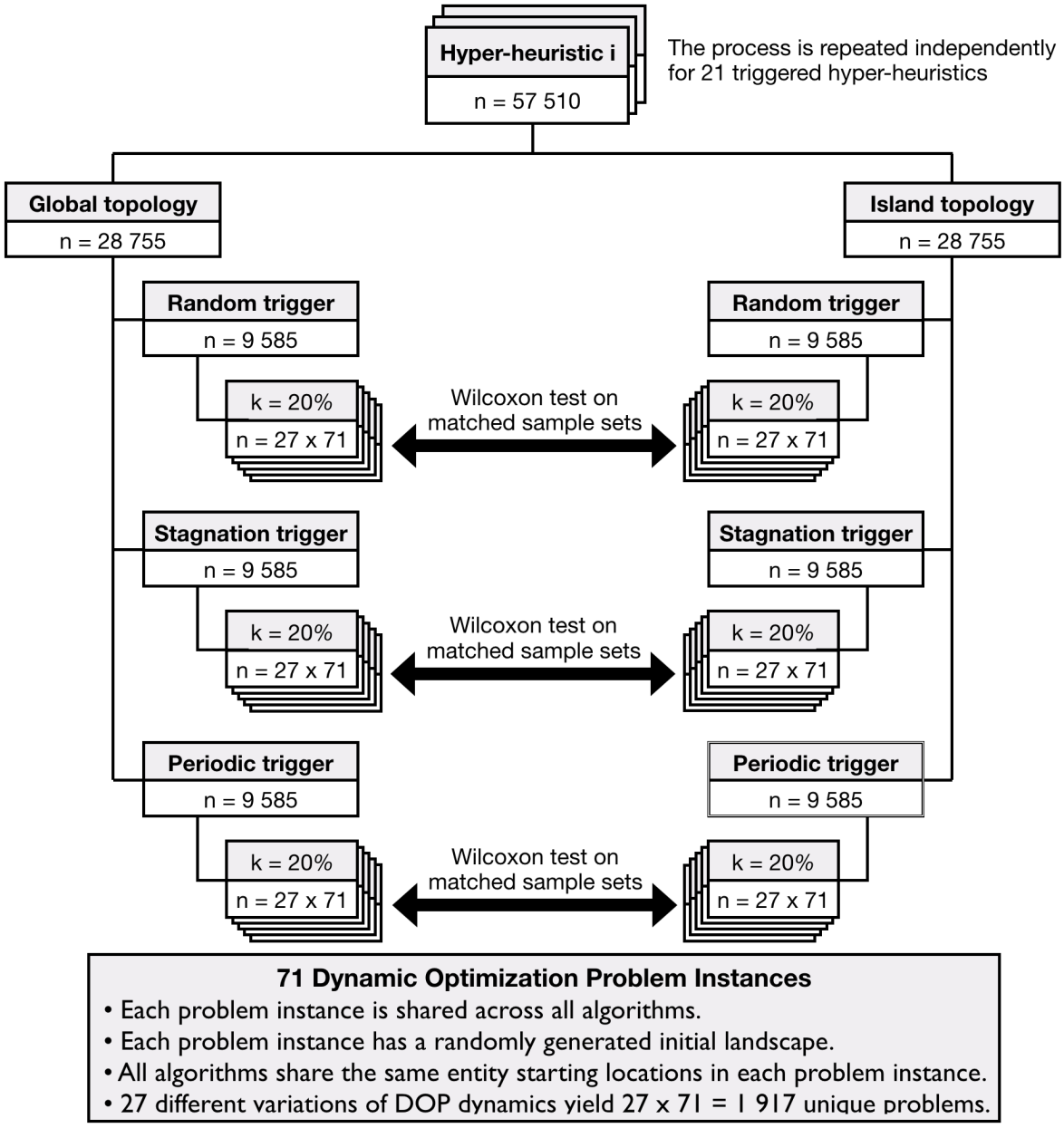


Figure 5.19: Focus of the Wilcoxon signed ranks tests for research question 3. The process was repeated for each triggered hyper-heuristic, each trigger type and the corresponding values for k, and the comparison was made between the *global* and the *island* topologies variations. The value n represents the number of algorithm runs underpinning each level.

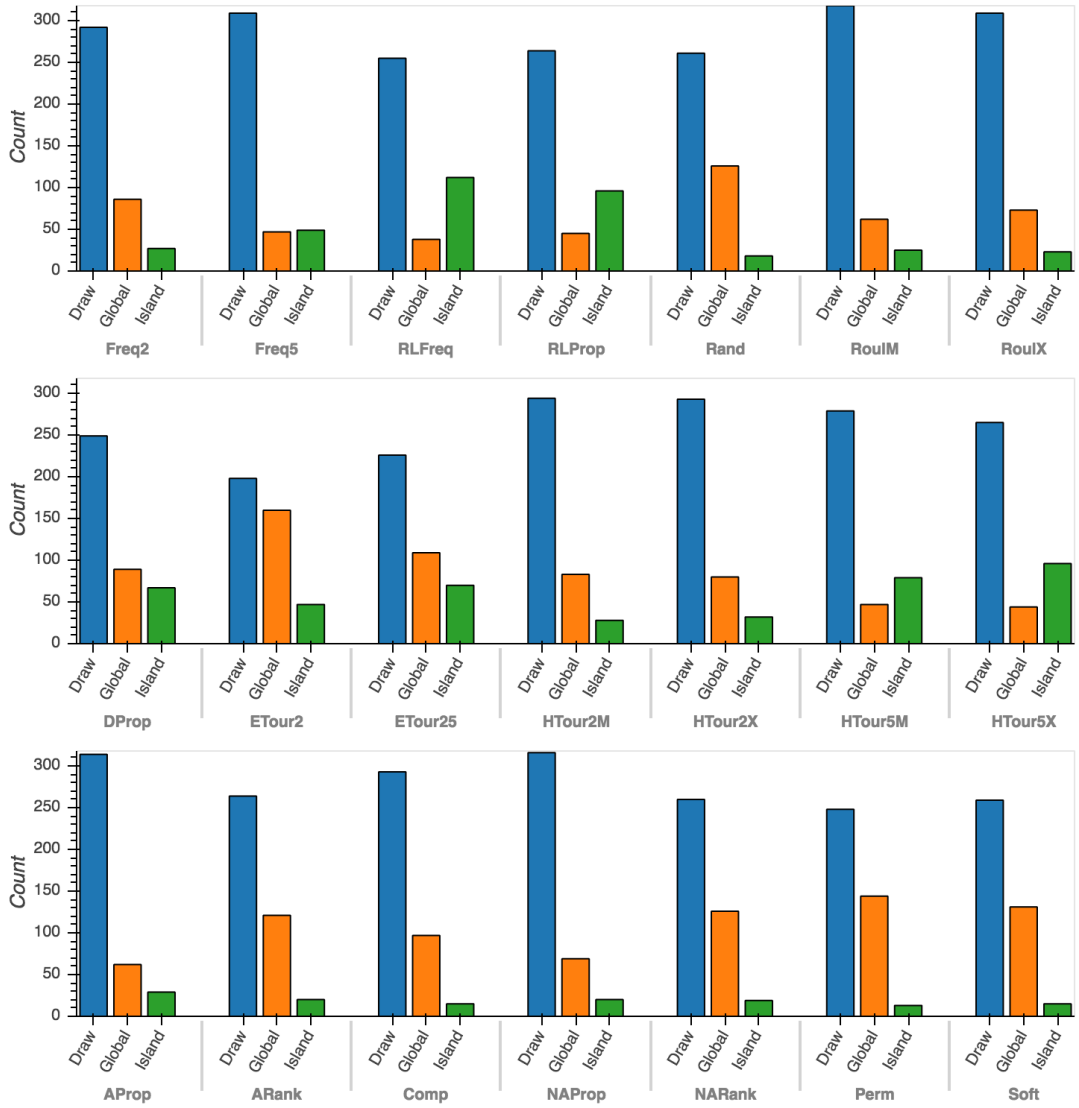


Figure 5.20: Pairwise wins and draws of the *island* versus the *global* topology by hyper-heuristics (across all DOPs and trigger configurations.)

- *Which topology was better across different hyper-heuristics?* Figure 5.20 shows that the *global* topology generally yielded more significant wins than the *island* topology for the majority of hyper-heuristics, the only exceptions being **RLFreq**, **RLProp**, **HTour5M**, and **HTour5X** and (to a smaller extent) **Freq5**. Figure 5.23 confirms these findings by revealing that the majority of cells were either white (zero values) or shades of blue (leaning towards *global* having more wins).

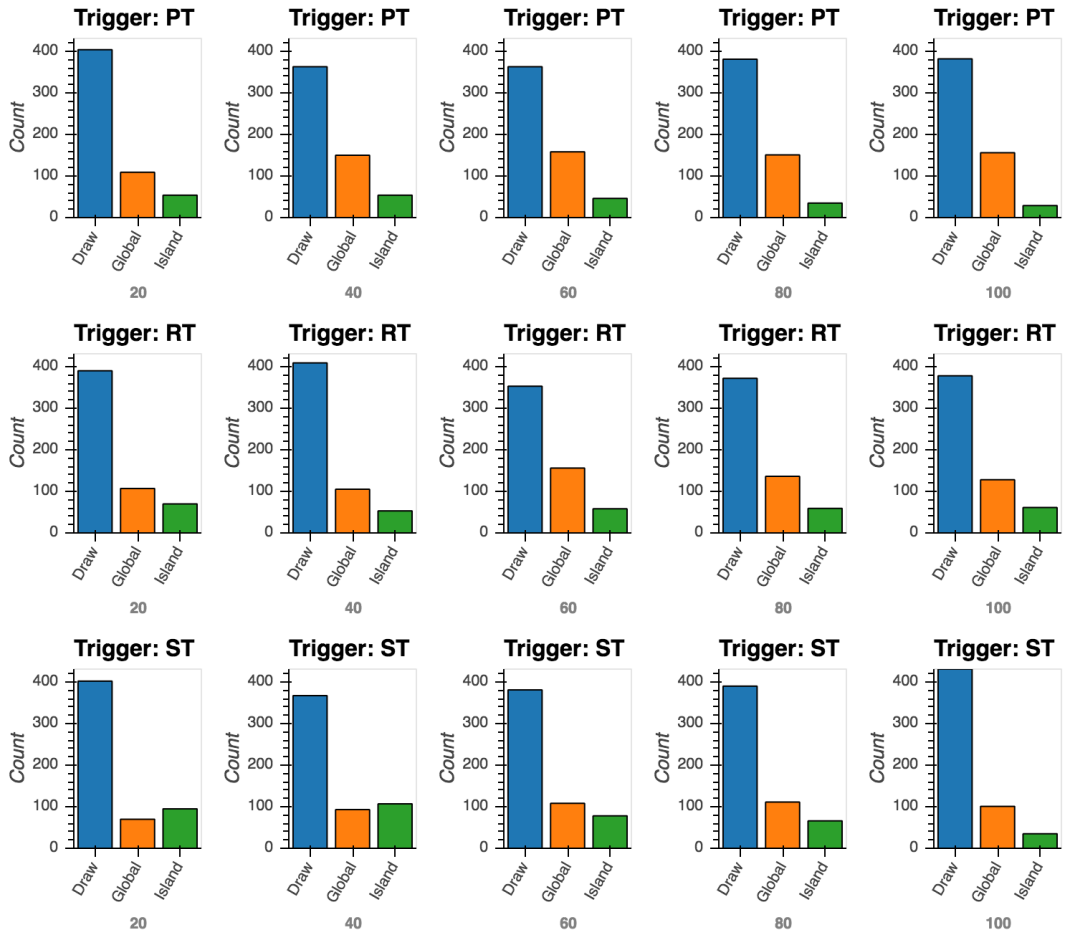


Figure 5.21: Pairwise wins, draws, and losses of the *island* versus the *global* topology in each trigger configuration (across all hyper-heuristics and DOPs.)

The columns for **RLFreq**, **RLProp**, **HTour5M**, and **HTour5X** show a large number of red values in all three heat maps across all three trigger types, which indicates that these hyper-heuristics showed a strong tendency to perform better when using the *island* topology.

Figure 5.20 shows that the various hyper-heuristics had different sensitivity profiles for the *island* and *global* topologies. **ETour2** is an example where the number of draws was approximately equal to the number of wins for the *global* topology, while the *island* topology had substantially fewer wins. In this case, the *global* topology was clearly a better choice, with comparable or better performance than the *island*

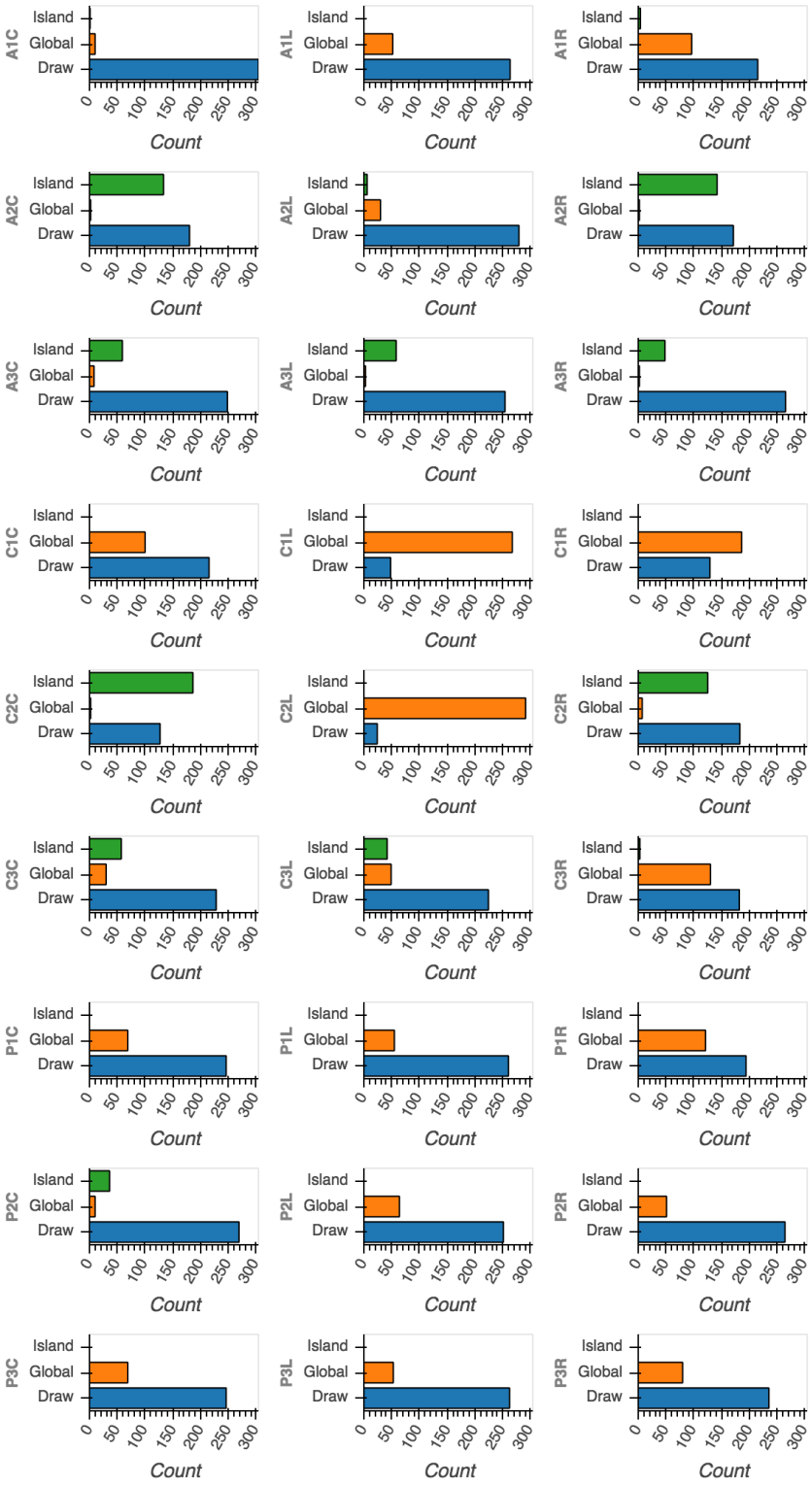


Figure 5.22: Pairwise wins, draws, and losses of the *island* versus the *global* topology in each environment (across all hyper-heuristics and trigger configurations.)

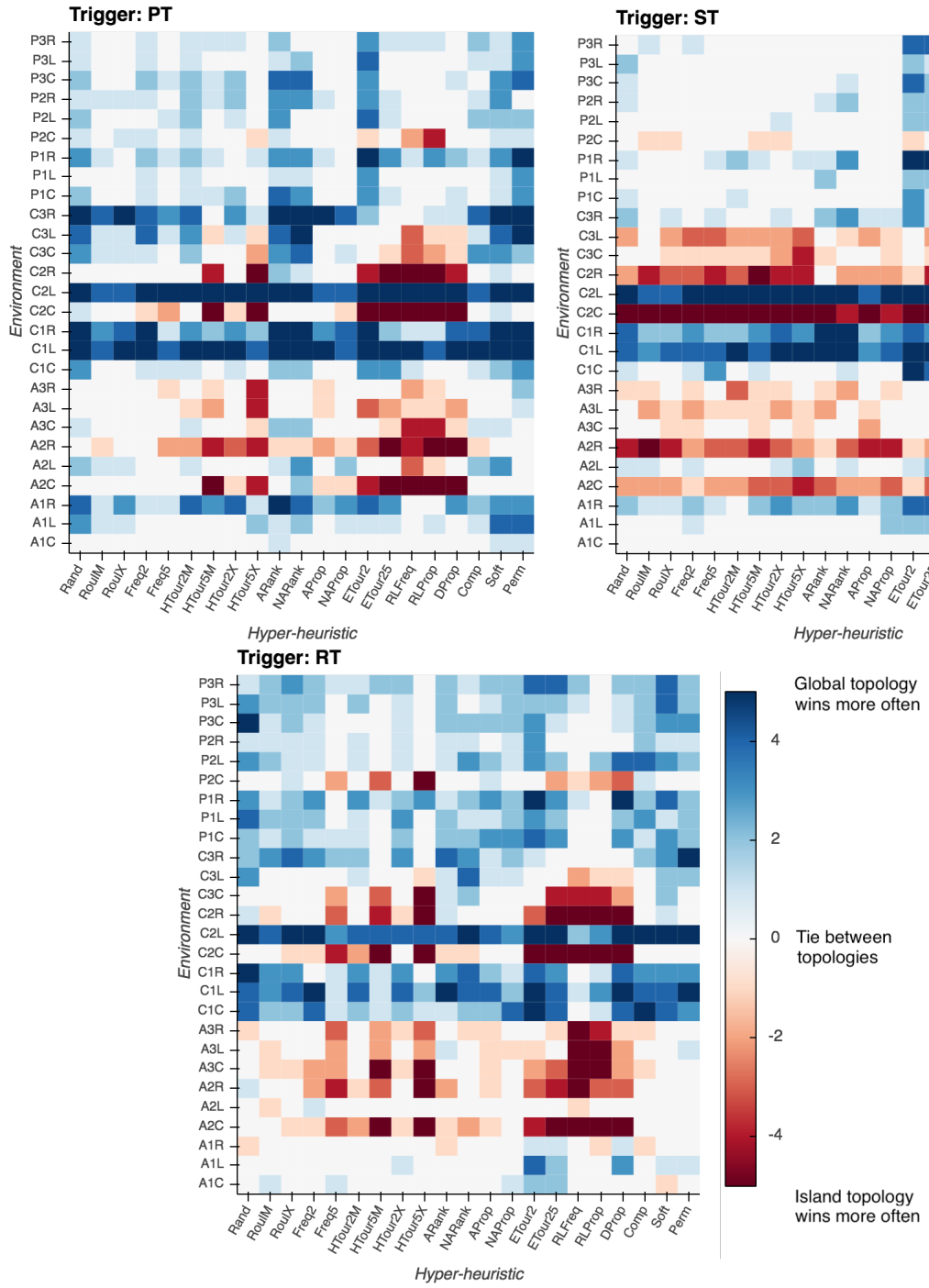


Figure 5.23: Heat map of the difference in pairwise wins between the *island* or *global* topology achieved by each hyper-heuristic in each environment, shown per trigger type. Positive numbers (blue) indicate that the *global* topology wins more times than the *island* topology and negative numbers (red) indicate the opposite.

topology. Similar trends can be seen in **Rand**, **ARank**, **NARank**, **Perm**, and **Soft** and (to a lesser degree) in **ETour25**, **Comp**, **HTour2M**, and **HTour2X**.

Another example is **Freq5**, which had a large proportion of draws and an equal number of wins for the *island* and *global* topologies. The fact that both topologies had the same (and non-zero) number of wins suggests that good performance for **Freq5** may depend on the specific DOP being solved. Figure 5.23 corroborates this interpretation by showing how the best topology for **Freq5** tended to vary across different environments (illustrated by the various shades of red, blue, and white).

- *Which topology was better across different triggers?* Figure 5.21 displays a clear trend that the *global* topology was often a better choice for the PT trigger, across all values of k. The same is true for the RT trigger, although the difference was less pronounced. In contrast, the ST trigger showed similar wins for both topologies, with the *global* topology eventually becoming superior as the value of k increased.

  The heat maps in figure 5.23 reinforce the observations above:

  - The PT trigger performed best in the *global* topology (i.e. blues), where almost all *island* topology wins (red values) were restricted to the intersection of just four environments (i.e. C2R, C2C, A2R, and A2C) with seven hyper-heuristics (i.e. **HTour5M**, **HTour5X**, **ETour2**, **ETour25**, **RLFreq**, **RLProp**, and **DProp**).
  - The RT trigger results were similar to the PT trigger, where similar intersections of environments and hyper-heuristics performed better with the *island* topology. **Freq5** combined with RT showed a noticeable deviation from **Freq5** that used PT.
  - The ST trigger heat map shows that most of the abrupt environments favored the *island* topology (i.e. red shades). Four chaotic environments (i.e. C3L, C3C, C2R, and C2C) performed noticeably better under the *island* than the *global* topology, while the exact opposite trend was visible for C2L, C1R, C1L. The ST trigger was sensitive to the which topology was better suited for a

given environment, as indicated by many rows being nearly all red, all blue, or all white.

- *Which topology was better across different environment types?* Figure 5.22 shows that the efficacy of both the *island* and *global* topologies was impacted by the type of environment. Abrupt and progressive environments had a high number of draws. However, the cases that were not draws show a clear split between environments: progressive environments tended to perform better with the *global* topology (visually visible as predominantly orange bars) and abrupt environments mostly favored the *island* topology (indicated by predominantly green bars). Chaotic environments showed high volatility: the *global* topology was a clear winner for C1C, C1L, C2L, C1R, and C3R, while the *island* topology dominated for C2C and C2R.

  The heat maps in figure 5.23 echo these patterns: progressive environments have mostly white and blue patterns across rows, indicating a preference for the *global* topology. On the other hand, abrupt environments show a high number of white and red cells, indicating a tendency that the *island* topology performed the same or better than the *global* topology. Chaotic environments show both strong blue and strong red bands, indicating that good performance in these environments was highly dependent on the choice of topology.

  At first glance, each heat map in figure 5.23 looks similar. Closer inspection reveals that the PT trigger had 12 out of 21 hyper-heuristics where the *global* topology dominated across all 27 environments. All of the hyper-heuristics that relied on roulette wheel selection showed this pattern. On the other hand, eight hyper-heuristics showed high volatility across different environments where the *island* topology was clearly superior. The common trend among the strongest four examples is that they all contained type II[1] environment dynamics and that most of the hyper-heuristics employed tournament selection. The RT trigger showed similar trends, but with abrupt environments leaning more towards the *island* topology (indicated by more light shades of red and less deep hues of blue).

  The ST trigger showed completely different behavior than the RT and PT triggers.

[1] Per Eberhart's DOP classification as presented in section 2.3.1

Each environment always (without exception) performed better with just one of the topologies, as indicated by all rows having the same color. The broadest observable pattern is that almost all progressive environments showed a tendency to perform better in the *global* topology while most abrupt environments performed better in the *island* topology. Chaotic environments were volatile and individual environment types strongly excel in either the one or the other topology. There was not a single hyper-heuristic that clearly performed better using one or the other topology, and performance was strictly dependent on the specific type of DOP being solved.

It is clear that the choice of neighborhood topology, all other factors being equal, had a distinct and noticeable influence on hyper-heuristic performance. Overall, the *island* and *global* topologies showed insignificant differences in roughly two thirds of the comparisons. If significant differences in performance were detected, then the *global* topology tended to perform better twice as often as the *island* topology.

That said, the difference in performance depended greatly on the combination of the type of DOP and hyper-heuristic. By excluding just four of the 27 environments (i.e. C2R, C2C, A2R, and A2C) or five of the hyper-heuristics (i.e. **ETour2**, **ETour25**, **RLFreq**, **RLProp**, and **DProp**), the *global* topology would have received an overwhelming majority of wins. This can be seen visually in figure 5.23: excluding the identified environments and hyper-heuristics would have eliminated the majority of red in the heat maps.

5.4 Summary

The original HMHH framework uses a *periodic* trigger to trigger changes every k algorithm iterations. Clear statistically significant differences in performance were found between different types of heuristic change triggers. The research questions in this chapter clearly showed that the *stagnation* and *random* triggers regularly outperformed the *periodic* trigger more often than not across different hyper-heuristics and across multiple types of DOPs.

The parameter that controls the frequency of heuristic change, namely k, had a

strong impact on the performance of each trigger. Most hyper-heuristics showed better performance at lower values for k rather than at higher values. This trend was noticeable across all three triggers, across all hyper-heuristics, and across both types of the HMHH framework neighborhood topologies.

The choice between the *island* versus *global* topology had a strong influence on the trade-offs between the best triggers and the best values of k. Different values of k showed higher volatility in both rank and win distribution patterns across most hyper-heuristics. The general trend across most hyper-heuristics, however, showed that lower values of k tended to yield better performance with smaller performance variance than larger values of k. The *global* topology tended to be a stable choice when using the PT and RT triggers when combined with any hyper-heuristic that relied on roulette wheel selection, regardless of the problem being solved. The ST trigger on the other hand was extremely sensitive to the choice of topology depending on the environment that was being solved, regardless of hyper-heuristic.

A fundamental difference between the original *periodic* trigger and the newly proposed *stagnation* and *random* triggers is the manner in which entities were triggered. All entities were triggered simultaneously when the *periodic* trigger was used (i.e. every k iterations), which forced the HMHH selection to potentially reassign the heuristics of all entities at the same time. The proposed triggers considered each entity individually, and 100% of the population was not forced to undergo heuristic selection together. This chapter shows that the newly proposed trigger types clearly influenced hyper-heuristic performance positively in most cases, possibly due to greater stability in heuristic sub-populations along with a more regular stream of incoming information.

The availability of various types of heuristic selection triggers makes the HMHH framework more flexible and allows richness in how heuristic selection is performed. The next chapter compares the performance and behavior of various hyper-heuristics against each other and against various control groups.

Chapter 6

Performance and Behavior Analysis of Selection Hyper-heuristics

> *"It is not a question of how well each process works, the question is how well they all work together."*
> – *Lloyd Dobens*

This chapter compares and analyzes the performance of various HMHH selection operators against each other. The heuristic space diversity and entity reassignment behavior of a variety of exemplary HMHH selection operators are investigated to characterize their behavior and performance.

6.1 Introduction

Chapter 5 examined how distinct combinations of different HMHH neighborhood topologies and heuristic change triggers (with varying triggering frequencies k) affected the performance of HMHH selection operators. The term *hyper-heuristic* will be used to refer to a HMHH selection operator in this chapter. Different research questions in section 5.3.1 systematically examined the sensitivity of a number of hyper-heuristics to variation across different trigger and topology choices. The investigation found that statistically significant differences in performance exist for any hyper-heuristic, given different neighborhood topology and trigger choices. Specifically, the majority of the in-

vestigated hyper-heuristics (for all topology and trigger configurations) generally tended to yield better performance at lower values of $k \approx 20\%$, i.e., when the hyper-heuristic was allowed to modify entity-to-heuristic assignments more often.

However, the performance comparisons of hyper-heuristics in chapter 5 were always inwardly focused, that is, each hyper-heuristic was only compared to different versions of itself using alternative trigger and neighborhood topology configurations. As such, no conclusions could be drawn about which hyper-heuristics outperformed each other, or if any hyper-heuristic performed better than any of the control methods. In this chapter the performance and behavior of various hyper-heuristic choices for HMHH are compared. The aim is to

- determine whether each hyper-heuristic raises performance above that of the individual heuristics or speciated heuristics,
- establish whether each hyper-heuristic performs better or worse than the fixed or random selection control methods,
- understand which hyper-heuristics perform better than other hyper-heuristics,
- learn more about how different hyper-heuristics operate by examining the heuristic space diversity and entity reassignment rates of select hyper-heuristics.

Section 6.2 describes the experimental procedure, analysis techniques, and measures used to explore the goals above. The results are organized into distinct research questions that are presented in section 6.3. Section 6.4 concludes the chapter.

6.2 Experimental Procedure

Chapter 5 compared different architectural variations of each hyper-heuristic against themselves. The experiments in this chapter directly compare the performance of different hyper-heuristics (all sharing the same architectural configurations) against each other. Additionally, each hyper-heuristic is compared against both fixed and random heuristic selection, so as to provide comparisons against a common baseline.

To make this analysis tractable, the focus of this chapter is on a smaller representative subset of hyper-heuristic configurations. Specifically, the heuristic change frequency

parameter of each trigger, k, is set to $k = 20\%$, i.e., 20% of the landscape change period length. The value of $k = 20\%$ is used based on the findings in chapter 5 that showed how lower values of k performed better in general across most hyper-heuristics, trigger types, and topologies.

Section 6.2.1 summarizes the experimental method as presented in section 5.2.1, and discusses any additional experimental design decisions.

6.2.1 Experimental Method

The same experimental procedure as described in section 4.3 and section 5.2.1 is used in the experiments in this chapter. The following points of clarification supersede the prior design decisions:

- **Heuristic algorithm and parameter value choices**: Section 4.3.2 outlined a selection of representative meta-heuristics from the SI and EC fields that make up a pool of nine heuristics. Suitable parameter choices were motivated for each heuristic. The same algorithms, parameter values, and implementation decisions for each heuristic are used in all experiments in this chapter. The parameter decisions for any given heuristic algorithm are the same across the heuristic pools of all hyper-heuristics as well as for the individual and speciated heuristic control groups.

- **Hyper-heuristic choices**: The same hyper-heuristic design decisions and parameter values are used as outlined in section 5.2.2. The parameters values for each hyper-heuristic are kept the same as was outlined in table 5.1. Each hyper-heuristic is set up to use both the *island* and *global* neighborhood topologies, respectively. The three heuristic change triggers outlined in section 3.4.4, namely a *periodic* trigger (PT), *stagnation* trigger (ST), and a *random* trigger (RT), are used in combination with each of the selection operators and neighborhood topologies. For each trigger configuration the heuristic change frequency parameter, k, was set to $k = 20\%$.

- **Entity assignment counts**: The number of entities assigned to each heuristic by any hyper-heuristic is recorded at every iteration of every algorithm run. The entity

count of each heuristic is used to calculate heuristic space diversity (as outlined in section 3.4.3) for each hyper-heuristic. The entity reassignment rates of each hyper-heuristic are also calculated and analyzed, as explained below in section 6.3.5.

6.3 Experimental Results

All hyper-heuristics and control group algorithms were configured and executed as outlined in section 6.2 above. The results are organized into distinct research questions which are listed in section 6.3.1 and answered in the subsequent sections below.

6.3.1 Research Questions

The following research questions are investigated:

1. *How does each hyper-heuristic perform relative to the pool of stand-alone heuristics?* What is the performance of each hyper-heuristic when configured with different triggers under the *global* and *island* topology, respectively?

2. *How does each hyper-heuristic perform relative to the pool of speciated heuristics?* The same investigation as in research question 1 above is repeated, but focusing on the speciated heuristics.

3. *How do the hyper-heuristics perform relative to each other, fixed heuristic selection, and random heuristic selection?* Which hyper-heuristics perform statistically significantly better than others across a variety of different types of DOPs? Which hyper-heuristics manage to outperform the random and fixed heuristic selection control groups? How do the results differ across different combinations of triggers and topologies?

4. *What are the entity reassignment characteristics of each hyper-heuristic?* How do different hyper-heuristics manage entity-to-heuristic assignments across all iterations of an algorithm run? How does the heuristic space diversity change over

time, as measured by $\mathcal{N}(t)$ using equation (3.2)? How is the heuristic space diversity behavior of each hyper-heuristic affected across different environment types? How strongly does the choice of topology and trigger affect the behavior of each hyper-heuristic?

Each of the above research questions are addressed as separate sections below.

6.3.2 Research Question 1

How does each hyper-heuristic perform relative to the pool of stand-alone heuristics?

A key question in this thesis is whether the investigated hyper-heuristics perform significantly better than the individual heuristics that are managed by each hyper-heuristic. An effective hyper-heuristic should be able to raise performance above what any of the individual heuristics would have achieved on their own.

The performance of each hyper-heuristic was assessed collectively relative to the performance of each one of the individual heuristics. For each individual environment, a Friedman test was used to compare the result set of an individual hyper-heuristic to the nine result sets of the nine standalone heuristics running in isolation. Subsequently, a Shaffer post hoc test was used to perform a pairwise comparison between the hyper-heuristic and all individual heuristics, as explained in section 2.7.4. Wins, draws, and losses were assigned using a significance level of $\alpha = 0.05$. This process was repeated for each individual hyper-heuristic using each of the six different combinations of trigger and topology. The outcome of this process is a set of tables that show the number of wins, draws, and losses of each hyper-heuristic versus the nine heuristics for each environment, for each of the six combinations of trigger and topology. Tables A.1, A.2, A.3, A.4, A.5, and A.6 in appendix A show the detailed results for each hyper-heuristic for each environment, for each configuration.

The results of the analysis above are shown in aggregate form, since the sheer volume of algorithms, hyper-heuristic configurations, and environment types makes it difficult to view in tabular form. Figure 6.1 shows multiple heat maps that aggregate (across all environments) the specific win counts achieved by each hyper-heuristic configuration.

Columns in each heat map represent different hyper-heuristic configurations that used either the *global* or *island* topology combined with the RT, PT, and ST triggers. Row tick values represent the number of individual heuristics a given hyper-heuristic configuration managed to record significant wins against. Darker shades of green represent a larger number of environments where the hyper-heuristic achieved each specific number of wins. For example, **ETour25** achieved wins against eight individual heuristics across 18 different environments when using the *Global_RT* configuration. The sum of the intensities of each column is always equal to 27, i.e. each hyper-heuristic configuration always had to record between zero to nine wins for each of the 27 environments.

Figure 6.2 shows the same type of chart as for wins, but illustrates the draws (in blue) for each hyper-heuristic. Note that there is no figure for losses, since no hyper-heuristic ever recorded a loss against any individual heuristic.

The following observations below, with regard to the stated research question, are noteworthy in figures 6.1 and 6.2.

A striking observation is that none of the 21 hyper-heuristics ever recorded a loss against any individual heuristic for any environment, as tables A.1, A.2, A.3, A.4, A.5, and A.6 in appendix A show. This definitively shows that none of the investigated hyper-heuristics ever yielded worse performance than any of the individual heuristics they managed. Each hyper-heuristic was able to either match the performance of the heuristics (i.e. draws) or raise performance in a statistically significant manner (i.e. wins) compared to simply using any of the low-level heuristics in isolation.

Since no losses were ever recorded (and wins, draws and losses must sum up to nine), figure 6.2 is a mirror image of figure 6.1 for each hyper-heuristic. For every hyper-heuristic respectively, the intensity of the shades of blue for each draw count in figure 6.2 corresponded exactly to the intensity of the shades of green for nine minus that draw count in figure 6.1. As an example, **ETour25**, when using the *Global_RT* configuration, achieved wins against eight heuristics for 18 different environments (dark green cell in figure 6.1) and, subsequently, recorded draws against one heuristic for 18 environments (dark blue cell in figure 6.2). This symmetrical relationship between wins and draws is only prevalent here, since there were zero losses overall (and wins, draws and losses must sum up to nine). The presence of non-zero losses would not necessarily yield such a symmetrical relationship between wins and draws.

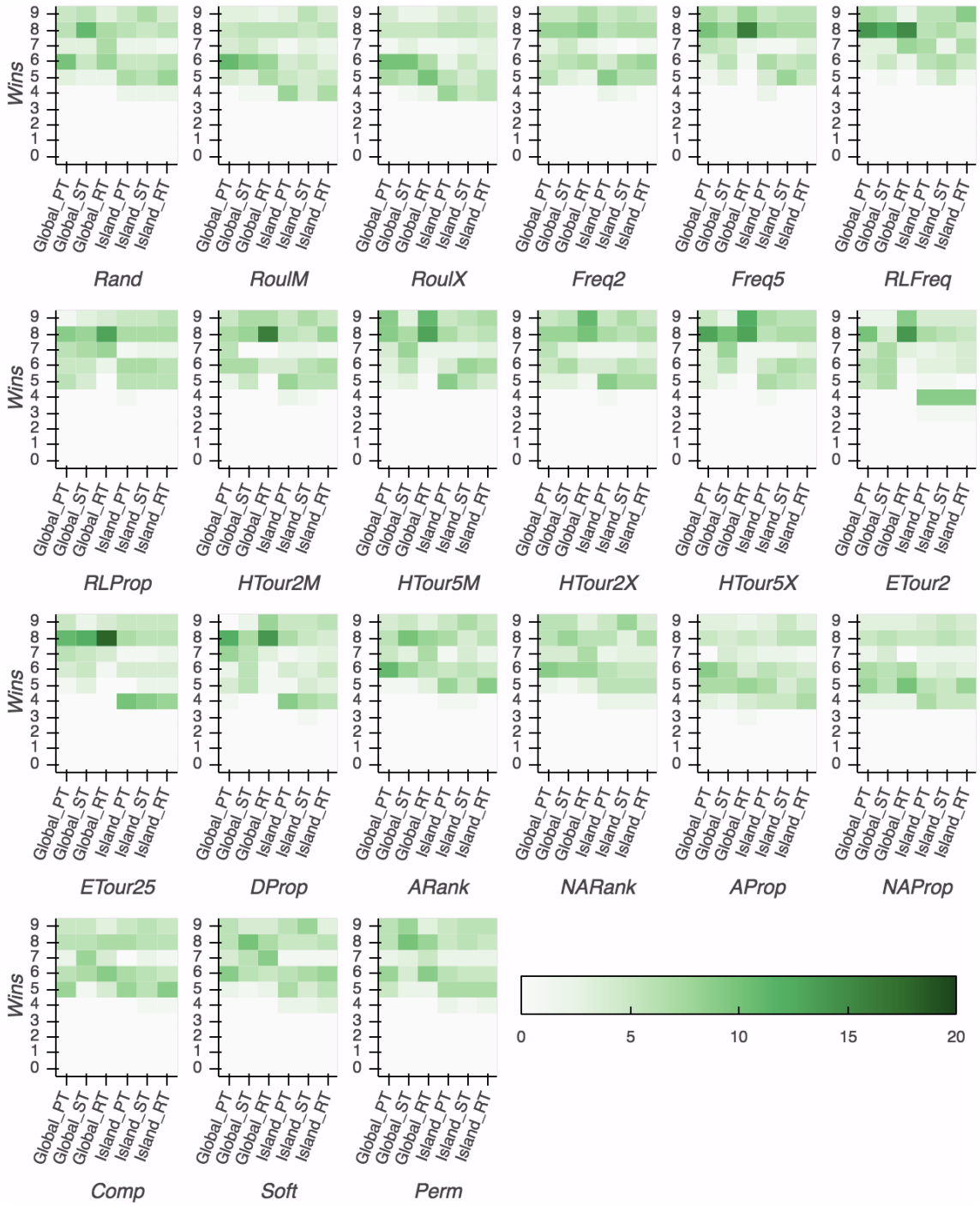


Figure 6.1: Heat map of the number of environments for which each hyper-heuristic managed to achieve specific win counts against the stand-alone heuristics, organized by trigger and topology type. Deeper shades of green indicate more environments for which the hyper-heuristic achieved a specific number of wins.

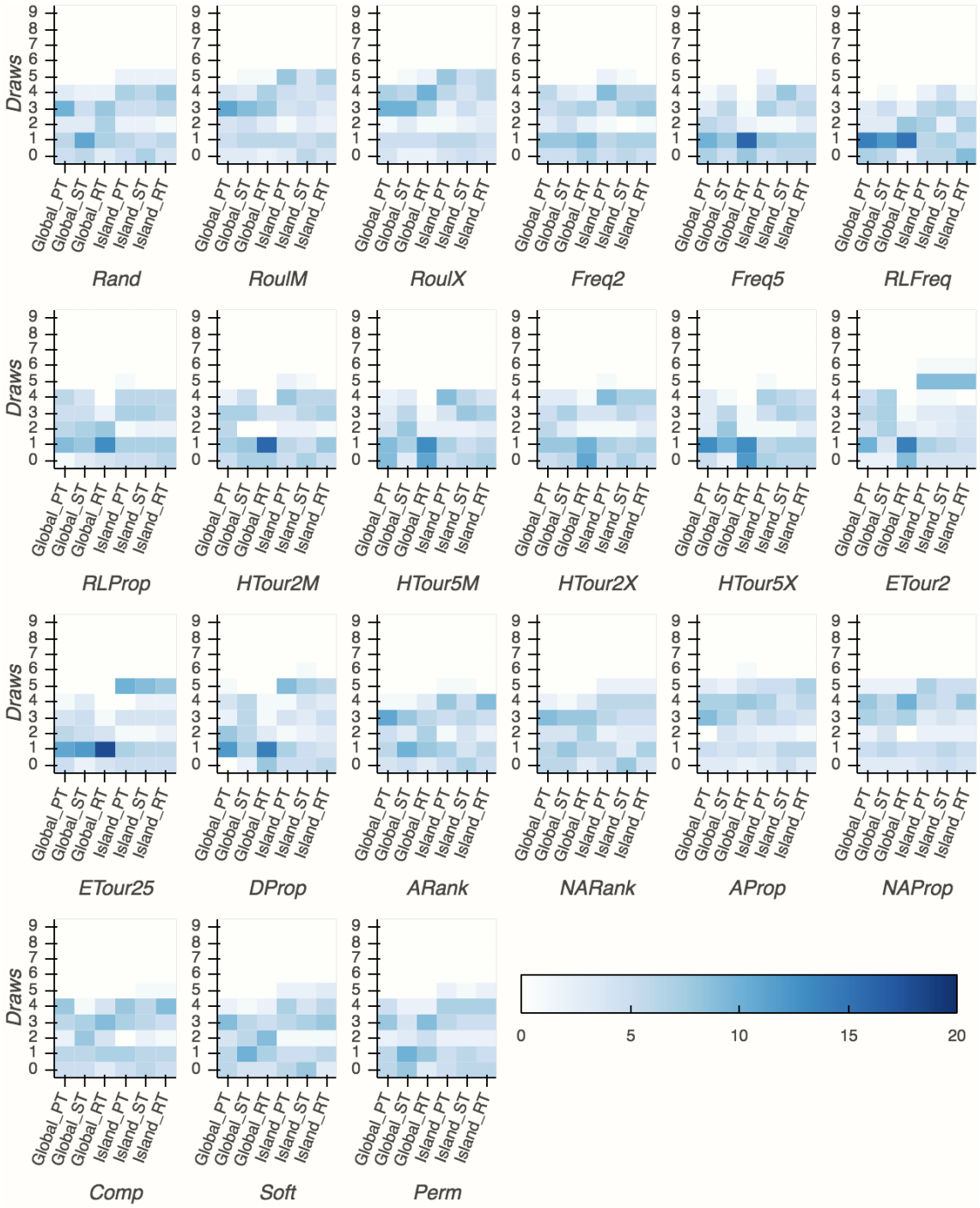


Figure 6.2: Heat map of the number of environments for which each hyper-heuristic managed to achieve specific draw counts against the stand-alone heuristics, organized by trigger and topology type. Deeper shades of blue indicate more environments for which the hyper-heuristic achieved a specific number of draws.

Every hyper-heuristic recorded at least four wins against the individual heuristics in every environment, regardless of the topology or trigger configuration. This can be seen graphically in figure 6.1 where no rows with tick values of three or less showed any non-zero values. Every hyper-heuristic had at least one environment where that hyper-heuristic recorded nine wins against all nine underlying heuristics (and by implication zero draws).

Overall, roughly half of the hyper-heuristics have the majority of their darker shades near the top of their respective heat maps, indicating that those hyper-heuristics had eight or nine wins against heuristics across the majority of environments. The other half of the hyper-heuristics show a more balanced gradient of green across the upper half of the heat map, where those hyper-heuristics recorded between five and nine wins against heuristics. The balanced shades of green indicate that these hyper-heuristics were more sensitive to the environment under consideration.

For many hyper-heuristics, the three *global* topology configurations show darker shades of green near the top of their heat maps in figure 6.1 than the corresponding *island* configurations do. This indicates that many hyper-heuristics configured with the *global* topology showed a greater propensity to outperform a larger number of heuristics than the *island* topology did. Figure 6.2 corroborates this finding by showing how the majority of environments yielded fewer draws between the hyper-heuristic and the heuristics (i.e. blue shades are deeper at lower row values) when using the *global* topology than the *island* topology.

Many hyper-heuristics using the *Global_RT* configuration achieved eight or nine wins for the majority of environments (with, respectively, either one or zero corresponding draws and zero losses). The implication is that these hyper-heuristics had a high propensity to outperform the majority of their constituent heuristics in most environment types that may be encountered. Examples include **Freq5**, **RLFreq**, **RLProp**, **HTour2M**, **HTour5M**, **HTour2X**, **HTour5X**, **ETour2**, **ETour25**, and **DProp**.

In contrast, a number of hyper-heuristics showed a more balanced blend of roughly the same shade of green across win counts, regardless of the topology or trigger type. Good examples are **RoulM**, **RoulX**, **Freq2**, **ARank**, **NARank**, **AProp**, **NAProp**, **Comp**, **Soft**, and **Perm**. The implication is that the ability of these hyper-heuristics to outperform their constituent heuristics was highly problem dependent. In other words,

the number of wins achieved by these hyper-heuristics against their constituent heuristics were sensitive to the specific environment at hand, which resulted in a wider spread of wins and, by implication, a wider spread of draws.

No hyper-heuristic, using any combination of trigger or topology, ever recorded any losses against any single individual heuristic. Tables 6.1 and 6.2 show the wins, draws, and losses of the RT, PT, and ST configurations for each hyper-heuristic against the individual stand-alone heuristics in direct 1×1 comparisons, respectively, for the *global* and *island* topology. For example, against **APSO**, the tuple "15-12-0" for **ETour25** (using the *Global_RT* configuration) recorded wins in 15 environments, draws in 12 environments, and had losses in zero environments.

Overall, a significant portion of hyper-heuristics showed a high proclivity to outperform the majority of their constituent heuristics across most of the environments. The remaining hyper-heuristics showed a more balanced blend of performance results, where the ability to outperform more than half of their constituent heuristics was problem dependent. Regardless, none of the hyper-heuristics ever recorded a loss against any of the standalone heuristics for any environment. It is clear that each of the investigated hyper-heuristics either performed on par with, or statistically significantly outperformed each of the individual heuristics running in isolation.

The choice of topology and trigger made a noticeable impact on the ability of each hyper-heuristic to outperform the individual heuristics. Hyper-heuristics using the *global* topology generally had a higher number of wins for more environments than the same hyper-heuristics using the *island* topology. The *Global_RT* configuration showed the strongest ability to yield the highest number of wins across the most environments for nearly half of the hyper-heuristics.

6.3.3 Research Question 2

> *How does each hyper-heuristic perform relative to the pool of speciated heuristics?*

The same analysis used in research question 1 above is applied to the nine homogeneous speciated heuristics, as described in section 4.3.5. Figures 6.3 and 6.4 show the same type of information as in research question 1. Note that losses were recorded by a

Table 6.1: Hyper-heuristics versus stand-alone heuristics (*global* topology). The notation *W-D-L* indicates the wins, draws, and losses of each hyper-heuristic.

HH	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm	Conf
APSO	10-17-0	7-20-0	7-20-0	12-15-0	16-11-0	16-11-0	20-7-0	16-11-0	19-8-0	9-18-0	9-18-0	8-19-0	7-20-0	18-9-0	15-12-0	13-14-0	12-15-0	16-11-0	9-18-0	10-17-0	9-18-0	**Global_RT**
CPSO	27-0-0	23-4-0	23-4-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	20-7-0	22-5-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	
DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
QPSO1	20-7-0	15-12-0	11-16-0	20-7-0	26-1-0	23-4-0	26-1-0	23-4-0	26-1-0	20-7-0	19-8-0	12-15-0	10-17-0	26-1-0	26-1-0	24-3-0	25-2-0	25-2-0	18-9-0	21-6-0	18-9-0	
QPSO25	6-21-0	4-23-0	3-24-0	13-14-0	19-8-0	18-9-0	19-8-0	18-9-0	21-6-0	7-20-0	5-22-0	3-24-0	5-22-0	18-9-0	16-11-0	13-14-0	14-13-0	17-10-0	6-21-0	7-20-0	8-19-0	
RIGA_B	19-8-0	17-10-0	13-14-0	21-6-0	24-3-0	23-4-0	23-4-0	23-4-0	24-3-0	20-7-0	19-8-0	12-15-0	15-12-0	23-4-0	22-5-0	21-6-0	23-4-0	23-4-0	17-10-0	22-5-0	20-7-0	
RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
APSO	11-16-0	8-19-0	10-17-0	11-16-0	13-14-0	11-16-0	16-11-0	11-16-0	14-13-0	10-17-0	12-15-0	8-19-0	9-18-0	10-17-0	14-13-0	16-11-0	12-15-0	12-15-0	11-16-0	12-15-0	12-15-0	**Global_PT**
CPSO	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	23-4-0	24-3-0	26-1-0	27-0-0	27-0-0	26-1-0	26-1-0	26-1-0	27-0-0	27-0-0	
DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
QPSO1	16-11-0	16-11-0	13-14-0	17-10-0	22-5-0	19-8-0	24-3-0	20-7-0	23-4-0	19-8-0	17-10-0	11-16-0	12-15-0	19-8-0	24-3-0	23-4-0	15-12-0	20-7-0	13-14-0	17-10-0	17-10-0	
QPSO25	7-20-0	5-22-0	5-22-0	10-17-0	15-12-0	10-17-0	14-13-0	9-18-0	13-14-0	6-21-0	9-18-0	5-22-0	5-22-0	8-19-0	11-16-0	17-10-0	6-21-0	6-21-0	7-20-0	8-19-0	8-19-0	
RIGA_B	20-7-0	17-10-0	16-11-0	17-10-0	23-4-0	20-7-0	20-7-0	21-6-0	18-9-0	20-7-0	21-6-0	13-14-0	13-14-0	16-11-0	20-7-0	23-4-0	16-11-0	16-11-0	18-9-0	22-5-0	17-10-0	
RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
APSO	12-15-0	10-17-0	8-19-0	12-15-0	11-16-0	12-15-0	11-16-0	11-16-0	9-18-0	11-16-0	11-16-0	9-18-0	10-17-0	8-19-0	13-14-0	12-15-0	11-16-0	10-17-0	11-16-0	10-17-0	15-12-0	**Global_ST**
CPSO	26-1-0	25-2-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	25-2-0	24-3-0	25-2-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	
DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
QPSO1	22-5-0	18-9-0	11-16-0	19-8-0	18-9-0	16-11-0	20-7-0	17-10-0	21-6-0	21-6-0	21-6-0	10-17-0	15-12-0	14-13-0	21-6-0	22-5-0	19-8-0	17-10-0	22-5-0	22-5-0	21-6-0	
QPSO25	13-14-0	6-21-0	5-22-0	10-17-0	10-17-0	12-15-0	7-20-0	11-16-0	10-17-0	11-16-0	12-15-0	6-21-0	6-21-0	4-23-0	8-19-0	16-11-0	6-21-0	6-21-0	8-19-0	10-17-0	14-13-0	
RIGA_B	22-5-0	16-11-0	15-12-0	19-8-0	21-6-0	19-8-0	17-10-0	22-5-0	20-7-0	21-6-0	20-7-0	14-13-0	14-13-0	17-10-0	17-10-0	21-6-0	19-8-0	15-12-0	20-7-0	22-5-0	22-5-0	
RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	

Table 6.2: Hyper-heuristics versus stand-alone heuristics (*island* topology). The notation *W-D-L* indicates the wins, draws, and losses of each hyper-heuristic.

Conf	HH	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
Island_RT	**APSO**	10-17-0	7-20-0	7-20-0	12-15-0	16-11-0	16-11-0	20-7-0	16-11-0	19-8-0	9-18-0	9-18-0	8-19-0	7-20-0	18-9-0	15-12-0	13-14-0	12-15-0	16-11-0	9-18-0	10-17-0	9-18-0
	CPSO	27-0-0	23-4-0	23-4-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	20-7-0	22-5-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0
	DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	QPSO1	20-7-0	15-12-0	11-16-0	20-7-0	26-1-0	23-4-0	26-1-0	23-4-0	26-1-0	20-7-0	19-8-0	12-15-0	10-17-0	26-1-0	26-1-0	24-3-0	25-2-0	25-2-0	18-9-0	21-6-0	18-9-0
	QPSO25	6-21-0	4-23-0	3-24-0	13-14-0	19-8-0	18-9-0	19-8-0	18-9-0	21-6-0	7-20-0	5-22-0	3-24-0	5-22-0	18-9-0	16-11-0	13-14-0	14-13-0	17-10-0	6-21-0	7-20-0	8-19-0
	RIGA_B	19-8-0	17-10-0	13-14-0	21-6-0	24-3-0	23-4-0	23-4-0	23-4-0	24-3-0	20-7-0	19-8-0	12-15-0	15-12-0	23-4-0	22-5-0	21-6-0	23-4-0	23-4-0	17-10-0	22-5-0	20-7-0
	RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
Island_PT	**APSO**	11-16-0	8-19-0	10-17-0	11-16-0	13-14-0	11-16-0	16-11-0	11-16-0	14-13-0	10-17-0	12-15-0	8-19-0	9-18-0	10-17-0	14-13-0	16-11-0	12-15-0	12-15-0	11-16-0	12-15-0	12-15-0
	CPSO	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	23-4-0	24-3-0	26-1-0	27-0-0	27-0-0	26-1-0	26-1-0	26-1-0	27-0-0	27-0-0
	DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	QPSO1	16-11-0	16-11-0	13-14-0	17-10-0	22-5-0	19-8-0	24-3-0	20-7-0	23-4-0	19-8-0	17-10-0	11-16-0	12-15-0	19-8-0	24-3-0	23-4-0	15-12-0	20-7-0	13-14-0	17-10-0	17-10-0
	QPSO25	7-20-0	5-22-0	5-22-0	10-17-0	15-12-0	10-17-0	14-13-0	9-18-0	13-14-0	6-21-0	9-18-0	5-22-0	5-22-0	8-19-0	11-16-0	17-10-0	6-21-0	6-21-0	7-20-0	8-19-0	8-19-0
	RIGA_B	20-7-0	17-10-0	16-11-0	17-10-0	23-4-0	20-7-0	20-7-0	21-6-0	18-9-0	20-7-0	21-6-0	13-14-0	13-14-0	16-11-0	20-7-0	23-4-0	16-11-0	16-11-0	18-9-0	22-5-0	17-10-0
	RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
Island_ST	**APSO**	12-15-0	10-17-0	8-19-0	12-15-0	11-16-0	12-15-0	11-16-0	11-16-0	9-18-0	11-16-0	11-16-0	9-18-0	10-17-0	8-19-0	13-14-0	12-15-0	11-16-0	10-17-0	11-16-0	10-17-0	15-12-0
	CPSO	26-1-0	25-2-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	25-2-0	24-3-0	25-2-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0
	DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	QPSO1	22-5-0	18-9-0	11-16-0	19-8-0	18-9-0	16-11-0	20-7-0	17-10-0	21-6-0	21-6-0	21-6-0	10-17-0	15-12-0	14-13-0	21-6-0	22-5-0	19-8-0	17-10-0	22-5-0	22-5-0	21-6-0
	QPSO25	13-14-0	6-21-0	5-22-0	10-17-0	10-17-0	12-15-0	7-20-0	11-16-0	10-17-0	11-16-0	12-15-0	6-21-0	6-21-0	4-23-0	8-19-0	16-11-0	6-21-0	6-21-0	8-19-0	10-17-0	14-13-0
	RIGA_B	22-5-0	16-11-0	15-12-0	19-8-0	21-6-0	19-8-0	17-10-0	22-5-0	20-7-0	21-6-0	20-7-0	14-13-0	14-13-0	17-10-0	17-10-0	21-6-0	19-8-0	15-12-0	20-7-0	22-5-0	22-5-0
	RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0

number of hyper-heuristics during this analysis, and figure 6.5 presents this information in the same format as the wins and draws are presented in the figures 6.3 and 6.4. Tables A.7, A.8, A.9, A.10, A.11, and A.12 in appendix A show the detailed results for each environment.

The following observations below, with regard to the stated research question, can be gleaned from the results in figures 6.3, 6.4, and 6.5.

In contrast to the results for the stand-alone heuristics, roughly half of the hyper-heuristics recorded a handful of losses against the speciated heuristics. Losses were isolated, where the hyper-heuristics in question were outperformed by just one or two speciated heuristics. Such losses were generally localized to one or two of the 27 environments. Exceptions were **RoulM**, **ETour25**, and **AProp** which recorded a single loss for three or four environments when the *Global_RT* configuration was used. Regardless, the overwhelming number of cases yielded zero losses.

The majority of hyper-heuristics scored four or more wins against the speciated heuristics in all environments. Isolated exceptions exist where hyper-heuristics showed only two or three wins in one environment. No hyper-heuristics ever managed to achieve more than seven wins against the nine speciated heuristics in any environment.

For the previous research question, the shades of green were relatively spread out for each of the individual heuristic cases in figure 6.1. In contrast, a greater number of hyper-heuristics in figure 6.3 show individual cells that have notably darker shades of green than other cells. This indicates that more hyper-heuristic configurations yielded predictably similar results against the speciated heuristics for a large number of environments. In other words, most hyper-heuristic variations tended to yield the same number of wins against the speciated heuristics for the majority of environments. Different hyper-heuristics show this stable number of wins across environments at different win counts. For many hyper-heuristics, the precise win count where this same number of wins occur differs depending on the trigger and topology configuration that was used. For example, **ETour2** yielded four wins for 15 environments when using either the *Island_PT* or *Island_ST* configuration, while **Freq2** yielded seven wins in most environments only when using the *Global_RT* configuration. Fourteen of the 21 hyper-heuristics using the *Global_RT* configuration recorded wins against seven heuristics for a sizable number of environments. A similar pattern is noticeable for the *Island_RT* configuration, but in

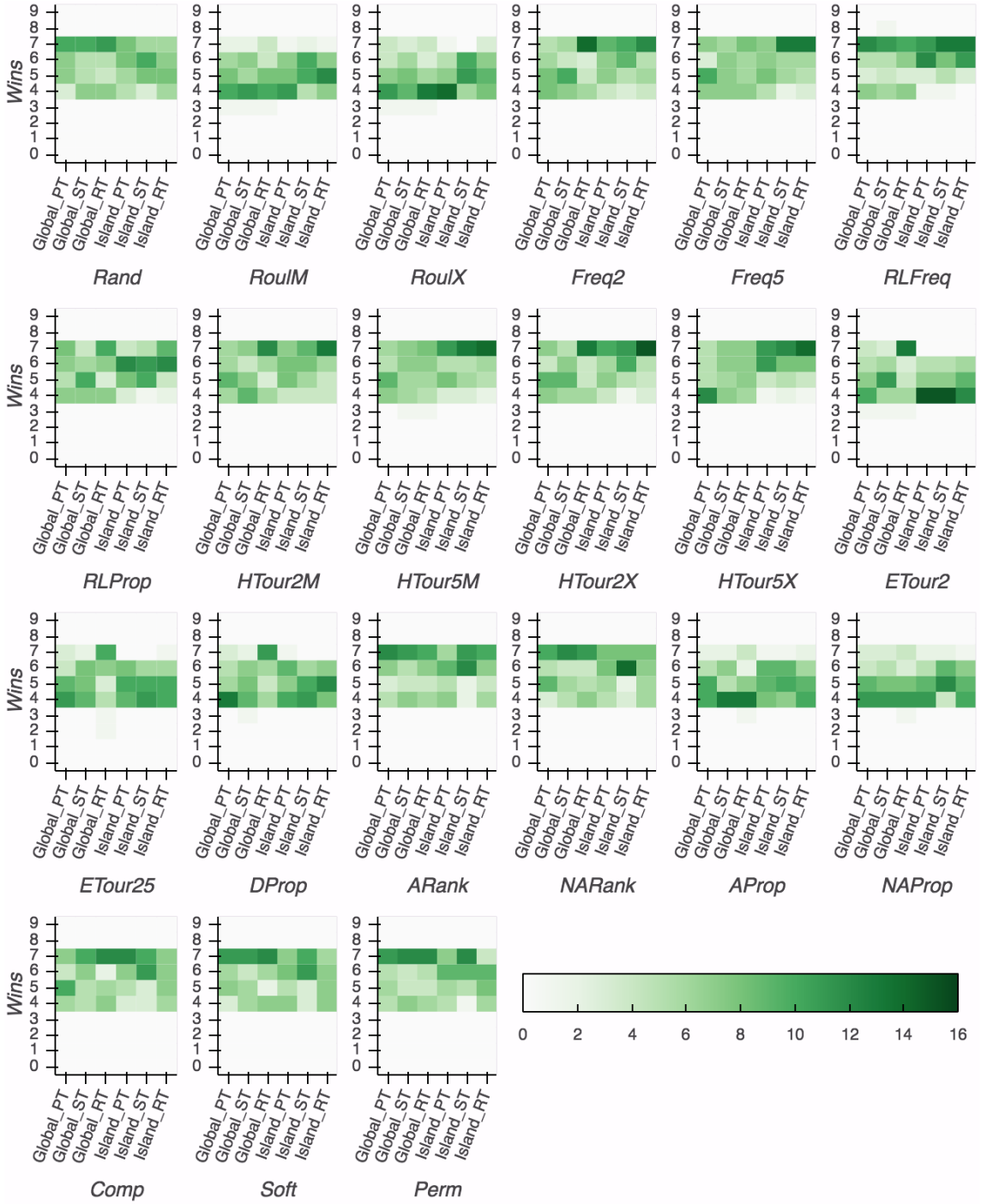


Figure 6.3: Heat map of the number of environments for which each hyper-heuristic managed to achieve specific win counts against the speciated heuristics, organized by trigger and topology type. Deeper shades of green indicate more environments for which the hyper-heuristic achieved a specific number of wins.

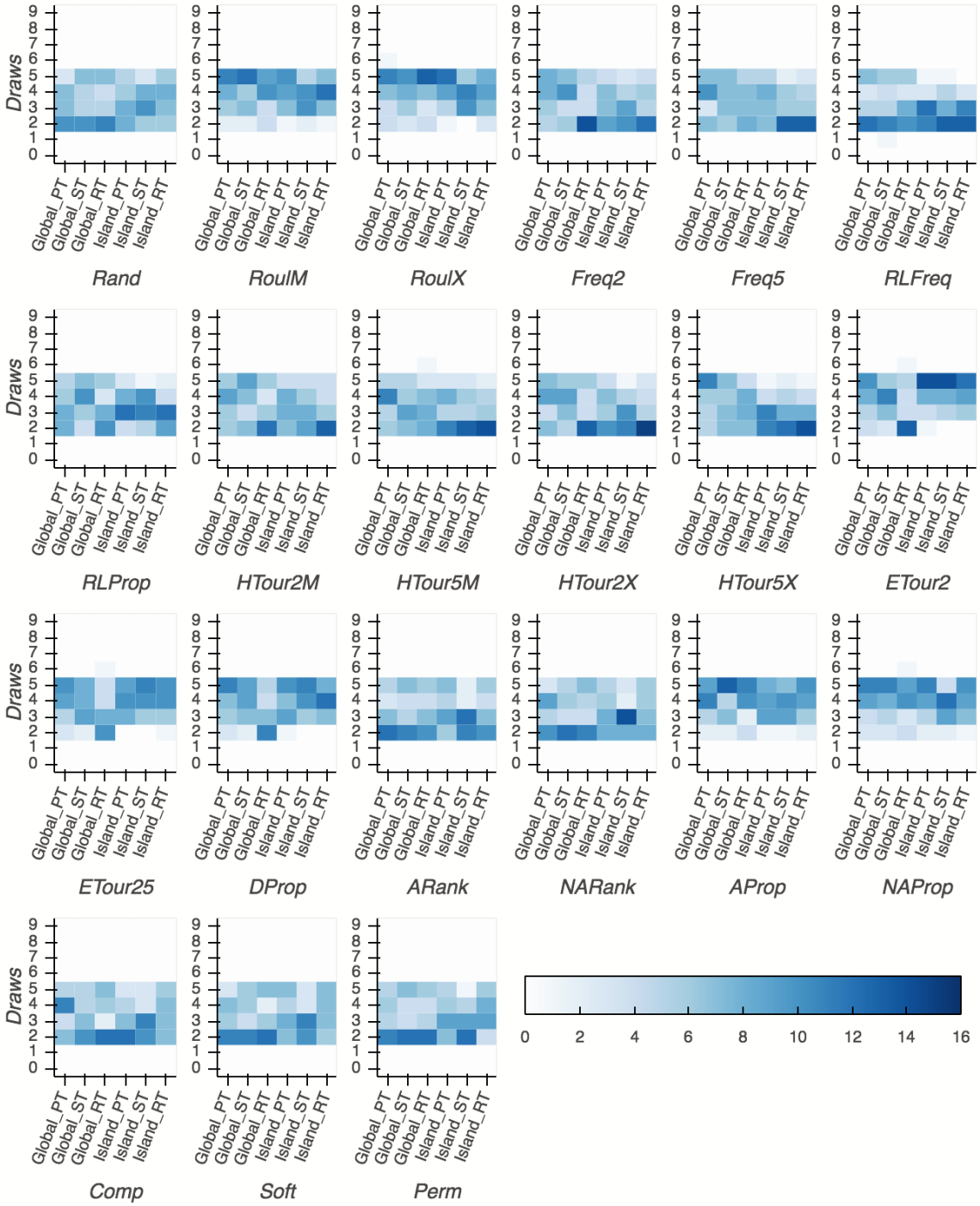


Figure 6.4: Heat map of the number of environments for which each hyper-heuristic managed to achieve specific draw counts against the speciated heuristics, organized by trigger and topology type. Deeper shades of blue indicate more environments for which the hyper-heuristic achieved a specific number of draws.

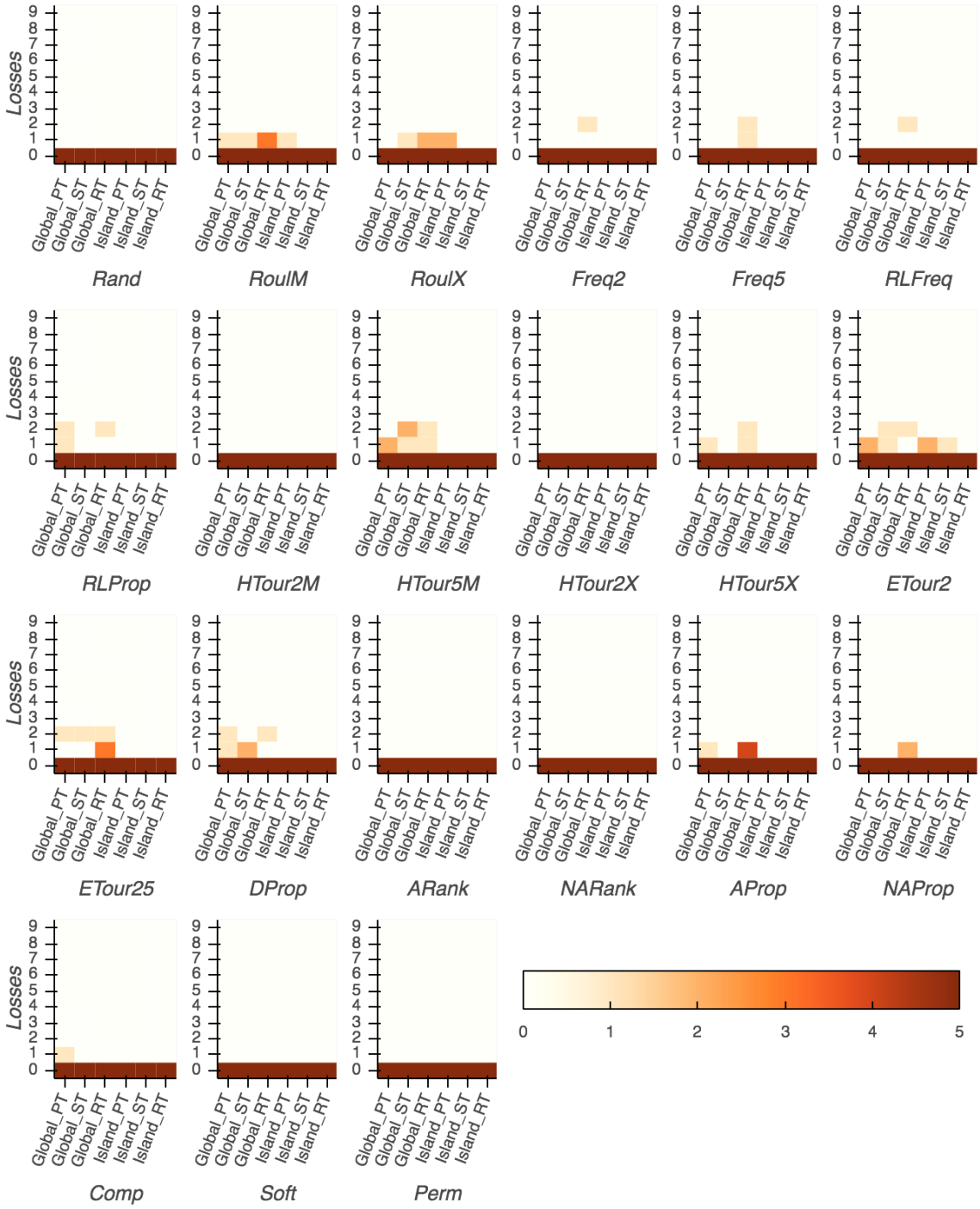


Figure 6.5: Heat map of the number of environments for which each hyper-heuristic achieved specific loss counts against the speciated heuristics, organized by trigger and topology type. Deeper shades of orange indicate more environments for which the hyper-heuristic suffered a specific number of losses.

only eight of the hyper-heuristics.

Three weak patterns were observed that differentiate the performance of the *global* topology from that of the *island* topology:

- **Rand**, **ETour2**, **ETour25**, **DProp**, **NARank**, **Soft**, and **Perm** exhibited a higher concentration of environments where these hyper-heuristics achieved more wins using the *global* topology than the *island* topology configurations.
- However, **Freq5**, **RLFreq**, **HTour5M**, **HTour2X**, and **HTour5X** show the opposite pattern where the *island* topology shows more wins in more environments.
- The remaining hyper-heuristics show a relative balance between the *global* and *island* topologies.

Figure 6.3 shows that **RLFreq** is the only hyper-heuristic that has a dark shaded band of green across the seven wins row in the heat map. Figure 6.4 shows a matching bottom row of dark blues at the two draws row tick, while figure 6.5 shows a single environment where two speciated heuristics managed to outperform **RLFreq** (and that in only the *Global_RT* configuration). This indicates that **RLFreq** managed to match or outperform seven of the nine speciated heuristics in the majority of environments, regardless of the trigger or topology configuration. In that sense, **RLfreq** is one of the most stable hyper-heuristic algorithms with respect to trigger and topology choices.

The hyper-heuristics recorded losses against the speciated heuristics in a handful of environments. Tables 6.3 and 6.4 show the wins, draws, and losses of each hyper-heuristic against the speciated heuristics in direct one-to-one comparisons, respectively, for the *global* and *island* topology. For example, against **APSO**, the tuple "6-20-1" for **ETour25** (using the *Island_RT* configuration) recorded wins in six environments, draws in 20 environments, and had losses in one environment.

The investigated hyper-heuristics were able to either outperform or match the performance of the homogeneous speciated heuristics in a statistically significant manner. Isolated losses were recorded by a small number of hyper-heuristics across a handful of environments. Superior performance of the hyper-heuristics over that of the speciated heuristics was not as clear-cut as for the stand-alone heuristics, and no hyper-heuristic ever managed to outright beat all nine speciated heuristics. Isolated cases exist where

Table 6.3: Hyper-heuristics versus speciated heuristics (*global* topology). The notation *W-D-L* indicates the wins, draws, and losses of each hyper-heuristic.

HH	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm	Conf
S_APSO	5-22-0	5-21-1	4-22-1	6-20-1	6-19-2	7-20-0	6-19-2	6-21-0	6-20-1	6-21-0	5-22-0	5-19-3	4-21-2	5-21-1	6-20-1	5-21-1	5-21-1	5-21-1	5-21-1	7-20-0	7-20-0	Global_RT
S_CPSO	13-14-0	7-20-0	3-24-0	14-13-0	12-15-0	14-13-0	15-12-0	14-13-0	13-14-0	13-14-0	11-16-0	2-25-0	2-25-0	15-12-0	14-13-0	13-14-0	13-14-0	14-13-0	10-17-0	13-14-0	12-15-0	
S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	26-1-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_QPSO1	11-16-0	8-19-0	4-23-0	12-15-0	11-16-0	11-16-0	12-15-0	11-16-0	11-16-0	11-16-0	11-16-0	3-24-0	5-22-0	12-15-0	10-17-0	10-17-0	10-17-0	10-17-0	11-16-0	10-17-0	12-15-0	
S_QPSO25	0-27-0	0-26-1	0-25-2	0-26-1	0-26-1	0-26-1	0-25-2	0-26-1	1-24-2	0-27-0	0-26-1	0-27-0	0-27-0	1-24-2	1-22-4	0-26-1	0-26-1	0-25-2	0-27-0	0-27-0	0-26-1	
S_RIGA_B	20-7-0	18-9-0	13-14-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	20-7-0	14-13-0	16-11-0	20-7-0	20-7-0	20-7-0	20-7-0	21-6-0	19-8-0	20-7-0	20-7-0	
S_RIGA_P	27-0-0	26-1-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	25-2-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_APSO	6-21-0	5-22-0	5-22-0	7-20-0	6-21-0	6-21-0	6-20-1	6-21-0	6-20-1	6-21-0	6-21-0	5-21-1	5-22-0	5-21-1	6-20-1	6-20-1	5-21-1	6-20-1	5-21-1	6-21-0	7-20-0	Global_PT
S_CPSO	12-15-0	7-20-0	6-21-0	9-18-0	11-16-0	13-14-0	10-17-0	11-16-0	6-21-0	14-13-0	11-16-0	2-25-0	3-24-0	5-22-0	7-20-0	13-14-0	10-17-0	4-23-0	5-22-0	13-14-0	13-14-0	
S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_QPSO1	10-17-0	3-24-0	2-25-0	7-20-0	11-16-0	12-15-0	8-19-0	10-17-0	6-21-0	11-16-0	10-17-0	4-23-0	2-25-0	5-22-0	6-21-0	11-16-0	9-18-0	2-25-0	7-20-0	11-16-0	11-16-0	
S_QPSO25	0-27-0	0-25-2	0-26-1	0-27-0	0-27-0	0-27-0	0-25-2	0-26-1	0-25-2	0-27-0	0-27-0	0-27-0	0-27-0	0-24-3	1-23-3	1-25-1	0-25-2	0-25-2	0-27-0	0-27-0	0-27-0	
S_RIGA_B	24-3-0	17-10-0	17-10-0	19-8-0	20-7-0	21-6-0	21-6-0	20-7-0	16-11-0	22-5-0	22-5-0	16-11-0	16-11-0	17-10-0	18-9-0	21-6-0	20-7-0	16-11-0	18-9-0	22-5-0	22-5-0	
S_RIGA_P	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_APSO	6-21-0	6-21-0	5-21-1	6-21-0	5-22-0	6-20-1	5-20-2	7-20-0	5-22-0	6-21-0	6-21-0	5-22-0	5-22-0	7-19-1	7-19-1	6-21-0	6-20-1	6-20-1	6-21-0	5-22-0	7-20-0	Global_ST
S_CPSO	14-13-0	6-21-0	7-20-0	11-16-0	11-16-0	10-17-0	10-17-0	9-18-0	12-15-0	15-12-0	14-13-0	6-21-0	5-22-0	7-20-0	9-18-0	14-13-0	12-15-0	5-22-0	13-14-0	13-14-0	14-13-0	
S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	
S_QPSO1	11-16-0	1-26-0	3-24-0	9-18-0	10-17-0	8-19-0	7-20-0	9-18-0	9-18-0	12-15-0	12-15-0	2-25-0	3-24-0	8-19-0	8-19-0	11-16-0	8-19-0	3-24-0	12-15-0	10-17-0	12-15-0	
S_QPSO25	0-27-0	0-25-2	0-25-2	0-26-1	0-26-1	0-26-1	0-25-2	0-26-1	0-26-1	0-26-1	0-27-0	0-26-1	0-27-0	0-25-2	0-26-1	1-25-1	0-26-1	0-24-3	0-27-0	0-27-0	1-26-0	
S_RIGA_B	20-7-0	17-10-0	17-10-0	20-7-0	20-7-0	19-8-0	20-7-0	21-6-0	20-7-0	21-6-0	22-5-0	15-12-0	16-11-0	19-8-0	17-10-0	21-6-0	19-8-0	17-10-0	22-5-0	20-7-0	21-6-0	
S_RIGA_P	27-0-0	26-1-0	26-1-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	

Table 6.4: Hyper-heuristics versus speciated heuristics (*island* topology). The notation *W-D-L* indicates the wins, draws, and losses of each hyper-heuristic.

Conf	HH	Perm	Soft	Comp	DProp	RLProp	RLFreq	ETour25	ETour2	NAProp	AProp	NARank	ARank	HTour5X	HTour2X	HTour5M	HTour2M	Freq5	Freq2	RoulX	RoulM	Rand
Island_RT	S_APSO	7-20-0	7-20-0	5-21-1	5-21-1	5-21-1	5-21-1	6-20-1	5-21-1	4-21-2	5-19-3	5-22-0	6-21-0	6-20-1	6-21-0	6-19-2	7-20-0	6-19-2	6-20-1	4-22-1	5-21-1	5-22-0
	S_CPSO	12-15-0	13-14-0	10-17-0	14-13-0	13-14-0	13-14-0	14-13-0	15-12-0	2-25-0	2-25-0	11-16-0	13-14-0	13-14-0	14-13-0	15-12-0	14-13-0	12-15-0	14-13-0	3-24-0	7-20-0	13-14-0
	S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	26-1-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_QPSO1	12-15-0	10-17-0	11-16-0	10-17-0	10-17-0	10-17-0	10-17-0	12-15-0	5-22-0	3-24-0	11-16-0	11-16-0	11-16-0	11-16-0	12-15-0	11-16-0	11-16-0	12-15-0	4-23-0	8-19-0	11-16-0
	S_QPSO25	0-26-1	0-27-0	0-27-0	0-25-2	0-26-1	0-26-1	1-22-4	1-24-2	0-27-0	0-27-0	0-26-1	0-27-0	1-24-2	0-26-1	0-25-2	0-26-1	0-26-1	0-26-1	0-25-2	0-26-1	0-27-0
	S_RIGA_B	20-7-0	20-7-0	19-8-0	21-6-0	20-7-0	20-7-0	20-7-0	20-7-0	16-11-0	14-13-0	20-7-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	21-6-0	13-14-0	18-9-0	20-7-0
	S_RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	25-2-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	26-1-0	27-0-0
Island_PT	S_APSO	7-20-0	6-21-0	5-21-1	6-20-1	5-21-1	6-20-1	6-20-1	5-21-1	5-22-0	5-21-1	6-21-0	6-21-0	6-20-1	6-21-0	6-20-1	6-21-0	6-21-0	7-20-0	5-22-0	5-22-0	6-21-0
	S_CPSO	13-14-0	13-14-0	5-22-0	4-23-0	10-17-0	13-14-0	7-20-0	5-22-0	3-24-0	2-25-0	11-16-0	14-13-0	6-21-0	11-16-0	10-17-0	13-14-0	11-16-0	9-18-0	6-21-0	7-20-0	12-15-0
	S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_QPSO1	11-16-0	11-16-0	7-20-0	2-25-0	9-18-0	11-16-0	6-21-0	5-22-0	2-25-0	4-23-0	10-17-0	11-16-0	6-21-0	10-17-0	8-19-0	12-15-0	11-16-0	7-20-0	2-25-0	3-24-0	10-17-0
	S_QPSO25	0-27-0	0-27-0	0-27-0	0-25-2	0-25-2	1-25-1	1-23-3	0-24-3	0-27-0	0-27-0	0-27-0	0-27-0	0-25-2	0-26-1	0-25-2	0-27-0	0-27-0	0-27-0	0-26-1	0-25-2	0-27-0
	S_RIGA_B	22-5-0	22-5-0	18-9-0	16-11-0	20-7-0	21-6-0	18-9-0	17-10-0	16-11-0	16-11-0	22-5-0	22-5-0	16-11-0	20-7-0	21-6-0	21-6-0	20-7-0	19-8-0	17-10-0	17-10-0	24-3-0
	S_RIGA_P	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0
Island_ST	S_APSO	7-20-0	5-22-0	6-21-0	6-20-1	6-20-1	6-21-0	7-19-1	7-19-1	5-22-0	5-22-0	6-21-0	6-21-0	5-22-0	7-20-0	5-20-2	6-20-1	5-22-0	6-21-0	5-21-1	6-21-0	6-21-0
	S_CPSO	14-13-0	13-14-0	13-14-0	5-22-0	12-15-0	14-13-0	9-18-0	7-20-0	5-22-0	6-21-0	14-13-0	15-12-0	12-15-0	9-18-0	10-17-0	10-17-0	11-16-0	11-16-0	7-20-0	6-21-0	14-13-0
	S_DEBest1Bin	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss1	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_Gauss10	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0
	S_QPSO1	12-15-0	10-17-0	12-15-0	3-24-0	8-19-0	11-16-0	8-19-0	8-19-0	3-24-0	2-25-0	12-15-0	12-15-0	9-18-0	9-18-0	7-20-0	8-19-0	10-17-0	9-18-0	3-24-0	1-26-0	11-16-0
	S_QPSO25	1-26-0	0-27-0	0-27-0	0-24-3	0-26-1	1-25-1	0-26-1	0-25-2	0-27-0	0-26-1	0-27-0	0-26-1	0-26-1	0-26-1	0-25-2	0-26-1	0-26-1	0-26-1	0-25-2	0-25-2	0-27-0
	S_RIGA_B	21-6-0	20-7-0	22-5-0	17-10-0	19-8-0	21-6-0	17-10-0	19-8-0	16-11-0	15-12-0	22-5-0	21-6-0	20-7-0	21-6-0	20-7-0	19-8-0	20-7-0	20-7-0	17-10-0	17-10-0	20-7-0
	S_RIGA_P	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	27-0-0	26-1-0	27-0-0	27-0-0	27-0-0	26-1-0	26-1-0	27-0-0

a number of hyper-heuristics suffered a small number of losses for two or three of the 27 environments (in sharp contrast to the stand-alone heuristic analysis where no loss was ever recorded by any hyper-heuristic in any environment). Overall, however, the use of any of the investigated hyper-heuristics yielded performance that is either comparable or superior for the majority of environments compared to simply using speciation (regardless of neighborhood topology or heuristic change trigger scheme).

6.3.4 Research Question 3

> *How do the hyper-heuristics perform relative to each other, fixed heuristic selection, and random heuristic selection?*

This research question considers the performance of each hyper-heuristic compared to the performance of all the other hyper-heuristics. The goal is to determine which hyper-heuristics are consistently better than other hyper-heuristics across multiple environments. Each hyper-heuristic is also compared against the two remaining control groups, namely fixed heuristic selection and random heuristic selection, as defined in section 4.3.5.

A Friedman test and the associated Shaffer post hoc test was conducted between all 22 algorithms (i.e. 21 hyper-heuristics and fixed heuristic selection) for each of the 27 environments. Wins, draws, and losses were assigned using a significance level of $\alpha = 0.05$. The analysis was repeated independently for all six configuration combinations comprising of *global* and *island* topologies together with the three types of triggers, namely random triggers (RT), periodic triggers (PT), and stagnation triggers (ST). Note that **GFix** and **IFix** (as defined in section 4.3.5) were, respectively, used in conjunction with the *global* and *island* topology as the fixed heuristic selection method (to ensure a fair apples to apples comparison in each instance).

Figure 6.6 shows, for each of the six possible hyper-heuristic configurations, the resulting wins and losses of each algorithm versus each one of the other algorithms using a 22×22 heat map. The darkness of each cell represents the number of environments for which the winning algorithm (i.e., a row on the y-axis) achieved a win against the losing algorithm (i.e., a column on the x-axis). Algorithms that show a combination of mostly white rows and white columns did not manage to record wins or losses against the other

methods, which indicates draws. On the other hand, an algorithm that has mostly dark shades along a row and white shades along a column managed to record wins against other methods across many environments without incurring losses. Similarly, an algorithm that shows white rows and darker columns indicates that the algorithm recorded few wins and many losses against the other methods. For example, in sub-figure 6.6b, **RLFreq** managed to achieve statistically significant wins against **RoulX** for 20 of the 27 possible environments, while (in the same sub-figure) **RLFreq** never recorded losses against any hyper-heuristic for any environments.

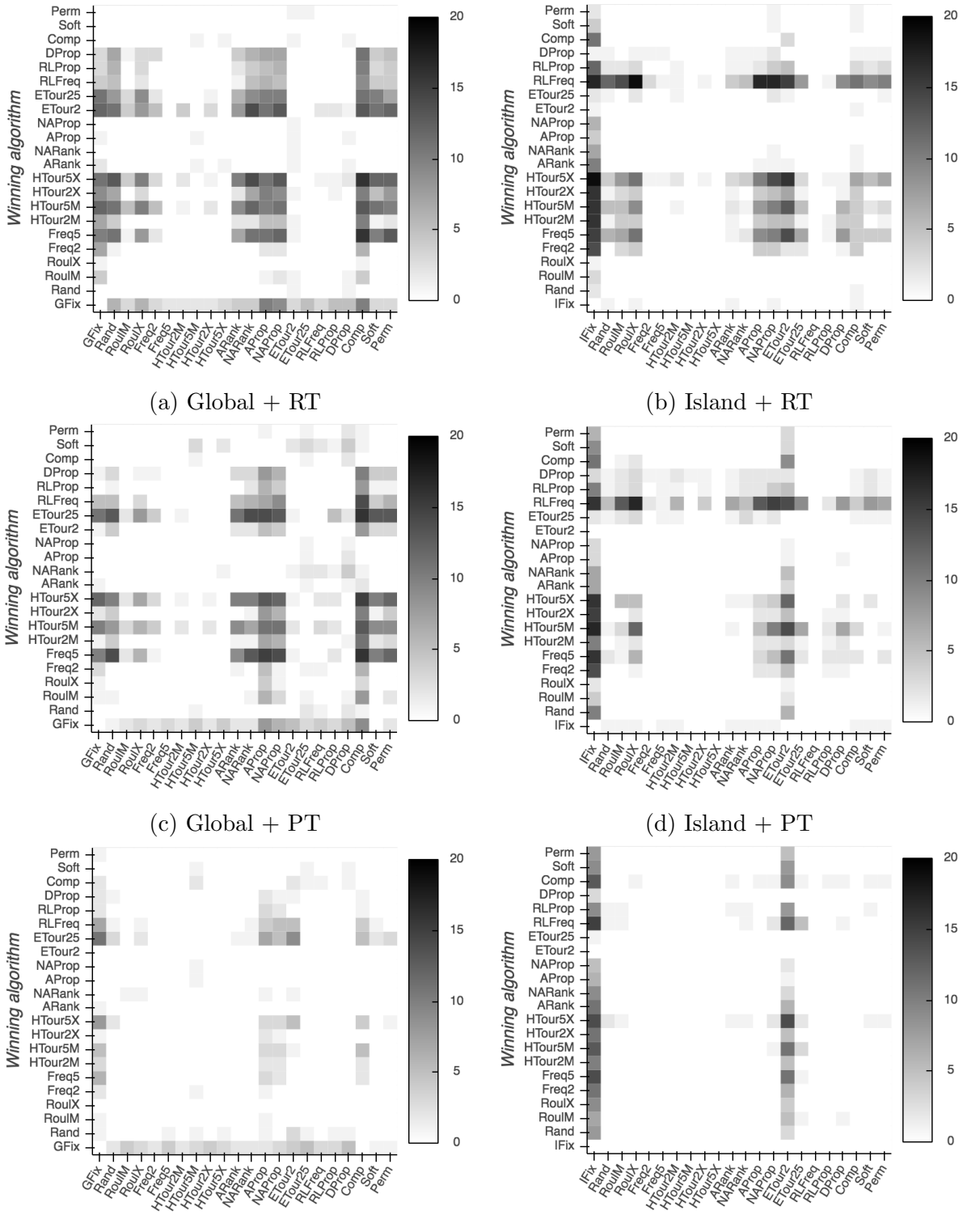
Figure 6.7 shows the mean number of wins and losses for each algorithm for each configuration (averaged over all 27 environment types). Figure 6.6 and 6.7 are deeply related: any hyper-heuristic's mean wins score (i.e. green diamond in figure 6.7) is equal to the row-wise average of the given hyper-heuristic wins in figure 6.6. Similarly, losses in figure 6.7 (i.e. the red stars) correspond to column-wise averages in figure 6.6. For example, the row for **DProp** in sub-figure 6.6d shows how most cells have a value of one and a roughly equal number of cells with values of either zero or two (yielding a row-wise average of roughly 1.0). Sub-figure 6.7d shows a corresponding mean wins value of approximately 1.0 and mean loss value of approximately 0.9 for **DProp**.

Appendix A contains additional tables A.14, A.16, A.18, A.20, A.22, and A.24 which show the detailed wins, draws, and losses of each of the six comparisons in each environment, while tables A.13, A.15, A.17, A.19, A.21, and A.23 show the associated Friedman ranks and p-values.

The following noteworthy observations below, with regard to the stated research question, can be seen in figures 6.6 and 6.7.

Overall visual observations

At a glance, the pair of sub-figures 6.6a and 6.6c as well as the pair 6.6b and 6.6d have relatively similar color distribution patterns across hyper-heuristics. For each pair, over half the number of rows contain a large number of darker shaded cells. The presence of rows with so many dark shaded cells indicates the presence of hyper-heuristics that outperformed numerous other hyper-heuristics in multiple environments. In a similar fashion there are numerous columns that contain multiple darker shaded cells, indicating that



(a) Global + RT

(b) Island + RT

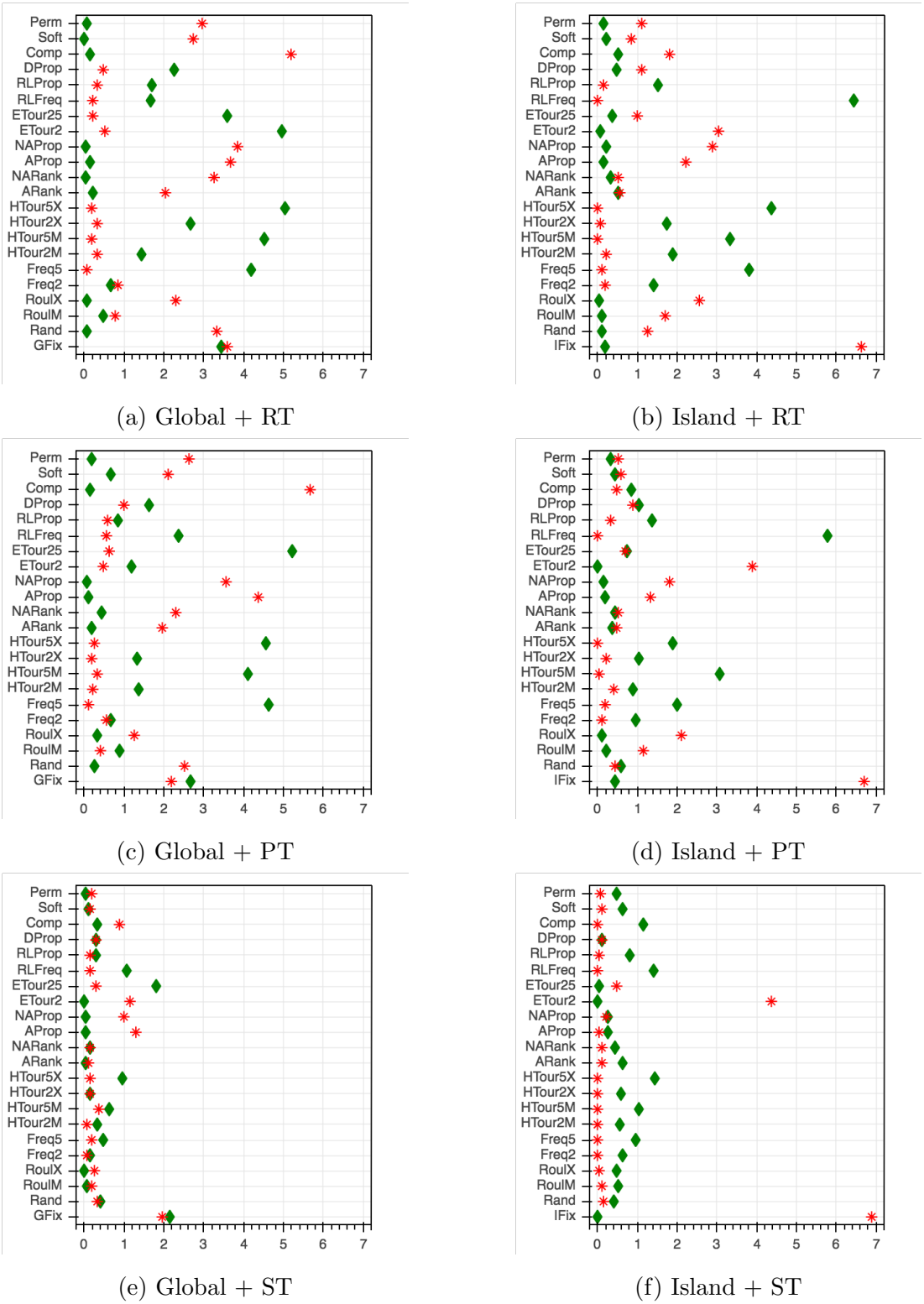
(c) Global + PT

(d) Island + PT

(e) Global + ST

(f) Island + ST

Figure 6.6: Heat map of wins achieved by all hyper-heuristics against each other. Darker cells indicate more environments where the y-axis hyper-heuristic beat the x-axis hyper-heuristic.



(a) Global + RT

(b) Island + RT

(c) Global + PT

(d) Island + PT

(e) Global + ST

(f) Island + ST

Figure 6.7: Number of wins (green diamonds) and losses (red stars) of each hyper-heuristic, averaged across all 27 environments.

those hyper-heuristics suffered significant losses against numerous other hyper-heuristics for many environments.

In contrast, sub-figures 6.6e and 6.6f are much sparser, showing a high number of white or very light gray cells. There are virtually no rows in sub-figures 6.6e or 6.6f that contain darkly shaded cells in any great quantities. The general lack of darkly shaded rows indicates that fewer algorithms managed to significantly outperform each other using the ST trigger (using either the *global* or *island* topology). Similarly, there are almost no columns with any significantly darker shades present across the majority of rows. Two exceptions are **IFix** and **ETour2** in figure 6.6f that completely defy this pattern by exhibiting moderate to very dark cells across nearly the entire column.

The high-level visual findings in figure 6.6 are echoed in figure 6.7. The pairs of sub-figures 6.7a and 6.7c as well as the pair of sub-figures 6.7b and 6.7d show high degrees of visual similarity. The pair of sub-figures 6.7e and 6.7f also show visual similarities, and show mean wins and loss values that are noticeably smaller than the other four sub-figures. The findings indicate the absence of any hyper-heuristics that dominated the other algorithms.

Performance against fixed heuristic selection

Each topology and trigger configuration yielded discernibly varied performance of all hyper-heuristics versus the fixed heuristic allocation control group (i.e. **GFix** and **IFix** for the *global* and *island* topologies respectively).

Arguably the most visually striking observation in figure 6.6 related to fixed heuristic selection is the dark left-most column in sub-figures 6.6b, 6.6d, and 6.6f. The dark column indicates that practically all hyper-heuristics using either the RT, PT, or ST trigger in conjunction with the *island* topology recorded wins against **IFix** for the majority of environments. Correspondingly, the three rows for **IFix** in each of the three listed sub-figures contains mostly white or very light gray cells, showing that **IFix** almost never recorded any wins against any hyper-heuristic. **RLFreq**, **HTour5X**, **HTour2X**, **HTour5M**, **HTour2M**, **Freq5**, and **Freq2** were among the top performers, and yielded wins against **IFix** for approximately 20 of the 27 environments. Most of the other hyper-heuristics also showed high numbers of wins against **IFix** across all three trigger types,

with nearly no losses.

GFix managed to record wins against nearly every hyper-heuristic in sub-figures 6.6a, 6.6c, and 6.6e for a small number of environments. This is visible as a light gray row for the **GFix** entry in each sub-figure. **AProp**, **NAProp**, and **Comp** were among the hyper-heuristics in sub-figures 6.6a and 6.6c that recorded the highest number of losses against **GFix**. These three hyper-heuristics virtually never recorded any wins against **GFix** when using the *global* topology and RT or PT triggers. However, the leftmost columns in the three heat maps show that roughly two thirds of all hyper-heuristics recorded wins against **GFix** for a sizable number of environments, with many hyper-heuristics recording more wins than losses against **GFix** (the remainder of environments yielded draws). A hand-full of hyper-heuristics achieved relatively high number of wins against **GFix** across all three sub-figures 6.6a, 6.6c, and 6.6e, most notably **ETour25**, **HTour5X**, **HTour5M**, and **Freq5**.

It is apparent that hyper-heuristics using the *island* topology have a clear tendency to outperform fixed selection across the majority of environment types, regardless of whether RT, PT, or ST triggers are used. The hyper-heuristics using the *global* topology are more sensitive to the type of environment under consideration, showing both wins and losses against fixed heuristic selection for a number of environments.

Performance against random heuristic selection

Random selection, as another control group, rarely yielded wins in many environments against any of the hyper-heuristics, regardless of their topology or trigger configuration. This is discernible from the fact that **Rand** shows nearly all-white rows in each of the six heat maps in figure 6.6.

Roughly half of all hyper-heuristics configured with the RT trigger recorded wins against **Rand** in many different environments, as indicated by the high number of darker shaded cells in the **Rand** columns of sub-figures 6.6a and 6.6b. Hyper-heuristics configured with the *global* topology showed a slightly stronger tendency (i.e. had a larger number of darker shaded cells) than the *island* topology to perform better across different environments. Virtually no hyper-heuristics using the RT trigger (regardless of topology) ever recorded losses against **Rand** for any environment.

Hyper-heuristics using the PT trigger yielded a varied number of wins against **Rand**, depending on the topology that was employed. Many hyper-heuristics using the *Global_PT* configuration managed to win against **Rand** across numerous environments. **Rand** almost never recorded any notable wins against any of the hyper-heuristics except **ETour25**, **RLFreq**, **DProp**, and **Comp** where **Rand** managed a small number of wins. In contrast, virtually no hyper-heuristics using the *Island_PT* configuration managed to either win or lose against **Rand** for the majority of environments, indicating that there is no real significant difference between random heuristic selection and hyper-heuristics using the PT trigger and an *island* topology. The only possible exception is **RLFreq** that outperformed **Rand** for roughly five of the 27 environments and recorded draws in the remainder of the environments. **Rand** never recorded wins against **RLFreq**.

Rand almost never recorded any noteworthy numbers of wins or losses against any hyper-heuristics that employed the ST trigger, regardless of whether the *global* or *island* topology was used. The interpretation is that every hyper-heuristic that used the ST trigger showed statistically indistinguishable differences from using random heuristic selection.

Observations about hyper-heuristics that employed tournament selection

A distinct pattern that is visible in figure 6.6 was the tendency for most hyper-heuristics that rely on tournament selection (refer to section 3.4.6) to show a strong correlation with darker shaded cells. In particular, larger tournament sizes resulted in noticeably more wins for each particular approach (compare **HTour5X** with **HTour2X**, **HTour5M** with **HTour2M**,and **Freq5** with **Freq2**). The pattern where larger tournament sizes dominated is less noticeable between **ETour25** and **ETour2**, but is present nonetheless (the pattern is, however, absent in sub-figure 6.6a). These observations are corroborated by the fact that figure 6.7 shows that **HTour5X**, **HTour5M**, and **Freq5** were consistently among the top performing hyper-heuristics.

In sub-figures 6.6a, 6.6b, and 6.6c, the methods that were based on tournament selection a recorded sizable numbers of wins against both fixed as well as random heuristic selection. Note that larger tournament sizes yielded wins for more environments compared to smaller tournament sizes. In contrast, sub-figures 6.6d, 6.6e, and 6.6f show

that the methods based on tournament selection showed strong performance against fixed heuristic selection, but did not manage to raise performance over that of random heuristic selection.

Observations about hyper-heuristics that employed roulette wheel selection

A number of methods (including random heuristic selection itself) relied on roulette wheel selection, as explained in section 3.4.6, specifically **RoulM**, **RoulX**, **ARank**, **NARank**, **AProp**, **NAProp**, **Comp**, and **Soft**. These methods generally yielded mostly white or light gray cells across the rows of figure 6.6 (indicating no noteworthy wins against other hyper-heuristics), and exhibited mostly dark gray cells in each relevant column (indicating strong losses against other hyper-heuristics). Figure 6.7 echoes these findings by showing how the roulette-based methods mostly showed low mean wins and high mean losses against the other methods in all six sub-figures.

All of the roulette-based hyper-heuristics that were configured with the *island* topology showed some level of success against fixed heuristic selection, generally showing darker shades of gray in the respective wins rows than in the matching losses columns. This pattern is not evident in the sub-figures for the *global* topology results, where the roulette-based hyper-heuristics almost never show wins and show a fair number of losses against **GFix**.

The roulette-based methods did not manage to significantly outperform random heuristic selection in any of the sub-figures in figure 6.6, but **Rand** never managed any wins against the roulette-based methods either. This indicates that the performance of roulette-based methods were indistinguishable from random heuristic selection, irrespective of whether mean or max feedback signals were used (as per equations (3.3) and (3.4), respectively, in section 3.4.6), whether ant-based learning was used or not, whether feedback was normalized using a soft-max function (see equation (3.20)), or whether competitive populations were used.

The overall best hyper-heuristics

At a minimum, any individual hyper-heuristic can only be considered good if it managed to outperform both the fixed and random selection control groups. This requirement

alone eliminates many topology and trigger combinations, as well as certain types of hyper-heuristics:

- Nearly half of the hyper-heuristics configurations that used the RT trigger and *global* topology managed to show strong performance against both control groups, as can be seen in sub-figure 6.6a. **ETour25**, **ETour2**, **HTour5X**, **HTour5M**, and **Freq5** were among the top performers in sub-figures 6.6a and 6.7a. A number of other hyper-heuristics showed comparably strong performance, namely **DProp**, **RLProp**, **RLFreq**, **ETour25**, **HTour2X**, and **HTour2M**. Sub-figure 6.7a supports these findings by showing how the listed hyper-heuristics had, on average, more wins than losses (compared to the other methods that shows sharply more losses than wins).

- The performance of the hyper-heuristics that used the *Global_PT* configuration, at a glance, showed similar visual patterns in sub-figure 6.6c compared to the corresponding *Global_RT* configuration in sub-figure 6.6a. Only a handful of hyper-heuristics can be considered superior to both control groups, namely **Etour25**, **HTour5X**, **HTour5M**, and **Freq5**. Sub-figure 6.7c confirms that these four hyper-heuristics were, on average, noticeably better than the other hyper-heuristics. Larger tournament sizes greatly increased the propensity for wins compared to smaller tournament sizes.

- Sub-figure 6.6e shows that most hyper-heuristics that used the *Global_ST* configuration did not perform well against either of the control groups. Only two hyper-heuristics, namely **ETour25**, and **HTour5X**, managed to achieve a notable number of wins against both **GFix** and **Rand**, but both hyper-heuristics also suffered a comparable number of losses against these control groups (indicating these methods were sensitive to the type of environment).

- Most hyper-heuristics that used the *Island_RT* configuration (see sub-figure 6.6b) show strong performance against **IFix** and, for most hyper-heuristics, inferior performance against **Rand**. However, **RLFreq**, **HTour5X**, **HTour5M**, and **Freq5** managed to outperform both control groups and showed strong performance against the other hyper-heuristics (but not necessarily each other). Sub-figure 6.7b

corroborates these findings: the four named hyper-heuristics had the four highest mean win scores and the among the lowest mean loss scores compared to all other hyper-heuristics.

- Similarly, hyper-heuristics using the *Island_PT* configuration showed strong performance against **IFix** in sub-figure 6.6d, but did not manage to appreciably outperform **Rand**. **RLfreq** is perhaps the only noteworthy exception, and achieved a high number of wins against **IFix** and a moderate number of wins against **Rand** while never losing to either **IFix** or **Rand**. None of the other hyper-heuristics that used the *Island_PT* configuration raised performance above the random selection control group, and cannot be considered as contenders for the overall best hyper-heuristic.

- Nearly every hyper-heuristic in sub-figure 6.6f (the exception being **ETour2** and **ETour25**) managed to outperform **IFix** across multiple environments, but were almost completely unable to yield wins against **Rand**. Neither **IFix** nor **Rand** ever recorded any wins against any hyper-heuristics either (with the exception of **ETour2**). Effectively, the *Island_ST* configuration failed to yield any hyper-heuristics that simultaneously performed better than both of the control groups.

Conclusions

The contenders for the best overall hyper-heuristic choice can therefor be constrained to the following options: **RLFreq**, **HTour5x**, **HTour5M**, and **Freq5**, when configured with the *Global_RT*, *Global_PT*, or *Island_RT* configurations, raised performance above both fixed and random selection control methods. These methods also managed to match or outperform most other hyper-heuristics. **ETour25** also raised performance above the control groups and other hyper-heuristics when using either the *Global_RT* or *Global_PT* configurations. The performance of **ETour25** was more susceptible to the type of environment at hand, and fixed heuristic allocation sometimes performed significantly better than **ETour25** for certain environments. **ETour2** configured with the *Global_RT* configuration outperformed both control groups and most other hyper-heuristics in many environments, while showing few losses against fixed heuristic selection. **ETour2** did not perform well with any other trigger or topology configurations.

In summary, the different hyper-heuristics showed varied performance relative to each other, fixed heuristic selection, and random heuristic selection. The choice of heuristic change trigger and neighborhood topology had a strong influence on the relative performance of hyper-heuristics. None of the hyper-heuristics that were configured with the ST trigger, regardless of neighborhood topology, raised performance above that of random selection.

RLFreq is an example of a hyper-heuristic that showed strong performance against both the fixed and random heuristic selection control methods, regardless of topology or trigger choices. **RLFreq** using the *island* topology and any trigger type consistently outperformed **IFix** in the majority of environments, while never losing to **IFix**. Similarly, **RLFreq** either outperformed or matched the performance of **Rand** when configured with the *island* topology and any trigger. **RLFreq** configured with the *global* topology was consistently among the top performers by always showing improved performance over both **Rand** and **GFix** across a fair number of environments. In these cases, **RLFreq** almost never lost to **Rand**, but was still susceptible to a small number of losses to **GFix** for a few environments. All in all, **RLFreq** was one of the investigated hyper-heuristics that consistently yielded good performance across many environments, regardless of neighborhood topology or heuristic change trigger type.

6.3.5 Research Question 4

What are the entity reassignment characteristics of each hyper-heuristic?

How a hyper-heuristic makes entity assignments over time is key to understand how computational resources are distributed across the pool of available heuristics. Different heuristic change triggers and neighborhood topologies may affect how a hyper-heuristic distributes entities. The assignment of entities to heuristics by any hyper-heuristic needs to be analyzed along a number of dimensions, namely

- the *balance* of entity assignments across the pool of heuristics over the entire algorithm run,
- the *number* of entities that are reassigned by the hyper-heuristic, and

- the *frequency* with which each hyper-heuristic configuration reassigns any entities to new heuristics.

The *balance* of entity assignments is measured using the heuristic space diversity (HSD) measure, or $\mathcal{N}(t)$, as discussed in section 3.4.3. The value $\mathcal{N}(t)$ is calculated after every algorithm iteration using equation (3.2). Values of $\mathcal{N}(t) \approx 1$ indicate perfect heuristic diversity, i.e. entities are spread out equally across all heuristics in a balanced fashion. On the other hand, values of $\mathcal{N}(t) \approx 0$ indicate that the majority of entities are assigned to a single heuristic in an unbalanced manner. A hyper-heuristic's HSD score can change every algorithm iteration as entities are reassigned between heuristics. Equation (3.2) yields a series of HSD values over time that may or may not vary depending on the actions taken by the hyper-heuristic.

As outlined in section 3.4.6, each heuristic in the pool had a minimum entity count of one for *global* topology configurations, and four when the *island* topology was used. Consequently, since all experiments were set up with 50 entities and nine heuristics, the minimum possible value for $\mathcal{N}(t)$ that could be achieved by any hyper-heuristic was $\mathcal{N}(t) = 0.3515$ for the *global* topology. This value resulted whenever all 41 free entities were assigned to any single heuristic while the remaining nine heuristics each had only one assigned entity. Similarly, the minimum possible value for $\mathcal{N}(t)$ that could be achieved when using the *island* topology was $\mathcal{N}(t) = 0.9031$, which occurred when all 14 free entities are assigned to any single heuristic.

The *number* of entities that are reassigned to different heuristics at time t expresses the magnitude of entity reassignments that occur. The subset $E^*(t) \subset E$ contains those entities in the parent population E that are triggered to be assigned new heuristics at time t, as explained in section 3.4.6. The hyper-heuristic may reassign any number of triggered entities in E^* to new heuristics, and may choose to leave many entity-to-heuristic assignments unchanged. Different hyper-heuristics do the reassignments in various ways, depending on the selection logic employed. The subset $E^{**} \subseteq E^* \subseteq E$ contains those entities that are assigned a different heuristic at time t. The number of entities that are reassigned to new heuristics at time t, $\upsilon(t)$, is simply the size of the subset $E^{**}(t)$, i.e.

$$\upsilon(t) = |E^{**}(t)| \tag{6.1}$$

The minimum value for $\upsilon(t)$ is zero, which occurs when no triggered entities are reassigned to new heuristics. The maximum value for $\upsilon(t)$ is $n_s - n_f$, where $n_s = |E|$ is the total number of entities in the HMHH parent population E, and n_f is the sum of the minimum entity counts across all n_h heuristics. Note that $n_f = n_h$ for the *global* topology where the minimum entity count is one, and $n_f = 4 \times n_h$ for the *island* topology where the minimum entity count is four.

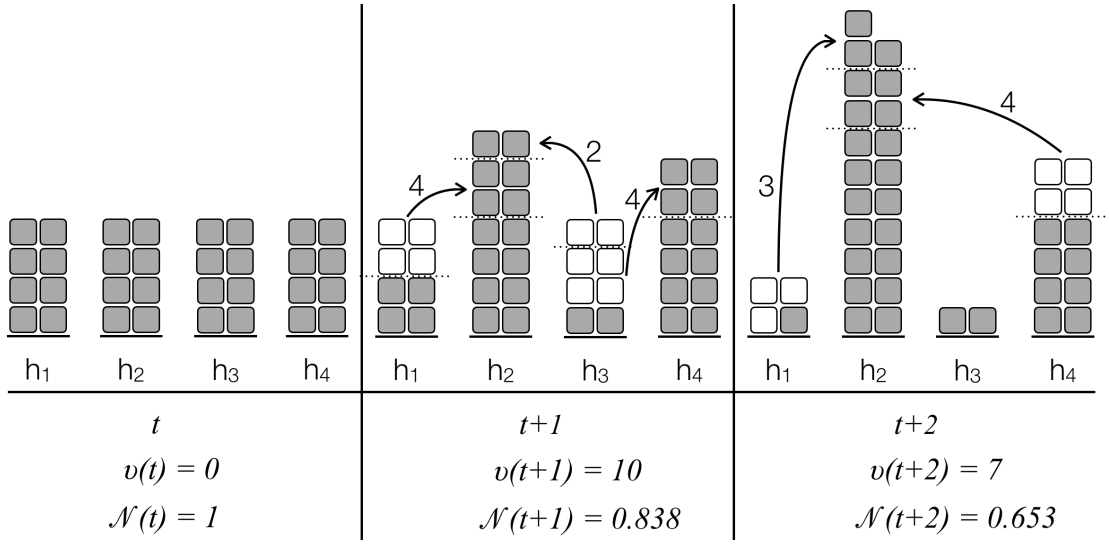
The balance and number of entity assignments, as measured by $\mathcal{N}(t)$ and $\upsilon(t)$, respectively, needs to be considered independently. It is possible for any given group of hyper-heuristics to have similar $\mathcal{N}(t)$ values over a period of time, yet have vastly different entity reassignment numbers. Figure 6.8 illustrates how $\mathcal{N}(t)$ and $\upsilon(t)$ are related. Both $\mathcal{N}(t)$ and $\upsilon(t)$ can change in different ways every algorithm iteration t. Consider any two hyper-heuristics that both have similar $\mathcal{N}(t)$ values across the entire algorithm run (i.e. where $\mathcal{N}(t) \approx 1$ over time). One hyper-heuristic could maintain relatively static entity-to-heuristic assignments, i.e., $\upsilon(t) \approx 0,\ \forall t$. The other hyper-heuristic might show large numbers of entity reassignments between heuristics, i.e., $\upsilon(t) \approx n_s,\ \forall t$. Since both of these hypothetical hyper-heuristics have the same HSD values where $\mathcal{N}(t) \approx 1,\ \forall t$, most of the entity reassignments "cancel each other out" from a HSD perspective (equal numbers of immigration and emigration of entities occur in a sense).

The *frequency* with which a hyper-heuristic reassigns entities to new heuristics is simply the proportion of all algorithm iterations $t \in T$ where entities were reassigned to new heuristics (regardless of the number of entities). In other words, the entity reassignment frequency, δ, is defined as

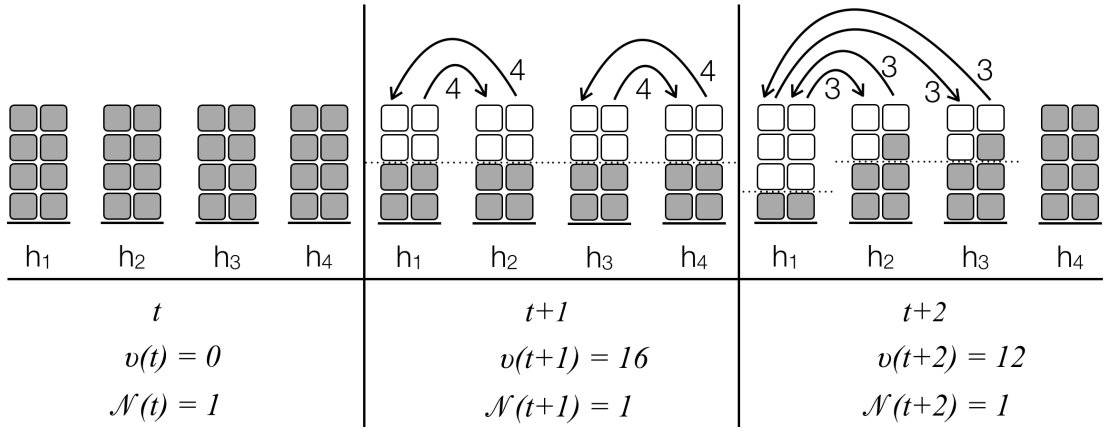
$$\delta = \frac{|T'|}{|T|} \tag{6.2}$$

where $T' \subseteq T$ represents the subset of algorithm iterations in T where at least one entity reassignment occurred (i.e. where $\upsilon(t) > 0$). A hyper-heuristic cannot reassign an entity if the entity is not first triggered by the heuristic trigger (the trigger thus sets an upper bound on δ). Once an entity is triggered, however, the hyper-heuristic selection logic may stipulate that the entity should not be reassigned to a new heuristic (which may yield variance in δ values across different algorithm runs). High variance in δ values across multiple algorithm run samples is indicative of selective behavior by a hyper-heuristic.

The measure φ reports the mean number of entities that were reassigned in an algo-



(a) The hyper-heuristic reassigns entities diverse across heuristics. $\mathcal{N}(t)$ and $\upsilon(t)$ change every iteration.



(b) The hyper-heuristic reassigns entities, but never changes the balance of entities between heuristics. $\mathcal{N}(t)$ remains constant while $\upsilon(t)$ changes every iteration.

Figure 6.8: Illustrative example of measuring heuristic space diversity using $\mathcal{N}(t)$ and the entity reassignment rate using $\upsilon(t)$ for four heuristics, h_1, h_2, h_3, and h_4, and 32 entities over three time steps t, $t+1$, and $t+2$. White blocks represent reassigned entities at time t.

rithm run, where the mean was computed relative to the proportion of iterations where entity reassignments occurred (i.e. T') as follows:

$$\varphi = \frac{\sum_{t' \in T'} \upsilon(t')}{|T'|} \tag{6.3}$$

In other words, algorithm iterations that did not contain any entity reassignments (i.e., $\upsilon(t) = 0$) do not skew the mean calculation. The value of φ depends on $\upsilon(t)$, which has different maximum possible values depending on whether the *island* or *global* topology is used, as explained above. Consequently, φ values are proportionately smaller for *island* configurations compared to the corresponding *global* configurations (all other variables being equal).

This research question characterizes the balance, number, and frequency of entity assignments of a number of exemplary hyper-heuristics. Figure 6.9 outlines how $\mathcal{N}(t)$ and $\upsilon(t)$ values were recorded at every iteration of each algorithm run. The median and inter-quartile range (IQR) of all $\mathcal{N}(t)$ values were subsequently computed for each individual algorithm run to provide a measure of centrality and spread for HSD values across the entire algorithm run. Median and IQR values were used since these statistical measures are less affected by outliers than mean and standard deviation values [162]. The process outlined in figure 6.9 was repeated for each of the 71 sample runs for each hyper-heuristic for each of the 27 environments. This yielded 1 917 median and IQR values for $\mathcal{N}(t)$ and 1 917 δ and φ values per hyper-heuristic, for each of the six possible configurations of *global* and *island* topologies together with random triggers (RT), periodic triggers (PT), and stagnation triggers (ST).

The sub-sections below present and discuss summary charts of HSD and entity reassignment rates for a number of exemplary hyper-heuristics. Each chart comprises of 12 scatter-plot panels that collectively present information for the different trigger and topology combinations for a specific hyper-heuristic. Each scatter-plot panel contains nine subplots that show sample values for the 27 different environment types. Each point in a scatter-plot represents a single algorithm run. *Circular*, *linear*, or *random* environments are represented across columns, *Type I*, *Type II*, or *Type III* environments are represented across rows, and *abrupt*, *progressive*, or *chaotic* environments are shown as different color and marker symbol combinations. The top six plots highlight the median

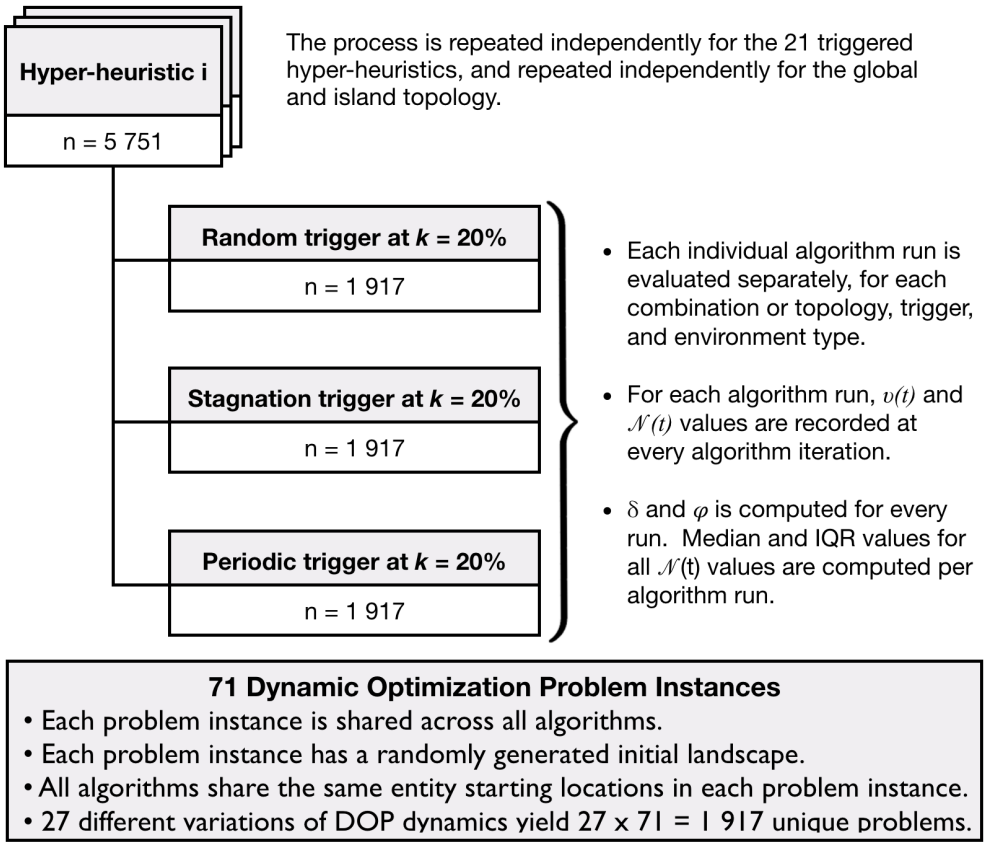


Figure 6.9: Focus of the HSD and entity reassignment analysis for each hyper-heuristic.

and IQR values of $\mathcal{N}(t)$ for each individual run, while the bottom six plots show the relationship between δ and φ (the bottom six plots are drawn with a white background for easy differentiation from the top six plots).

The goal of the approach above is to:

- Illustrate how different combinations of neighborhood topology and heuristic change triggers can drastically alter the operation of certain hyper-heuristics, while not necessarily having any measurable effect on other hyper-heuristics.
- Show how each hyper-heuristic may or may not behave differently across different samples for the same environment, i.e., investigate the variance of problem samples

for the same type of environment.

- Reveal how certain hyper-heuristics may show highly varied behavior across different environments, while other hyper-heuristics may show predictable behavior across all problems for all environments. In other words, the goal is to investigate the variance of problem samples between different environment types.
- Bring to light how parameters value choices (such tournament size for example) affects the behavior of a hyper-heuristic.

The goal is not to exhaustively discuss the behavior of every investigated hyper-heuristic, nor to identify the detailed behavior patterns for each of the individual 27 different environments – this is left as future work.

The behavior of the following hyper-heuristics, for all combinations of triggers and topologies, are examined in more detail in the subsections below:

1. **Rand**, where the aim is to study the behavior of a control group approach that does not rely on intelligence.
2. **RLFreq**, to showcase the behavior of a hyper-heuristic that employs intelligent selection and learning.
3. Hyper-heuristics with HSD and entity reassignment behavior that is similar to **Rand**.
4. **ETour2** and **ETour25**, that both rely on entity tournament selection.
5. **Comp**, which did not perform well relative to the other methods or the control groups.

The behavior of random heuristic selection

Rand applies no intelligence to reassign triggered entities to new heuristics. Consequently, since all heuristics always have an equal probability of being selected, the expectation is that **Rand** would maintain nearly perfect heuristic space diversity over time (i.e., $\mathcal{N}(t) \approx 1, \forall t \in T$). Since no intelligence is used to make assignments, another

expectation is that entity reassignment rates would be relatively uniform over time, similar between problem instances, and similar across different environments. That is, φ variance would be small between samples within each environment and across different environments.

Figure 6.10 shows that the actual behavior for **Rand** matches both these expectations. All six HSD plots show high median and low IQR values for $\mathcal{N}(t)$ values, indicating that **Rand** maintained very balanced heuristic assignments throughout each algorithm run. Five of the six HSD plots look identical, while the sixth plot (for the *Global_ST* configuration) strongly resembles the other five plots.

The six entity reassignment plots each show different δ and φ values across configurations, which indicates that alternative topology and trigger configurations yielded very different behavior in terms of the frequency and volume of entity reassignments. For each configuration, there was a noticeable difference in the entity reassignment behavior for the *abrupt* versus the *progressive* and *chaotic* environments. The similarity of the subplots for each configuration show that, for each environment, **Rand** was impartial to the particular problem instance under consideration. The ST configurations are an exception, and will be discussed in more detail below.

The behavior of different topology and trigger configurations for **Rand**, as visible in figure 6.10, is described in more detail in the subsections below.

Rand using the *Global_RT* configuration: The HSD plot shows how the median $\mathcal{N}(t)$ values were approximately 1 for every sample, while the IQR spread values of $\mathcal{N}(t)$ were close to 0 for all samples. These observations hold for all environments. This highlights a very narrow distribution of $\mathcal{N}(t)$ values over time that rarely deviated from the median value of 1, which indicates balanced and impartial heuristic assignments throughout each algorithm run in every problem instance in every environment.

The entity reassignment plot reveals that δ values were very high for all samples across all environments, indicating that entities were reassigned to new heuristics practically every algorithm iteration. Values for φ reveal that different numbers of entities were reassigned for the *abrupt* versus the *progressive* and *chaotic* environments. The heuristic change frequency parameter was set to $k = 20\%$ of the environment change frequency, which yielded more frequent opportunities for heuristic changes to be made for *progressive*

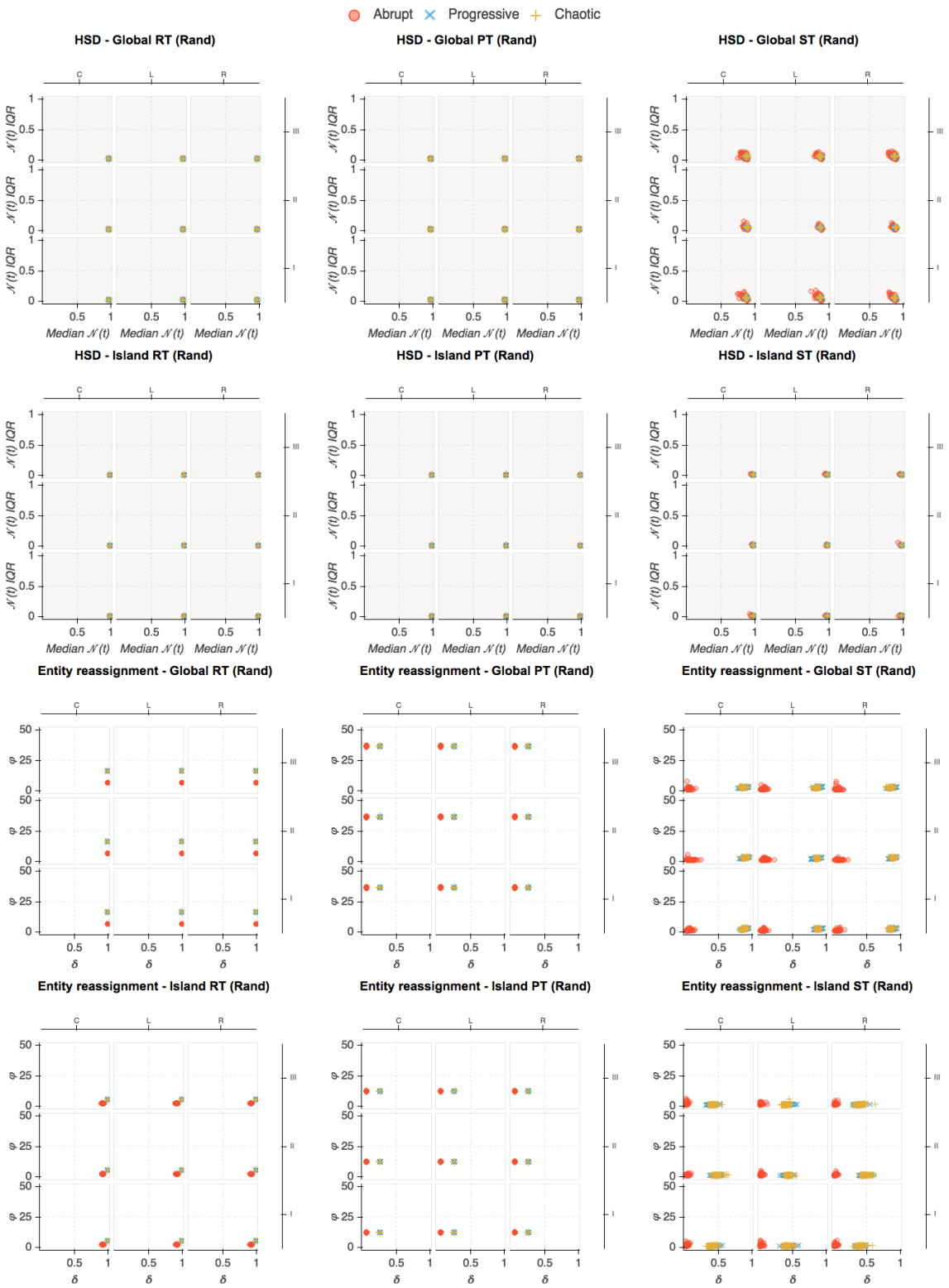


Figure 6.10: HSD and entity reassignment rate analysis for **Rand**

and *chaotic* environments (i.e. 20% of 20 iterations) than for the *abrupt* environments (i.e. 20% of 100 iterations)[1]. Consequently, each entity had a higher probability to be triggered by the RT trigger for *progressive* and *chaotic* environments than for *abrupt* environments, causing higher φ values for the former environments.

Rand using the *Island_RT* configuration: The HSD behavior was identical to the *Global_RT* configuration, which shows that the choice of topology did not affect the heuristic space diversity behavior for **Rand** when the RT trigger was used.

Values for φ were lower for all environments compared to that of the *Global_RT* configuration, since the *island* topology had fewer free entities to reassign, as discussed above. The difference in φ values between the *abrupt* and the *progressive* and *chaotic* environments were, again, due to the choice of $k = 20\%$ yielding different triggering probabilities for the RT trigger. Values for δ were identical as for the *Global_RT* configuration for all *progressive* and *chaotic* environment samples, but the *abrupt* environment samples, however, showed noticeably lower δ values. The *abrupt* environment had a lower probability of triggering each entity, and only 14 of the 50 entities that could be triggered were free for reassignment. In aggregate, slightly fewer opportunities to move slightly fewer entities resulted in a lower δ values for the *abrupt* versus the *progressive* and *chaotic* environments.

Rand using the *Global_PT* and *Island_PT* configurations: The HSD behavior of each configuration was exactly the same as that of the *Global_RT* and *Island_RT* configurations, respectively. This confirms that the choice between PT or RT trigger did not affect the heuristic space diversity behavior for **Rand**.

In contrast, the entity reassignment plots for both the *Global_PT* and *Island_PT* configurations show much lower δ values than their respective RT counterparts. This is a result of the fact that the PT trigger periodically triggered all entities simultaneously at specific algorithm iterations (i.e. based on $k = 20\%$ of the environment change frequency). This subsequently yielded much lower δ values compared to the RT trigger that tended to trigger a small number of entities nearly every iteration. Similar to the RT cases, a discrepancy is visible between the δ values of the samples for the *chaotic*

[1]Refer to Table 2.2

and *progressive* environments and the *abrupt* environment. This was, again, due to the fact that *abrupt* environments had longer change periods, which resulted in less frequent heuristic changes. The size of the difference in δ values is identical for the *Global_PT* and *Island_PT* configurations, since the PT trigger was unaffected by the choice of topology.

Values for φ were higher than for the corresponding RT trigger configurations, which is to be expected given that all entities were triggered simultaneously by the PT trigger. The φ values for the *Island_PT* configurations were lower than the corresponding *Global_PT* configuration, which was due to the hyper-heuristic enforcing the larger minimum entity count of four entities per heuristic for the *island* topology. Values for φ were approximately 37 for the *Global_PT* configuration (the maximum was 41) and 14 for the *Island_PT* configuration (where the maximum was 14). This resulted since **Rand** used a uniform probability of assignment over nine heuristics, which caused roughly 88% of triggered entities to be reassigned (on average).

Rand using the *Global_ST* configuration: The *Global_ST* HSD plot is roughly similar to the other five HSD plots, but shows a number of departures from the otherwise steadfast trends that are present in the other plots:

- Increased variance is visible in the *abrupt* environment data in all nine subplots (visible as red circles), which indicates that the *Global_ST* configuration was more prone to create imbalanced heuristic assignments across different samples for *abrupt* environments. In contrast, the tight groupings of *progressive* and *chaotic* values across all samples in all nine subplots show that the *Global_ST* configuration consistently yielded similar behavior for these environments.

- The median $\mathcal{N}(t)$ values for all samples were closer to 0.85 than 1, while the corresponding IQR spread of $\mathcal{N}(t)$ values were closer to 0.15 than 0. These readings indicate a wider distribution of $\mathcal{N}(t)$ values for each environment, which shows that the *Global_ST* configuration created more imbalanced heuristic assignments within each individual algorithm run.

- The entity reassignment plot for *Global_ST* shows noticeably higher variance in δ values across samples than is present for the other configurations. Most variation was contained inside a relatively defined range for each environment.

- All δ values for *abrupt* environments were low, in sharp contrast to the high δ values for *progressive* and *chaotic* environments. The heuristic change frequency parameter was set to $k = 20\%$ of the environment change frequency, which resulted in a much lower tolerance of what was deemed stagnating behavior for entities in *progressive* and *chaotic* environments. For the ST trigger, this yielded more frequent entity reassignments for *progressive* and *chaotic* environments and relatively fewer entity assignments for *abrupt* environments.

- The φ values were always very low and relatively consistent for all samples for each environment, as well across different environments. This shows that a consistently small number of entities were reassigned during iterations that saw reassignments, regardless of the type of environment or specific problem instance.

Rand using the *Island_ST* configuration: The HSD plot strongly resembles the HSD plots for the *Global_RT*, *Global_PT*, *Island_RT*, and *Island_PT* configurations. There is a very weak indication of higher variance of $\mathcal{N}(t)$ values for *abrupt* environment samples (reminiscent of the *Global_ST* configuration). However, the majority of samples for all environments were tightly grouped, indicating that the *Island_ST* configuration had the same HSD behavior regardless of the type of environment.

Similar to the *Global_ST* configuration, the entity reassignment plot shows a discrepancy between *progressive* and *chaotic* environment δ values and *abrupt* environment δ values. This difference was, again, due to shorter stagnation tolerances for *progressive* and *chaotic* environments, which affected entity reassignment frequencies. However, the δ values for *progressive* and *chaotic* environment samples were much lower than for the *Global_ST* configuration. The variance of δ between *abrupt* environment samples was also noticeably lower than for the corresponding *Global_ST* configuration. Both these observations indicate that there was significantly less stagnation of entities when using the *Island_ST* configuration compared to using the *Global_ST* configuration.

Similar to the *Global_ST* configuration, φ values were always very low for all samples across all environments, which indicates that the configuration had the same entity reassignment volume and frequency for any sample in any environment.

Conclusions about Rand: As one of the control group methods, the behavior of **Rand** epitomizes the overall operation of a hyper-heuristic that does not apply any intelligence in selecting new heuristics for entities. With the exception of the ST trigger configurations (which, arguably, apply intelligence in deciding how to trigger heuristic changes based on observed fitness values), **Rand** yields predictable and consistent HSD and entity reassignment behavior for every problem sample across all 27 environment types. Hyper-heuristics that apply alternative selection logic may or may not show similar behavior to **Rand**, which is explored further in the next sections.

The behavior of frequency improvement reinforcement learning selection

As described in section 3.4.6, **RLFreq** increases or decreases a rank score for each heuristic based on the observed frequency of improvements made by entities assigned to each heuristic. **RLFreq** subsequently assigns all triggered entities to the highest ranked heuristic, while always honoring the minimum entity counts of each heuristic (one for the *global* topology and four for the *island* topology).

Previous research questions showed how **RLFreq** was consistently among the top performing hyper-heuristics across multiple trigger and topology configurations. While the hyper-heuristic was not the outright best for all environments at all times, **RLFreq** did manage to match or outperform **Rand** across the majority of environments. **RLFreq** also consistently matched or outperformed the majority of other hyper-heuristics (particularly when using the *island* topology), recording almost no losses in any environments. **RLFreq** also showed a strong proclivity in section 6.3.2 and 6.3.3 to raise performance above using the stand-alone or speciated versions of the heuristics that make up the heuristic pool, across most trigger and topology configurations. As such, **RLFreq** provides good insights into the HSD and entity reassignment behavior of a generally successful hyper-heuristic strategy.

Figure 6.11 shows the heuristic space diversity and entity reassignment behavior for **RLFreq** across all environment samples for all six topology and trigger configurations. At a glance, the plots in figure 6.11 appear substantially different to the corresponding plots for **Rand** in figure 6.10. The HSD plots show drastically different behaviors across different trigger and topology types. Entity reassignment rates were also highly

varied based on the configuration. Sample variance was much more noticeable in almost every plot in figure 6.11, and sample variances and value ranges were different across environments in many configurations.

The following observations below about the behavior of different trigger and topology configurations for **RLFreq** are visible in figure 6.11.

RLFreq using the *Global_RT* configuration: A striking pattern in the HSD plot is the vertical line where the median $\mathcal{N}(t)$ value was approximately 0.35 (the minimum possible HSD value). This shows that the configuration assigned all available entities to the single top-ranked heuristic at least 50% of the time. The IQR spread of $\mathcal{N}(t)$ values ranged between $[0, 0.5]$, which indicates that the configuration alternated between assigning all triggered entities to the single top-ranked heuristic all of the time (low IQR values) versus yielding more varied heuristic assignments with balanced HSD behavior (higher IQR values). Each HSD subplot shows a number of outliers along relatively predictable paths. In that sense, it is clearly visible from the HSD plot that the *Global_RT* configuration showed adaptive behavior based on the problem instance at hand.

The entity reassignment plot shows that the sample variance for δ values was high in every environment, and that the variance pattern was different across environments. For many environments, φ values, however, remained largely unchanged as δ values varied, visible as flat horizontal lines in the entity reassignment plot. Exceptions exists among a handful of environments, i.e. P3R that shows variance in both δ and φ values at the left side of the subplot (visible as a grouping of points compared to the straight lines of other environments). One pattern stands out clearly, namely the clear differentiation between the φ values for *abrupt*, *progressive*, and *chaotic* environment samples. Similar to **Rand**, this variation is explained by how the different change period of the landscape affects the heuristic selection triggering frequency at $k = 20\%$. The relatively constant φ trend for each environment type reveals how the *Global_RT* configuration often reassigned the same number of entities, whether that happened frequently (i.e. high δ values) or infrequent (i.e. low δ values).

The vertical and horizontal lines that are visible in, respectively, the HSD and entity reassignment plots raises the question of whether there was any positive or negative correlation between increased δ values and increased IQR spreads for $\mathcal{N}(t)$. Figure 6.12

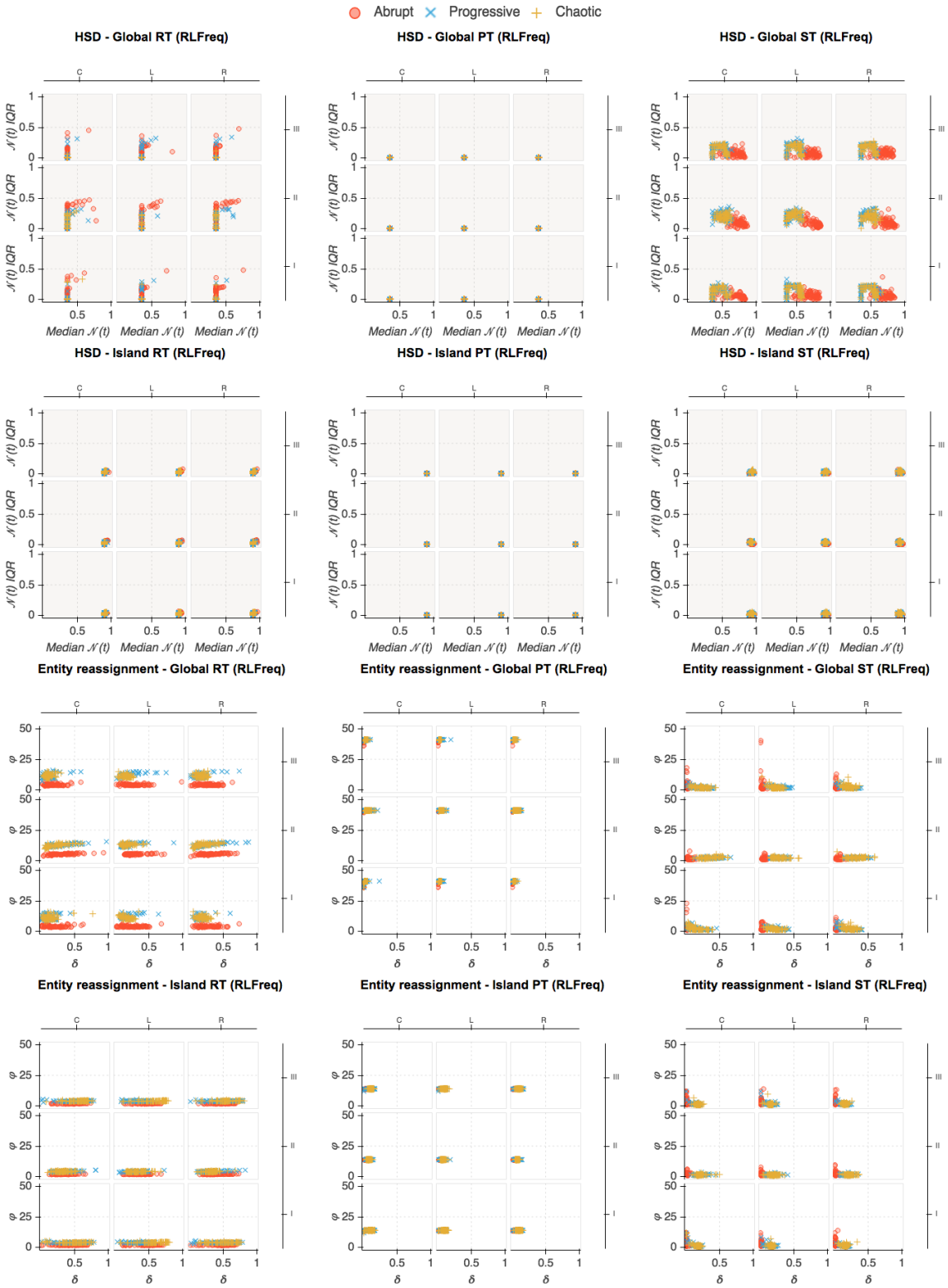


Figure 6.11: HSD and entity reassignment rate analysis for **RLFreq**

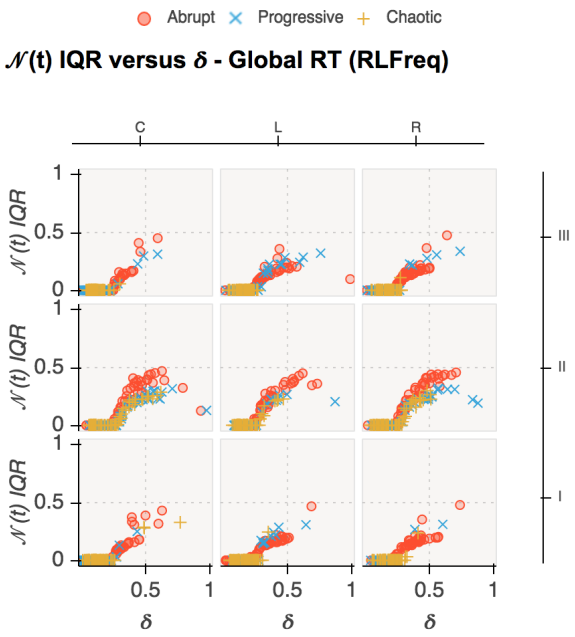
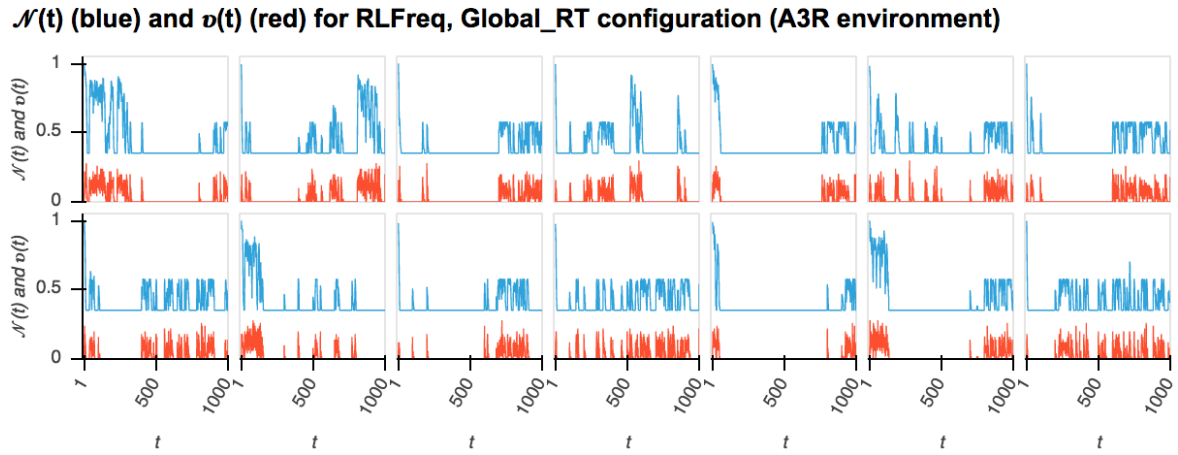


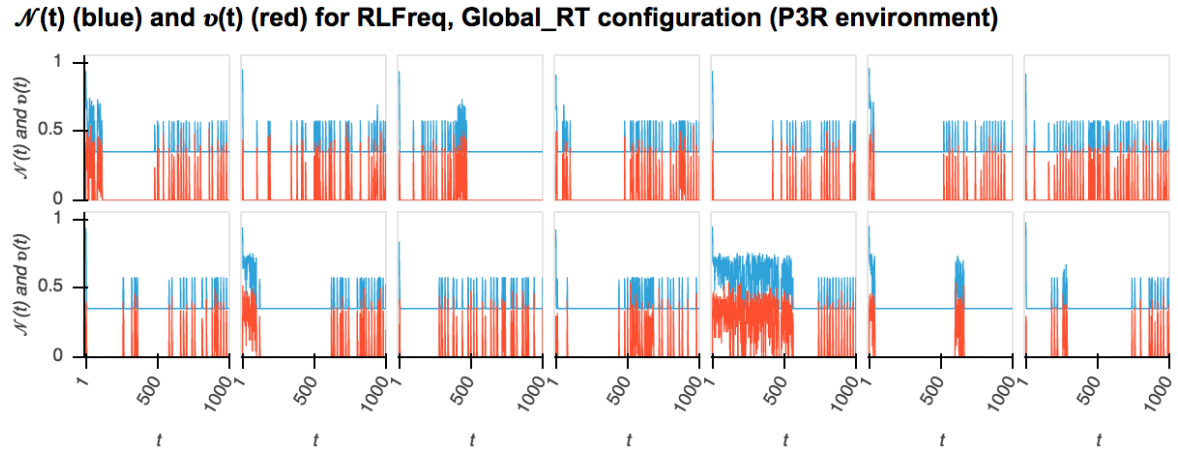
Figure 6.12: Median $\mathcal{N}(t)$ versus δ values for each environment for **RLfreq** using the *global* topology and RT trigger.

shows a clear positive correlation between the entity reassignment frequency and more diversified HSD behavior. Higher δ values were, generally, associated with constant $\mathcal{N}(t)$ behavior up until roughly $\delta \approx 0.3$, after which there was a clear positive linear relationship between increasing δ values and wider IQR ranges for $\mathcal{N}(t)$. Since the median $\mathcal{N}(t)$ value was approximately 0.35 (the minimum possible value) for the majority of samples, the wider IQR ranges with $\delta > 0.3$ clearly shows that **RLFreq** using the *Global_RT* configuration exhibited adaptive, exploratory behavior for some problem samples, and not for others. These observations are testimony to the fact that the *Global_RT* configuration employed intelligent selection behavior. The configuration exhibited unique behavior depending on the type of environment as well as the particular problem instance under consideration.

Another key question is whether the *Global_RT* configuration could re-diversify entities across heuristics after all entities had been assigned to a single heuristic (i.e. escape from local minima in heuristic space). In other words, did $\mathcal{N}(t)$ values ever increase again once values hit the minimum value of 0.35? Figure 6.13a and 6.13b show the $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R



(a) A3R



(b) P3R

Figure 6.13: $\mathcal{N}(t)$ and normalized $v(t)$ for **RLFreq** using the *global* topology and RT trigger.

and P3R environments, respectively, for the *Global_RT* configuration. The plots visually show that $\mathcal{N}(t)$ values did rise and fall constantly over time, indicating adaptive HSD behavior by the hyper-heuristic. The plots also show how *progressive* environment samples had higher numbers of entity reassignments than *abrupt* environments, illustrating the higher φ values presented in figure 6.11.

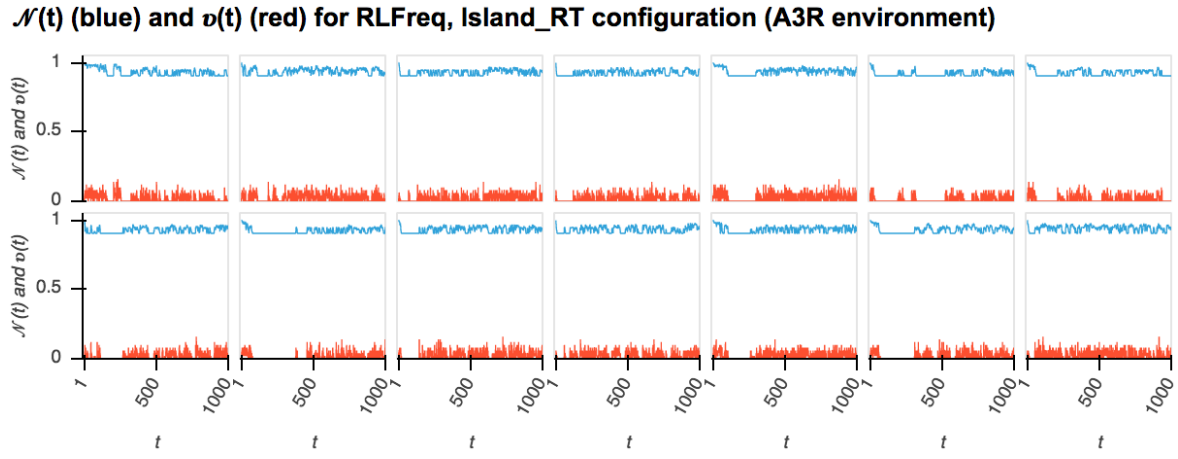
RLFreq using the *Island_RT* configuration: The *Island_RT* configuration showed variance across the entire spectrum of possible $\mathcal{N}(t)$ values: the bulk of median $\mathcal{N}(t)$ values were approximately 0.9 (the minimum possible value for the *island* topology),

while the IQR spread was approximately 0.1. This reveals that, even though the value ranges were much smaller, the *Island_RT* configuration behaved similar to the *Global_RT* configuration by mostly assigning all free entities to a single heuristic, yet still managing to periodically diversify entities across multiple heuristics when needed. Using the *island* topology greatly impacted the ability of the hyper-heuristic to exploit any single heuristic.

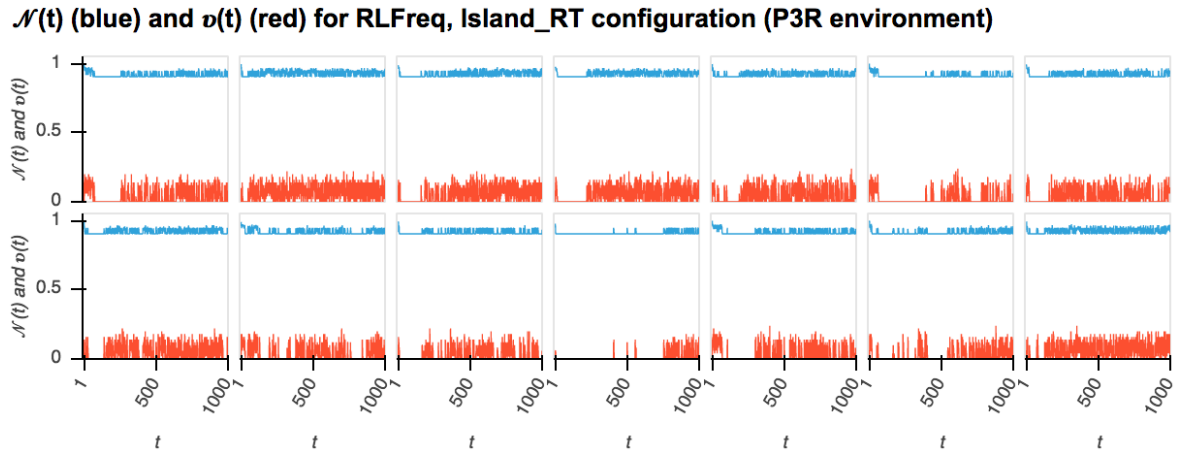
The entity reassignment plot shows a wide range of δ values across samples for each type of environment, indicating very adaptive behavior in deciding whether or not triggered entities should be reassigned or not. The lower φ variance compared to the *Global_RT* configuration was, again, attributable to the fact that the *island* topology has fewer available entities. Similar to the *Global_RT* configuration, the samples for the *abrupt*, *progressive*, and *chaotic* environments were separated into horizontal lines in the entity reassignment plot, although the variance was noticeably lower than for the the *Global_RT* configuration.

As another illustrative example, figure 6.14 shows the $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments, respectively, for the *Island_RT* configuration. The plots visually show how HSD values were repeatedly able to escape local minima in heuristic space. A comparison of the plots with figure 6.13 gives a visual understanding of how both $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values were much more restricted when using the *island* topology compared to using the *global* topology.

RLFreq using the *Global_PT* and *Island_PT* configurations: All samples across all environments always yielded median $\mathcal{N}(t)$ values of approximately 0.35 (the minimum possible HSD value). Contrary to the *Global_RT* configuration, however, the IQR spread values for $\mathcal{N}(t)$ of all samples were always close to 0. These results indicate an extremely narrow distribution of HSD values where all available entities were always and predictably assigned to a single heuristic the majority of the time during each individual run. Similar to the *Global_PT* configuration, the *Island_PT* configuration yielded a very narrow HSD distribution: median $\mathcal{N}(t)$ values were always near the minimum possible value (0.9030 for the *island* topology) and the IQR spread was approximately 0. This reveals that **RLFreq** showed similar HSD behavior for both the *island* and *global* topologies when using the PT trigger.



(a) A3R



(b) P3R

Figure 6.14: $\mathcal{N}(t)$ and normalized $v(t)$ for **RLFreq** using the *island* topology and RT trigger.

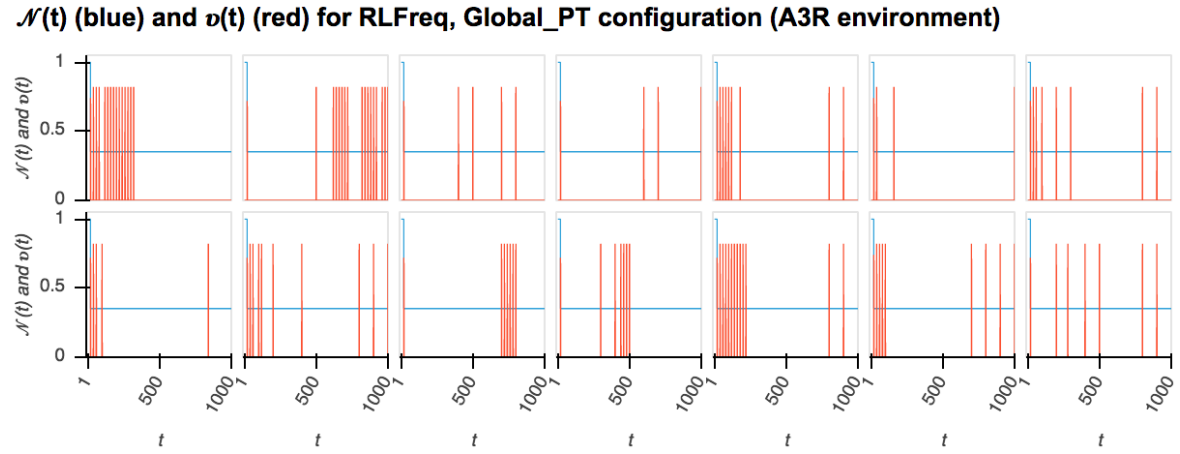
Since all entities were triggered together on a periodic basis, the maximum possible δ value was approximately 0.25 (for *progressive* and *abrupt* environments) and approximately 0.1 (for *abrupt* environments). The horizontal lines visible for δ values in the entity reassignment plot show that entity reassignment frequencies varied across the entire possible range. This indicates selective behavior by the hyper-heuristics: if the entities were already assigned to the top ranked heuristic, no reassignments would occur that particular iteration. In that sense, the *Global_PT* configuration was reactive to the particular problem instance at hand. The high entity reassignment rates, where φ were

in the range $[35, 41]$, reveal that the majority of entities were reassigned every time reassignments occurred. The low HSD values combined with infrequent but severely large entity reassignments show that *Global_PT* alternated between rarely changing heuristic assignments for some algorithm runs, to always reassigning all entities to a single new heuristic every change period. The entity reassignment behavior for the *Island_PT* configuration was also similar to that of the *Global_PT* configuration in that entity reassignment frequencies varied across the entire possible range of values for each environment, and similar conclusions can be made.

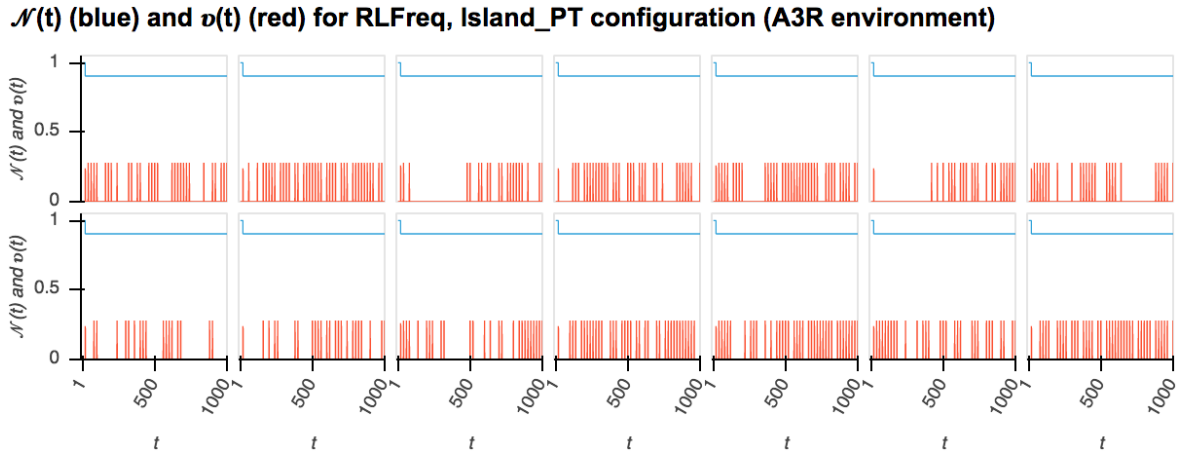
Figure 6.15 shows the $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R environment for the *global* and *island* topologies, respectively. The HSD values remain constant at $\mathcal{N}(t) \approx 0.35$ over all iterations. High entity reassignment rates show that all entities were reassigned every time a new heuristic was selected. Large periods of inactivity (visible as horizontal blue and red lines) clearly show where the hyper-heuristic did not change heuristic assignments (despite all entities being triggered periodically).

RLFreq using the *Global_ST* and *Island_ST* configurations: The HSD plot shows a wide spectrum of behavior. Visually, the samples appear to form a crude rectangular shape in the HSD plot. The *progressive* and *chaotic* environment samples are clearly separated from the *abrupt* environment samples. All environments samples have IQR spread values for $\mathcal{N}(t)$ that varied widely across the approximate range $[0, 0.4]$. This reveals how the *Global_ST* configuration showed every possible behavior along two dimensions. The first dimension ranged between well-balanced entity assignments across heuristics (high $\mathcal{N}(t)$ values) versus complete dominance by a single heuristic (low $\mathcal{N}(t)$ values), while the second dimension ranges between consistent HSD behavior over time (low IQR values for $\mathcal{N}(t)$) versus alternating HSD behavior over time (high IQR values for $\mathcal{N}(t)$).

Considering that the minimum possible value for $\mathcal{N}(t)$ was 0.9030 for the *island* topology, the HSD plot for the *Island_ST* configuration shows comparable behavior to the *Global_ST* configuration. Median $\mathcal{N}(t)$ values varied across the entire viable range, as visibly indicated by the spread of samples across the x-axis along with corroborating IQR values of approximately 0.1.



(a) *Global* topology



(b) *Island* topology

Figure 6.15: $\mathcal{N}(t)$ and normalized $v(t)$ for **RLFreq** using the PT trigger (both topologies).

The entity reassignment plot shows that δ values varied widely for each environment. A clear pattern is visible where *abrupt* environments yielded lower δ values (as a result of a higher tolerance for stagnation) while *progressive* and *chaotic* environment samples showed higher and more varied δ values (as a result of lower tolerances for stagnation). The corresponding φ values were low, with a tendency to increase slightly as δ values decreased (this was especially prominent in *Type I* environments).

The entity reassignment behavior for the *Island_ST* configuration was similar to the *Global_ST* configuration, although δ values did not vary quite as widely. The lower

variance for δ values reveals that the ST trigger was activated less frequently when using the *island* topology than when using the *global* topology. Further study is needed to determine if the lower stagnation rate was due to slower, steadier convergence of entities in each heuristic's disjoint sub-population compared to the sub-populations of the *Global_ST* configuration that had purview across the entire global population.

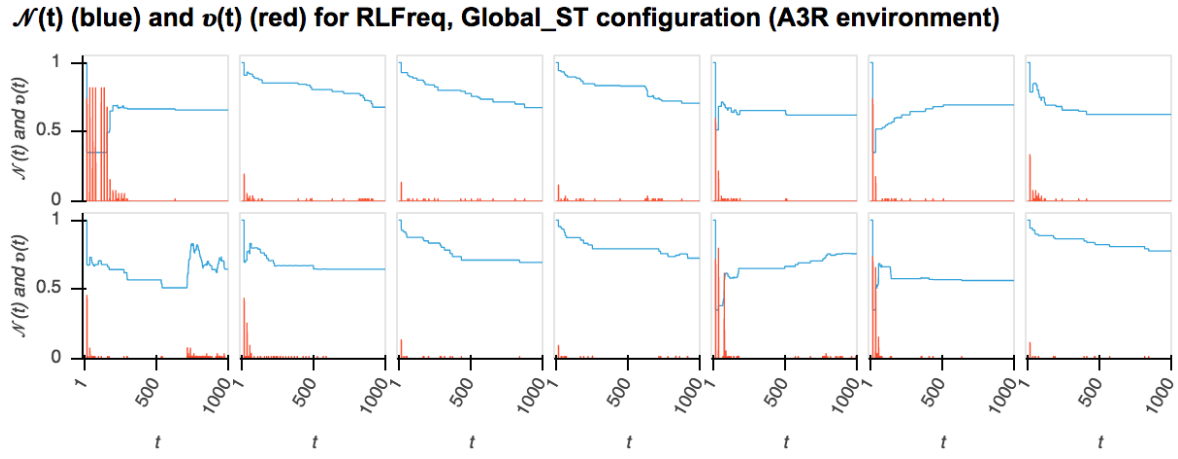
Figure 6.16a and 6.16b show the $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments, respectively, for the *Global_ST* configuration. HSD and entity reassignment behavior within each sample and across different samples match the wide range of behaviors described in the paragraphs above. The corresponding plots for the *island* topology (not illustrated here) appear similar, albeit with smaller $\mathcal{N}(t)$ and $\upsilon(t)$ ranges.

For the *Global_ST* configuration, the question arises whether there was any correlation between the highly varied HSD behavior and the stratified variance pattern that was visible in δ values. Figure 6.17 shows scatter-plots for the median $\mathcal{N}(t)$ values versus δ values for each environment for the *Global_ST* configuration. There was noticeably less linear correlation between the variables compare to the *Global_RT* configuration data in figure 6.12, and many subplots show a distinctive inverted *"U"*-shaped relationship. This reveals that there was no clear-cut linear relationship between increased entity reassignment frequency and heuristic space diversity.

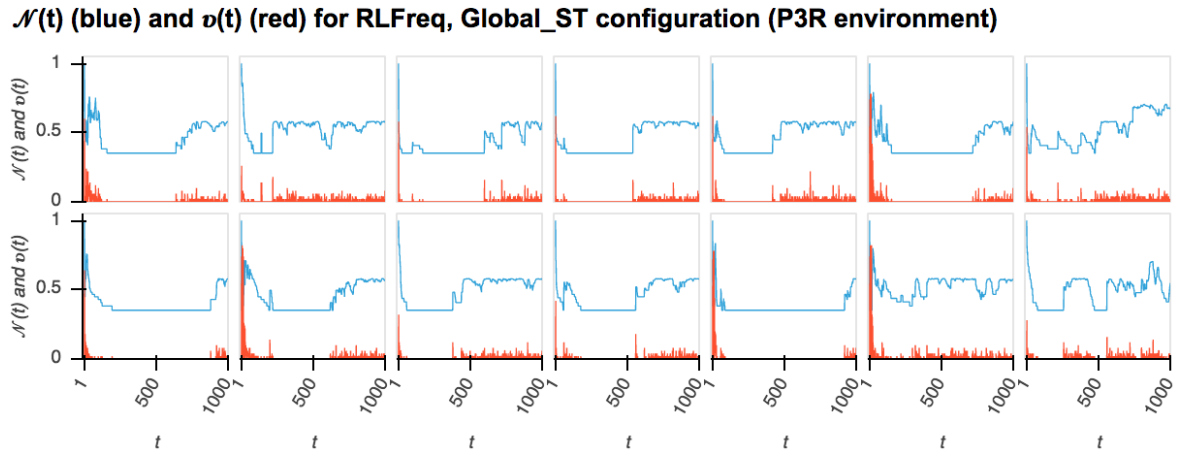
Conclusions about RLFreq: The **RLFreq** hyper-heuristic showed markedly different behavior for every choice of topology and trigger compared to the corresponding results for **Rand**. Overall, **RLfreq** showed a strong tendency to adapt to the particular problem instance under consideration. Future work should focus on correlating the behavior of **RLFreq** with environment change characteristics.

Hyper-heuristics with similar behavior to random heuristic selection

A number of hyper-heuristics showed nearly identical behavior to **Rand** with respect to HSD and entity reassignment rates, namely **Perm**, **Soft**, **NARank**, and **ARank** (refer to figures A.1, A.2, A.11 and A.12 in appendix A, respectively). These hyper-heuristics all maintained HSD scores, entity reassignment frequencies, and entity reassignment volumes that were approximately equal to the values recorded for **Rand** for corresponding



(a) A3R



(b) P3R

Figure 6.16: $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **RLFreq** using the *global* topology and ST trigger.

topology and trigger configurations.

Section 3.4.6 outlines how each hyper-heuristic employed decidedly different types of selection logic to reassign entities. This means that, even though these hyper-heuristics exhibited similar $\mathcal{N}(t)$, δ, and φ values to that of **Rand**, the fundamentally different selection logic ensured that the specific entities being reassigned would be different between methods. The HSD and entity reassignment plots cannot express this information, and further study is needed to track the individual trajectories of entities across heuristics and correlate these behaviors between different hyper-heuristics.

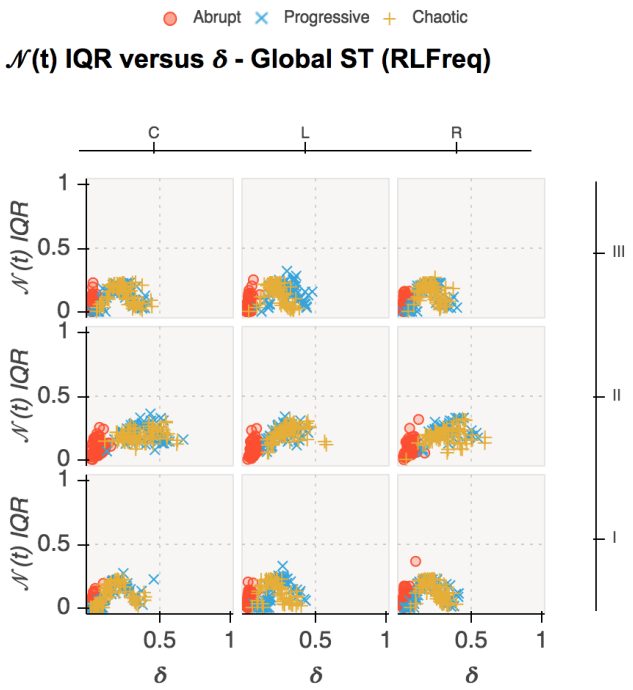


Figure 6.17: Median $\mathcal{N}(t)$ versus δ values for each environment for **RLfreq** using the *global* topology and ST trigger.

However, none of the four hyper-heuristics recorded wins or losses against **Rand** in any environment, as visible in figure 6.6. Figure 6.7 shows that **Rand** and these four hyper-heuristics also had comparable mean wins and losses for each different trigger and topology configuration. As such, both the performance and behavioral findings suggest that there was no statistically significant difference in performance or behavior between these hyper-heuristics.

The results for various tournament selection-based hyper-heuristics (as outlined in section 3.4.6) reveal how progressively larger tournament sizes resulted in steadily lower median $\mathcal{N}(t)$ values with wider IQR spreads, as well as larger variances between problem samples. The specific groups of hyper-heuristics that showed this trend are

- **Rand**, **HTour2X**, and **HTour5X** (see figures A.13 and A.14),
- **Rand**, **HTour2M**, and **HTour5M** (see figures A.15 and A.16), and
- **Rand**, **Freq2**, and **Freq5** (see figures A.17 and A.18).

Engelbrecht [60] remarks that random selection is logically equivalent to tournament selection with a tournament size of one. In that sense, the results of **Rand** can also be compared against these various groups of hyper-heuristics that employed tournament selection.

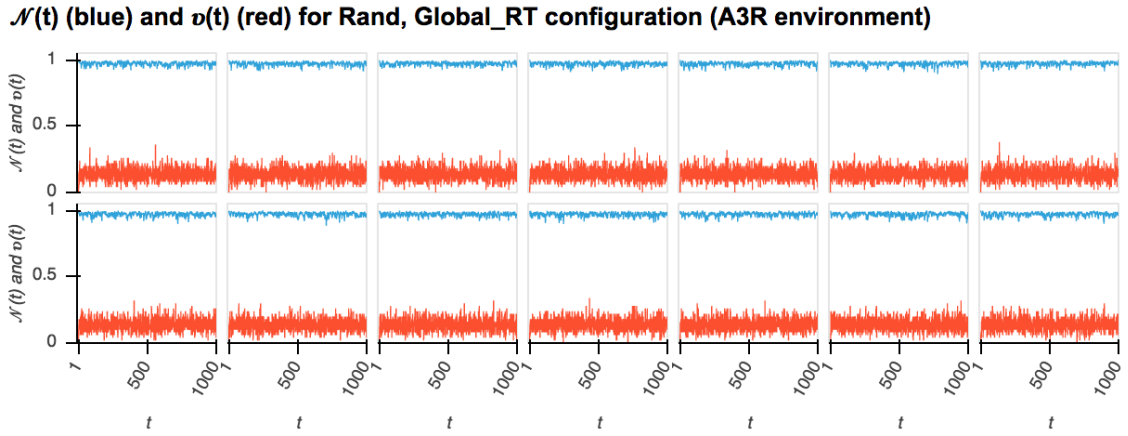
In each respective group outlined above, higher tournament sizes created greater imbalances between the number of entities that were assigned to each heuristic over time (i.e. lower $\mathcal{N}(t)$ values). This is a direct measurement indicating that the selection pressure of the hyper-heuristic increased with larger tournament sizes, driving the hyper-heuristic from balanced assignments across heuristics towards favoring one heuristic over others. The pattern was only present when the *global* topology was used, and the HSD plots for the *island* configurations appear identical across different tournament sizes in each group. Entity reassignment frequencies, δ, remained nearly identical for each group across all trigger and topology combinations except for the *Global_ST* configuration, where *progressive* and *chaotic* environment samples steadily yielded lower δ values as the tournament size increased. Values for φ remained relatively unchanged.

Figure 6.18 shows illustrative examples of $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R environment for **Rand**, **HTour2X**, and **HTour5X**, respectively, using the *Global_RT* configuration. It is clear how the median $\mathcal{N}(t)$ value decreases and variance increases as the tournament size increases. Entity reassignment volumes were largely unaffected by tournament size.

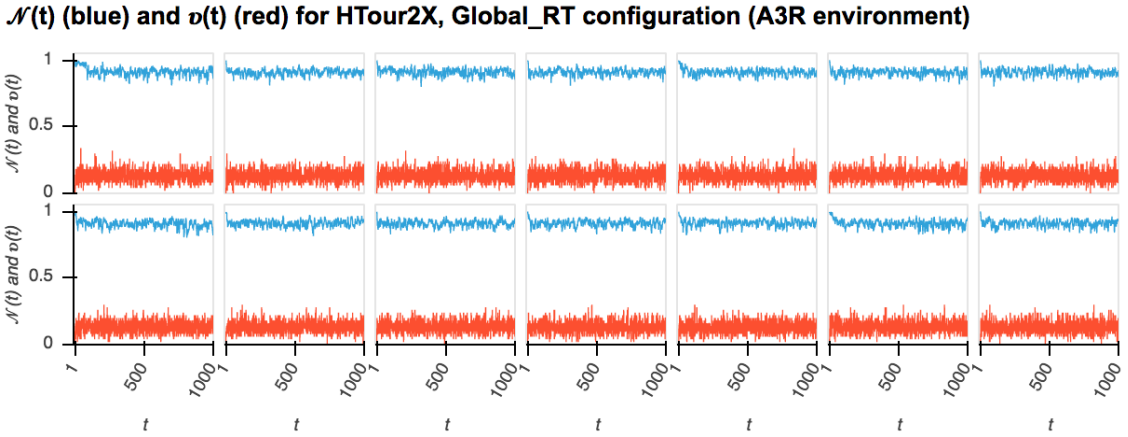
The behavior of entity tournament selection (ETour2 and ETour25)

The previous subsection outlined how **HTour2X**, **HTour5X**, **HTour2M**, **HTour5M**, **Freq2**, and **Freq5** were similar to **Rand**, and how larger tournament sizes influenced HSD behavior in configurations that used the *global* topology. In contrast, entity tournament selection (as discussed in section 3.4.6), exhibited completely different behavior. **ETour2** and **ETour25** conducted tournaments between individual entities as opposed to conducting tournaments between heuristics. Figures 6.19 and 6.20 show the HSD and entity reassignment rates for **ETour2** and **ETour25**, respectively.

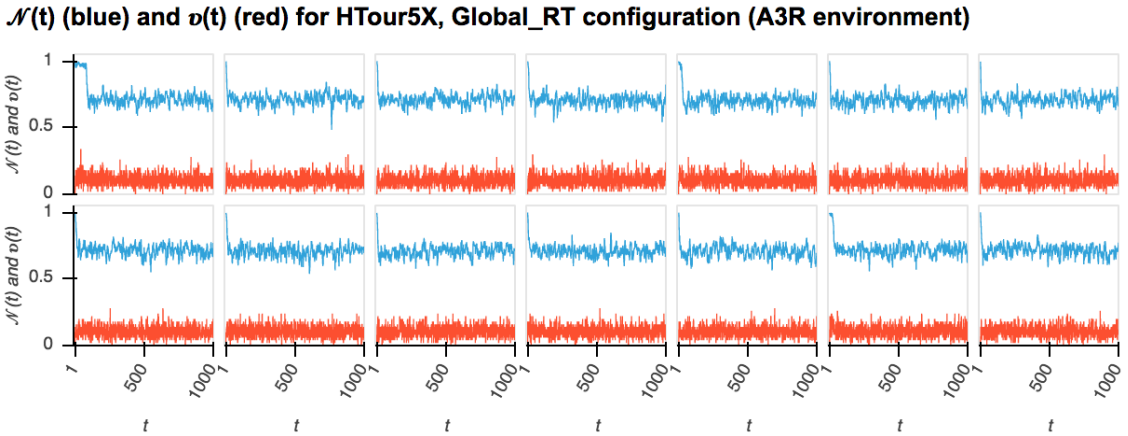
A holistic glance at each figure shows that there are noteworthy differences between the HSD and entity reassignment behaviors of **ETour2** and **ETour25**. Both hyper-



(a) **Rand**



(b) **HTour2X**



(c) **HTour5X**

Figure 6.18: $\mathcal{N}(t)$ and normalized $v(t)$ for **Rand**, **HTour2X**, and **HTour5X** using the *Global_RT* configuration.

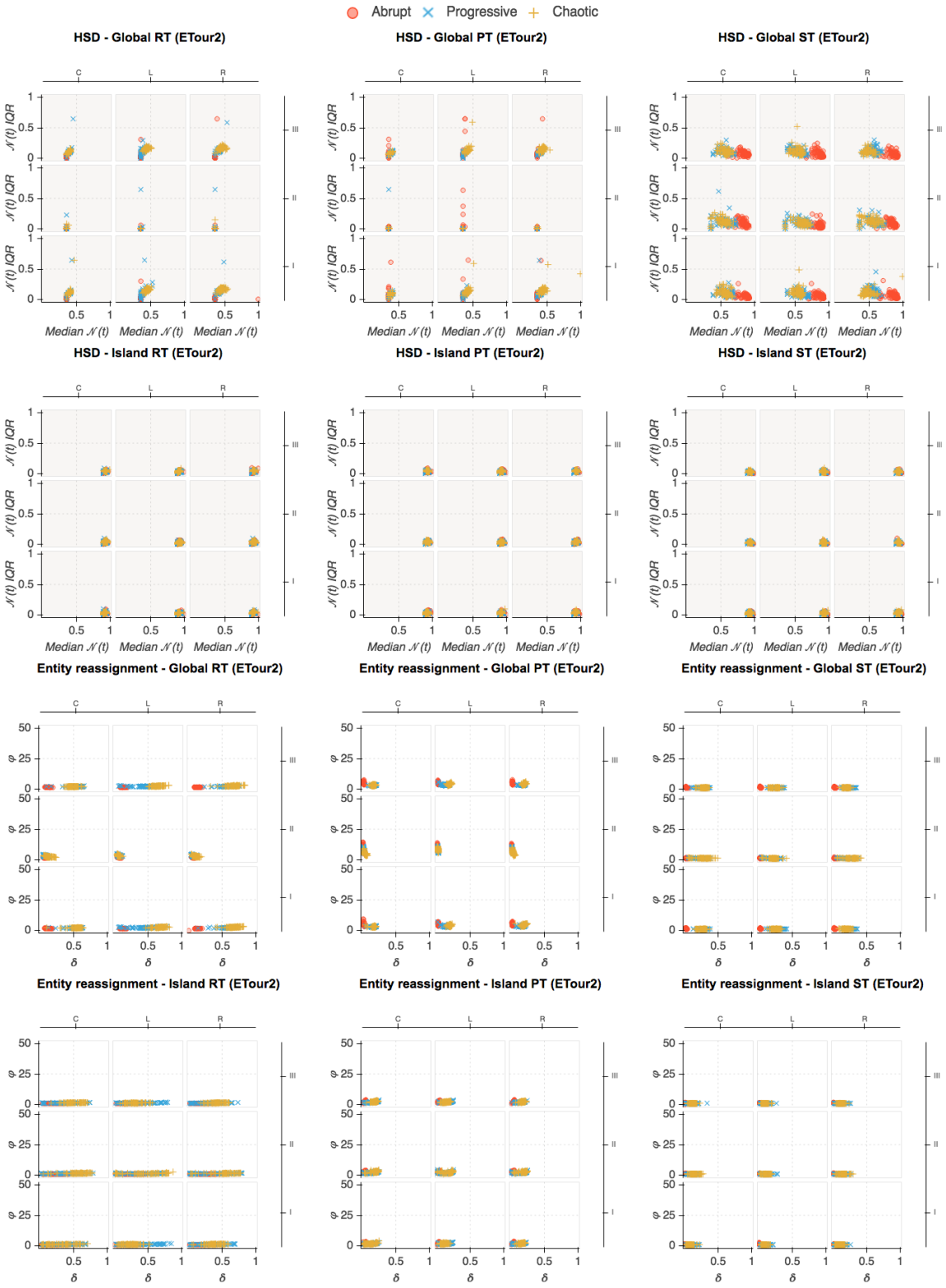


Figure 6.19: HSD and entity reassignment rate analysis for **ETour2**

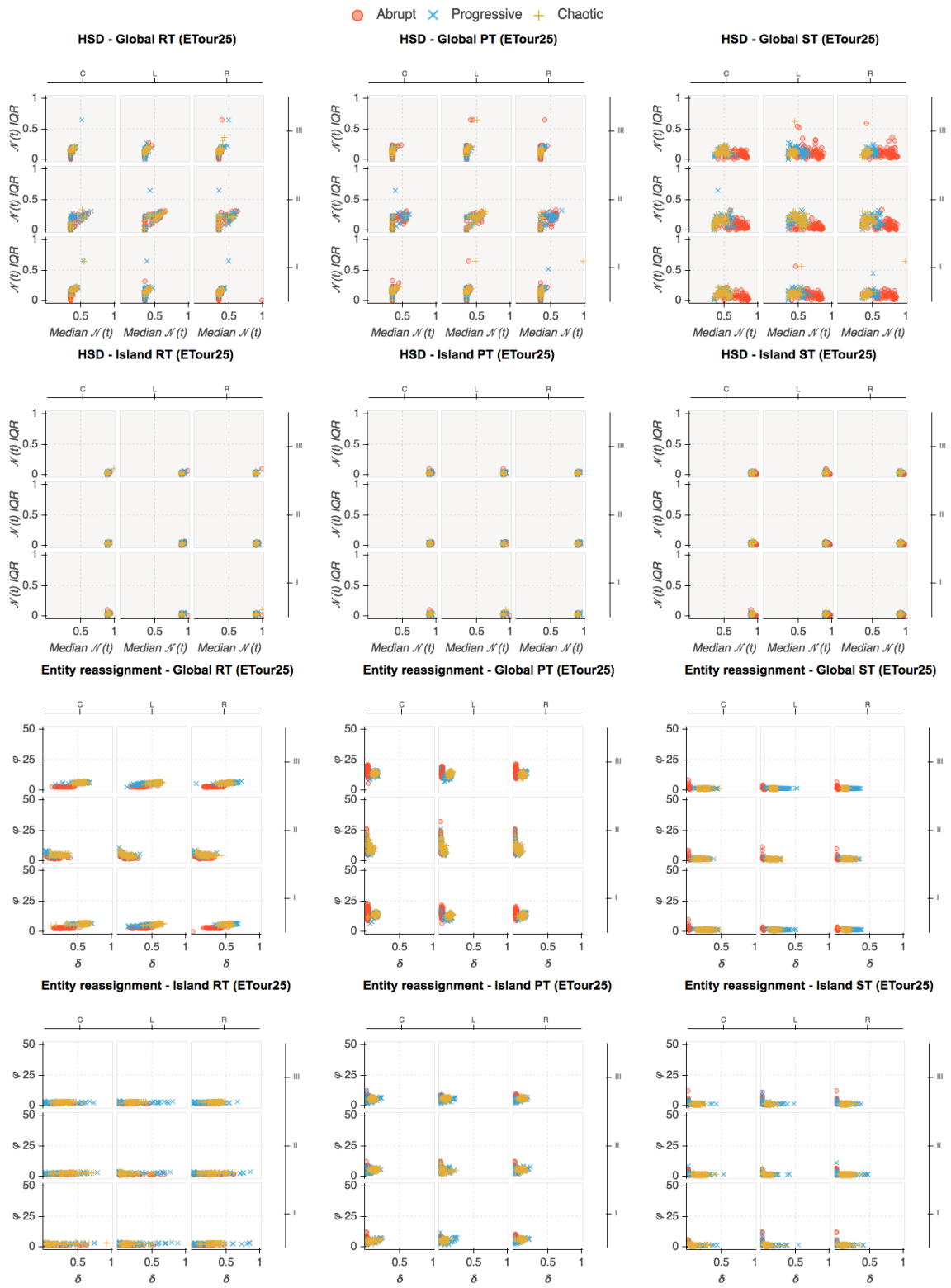


Figure 6.20: HSD and entity reassignment rate analysis for **ETour25**

heuristics depart substantially from the behavior of **Rand**, in stark contrast to the hyper-heuristics that are based on heuristic tournament selection. The HSD and entity reassignment charts reveal that each trigger and topology configuration had clearly defined ranges of sample variation. This is especially visible for **ETour25** in figure 6.20, for example the triangular areas that are visible for the *Global_RT* configuration.

Figures 6.6 and 6.7 show the erratic performance of entity tournament selection, depending on the tournament size that was used. **ETour2** was only a top performer when using the *Global_RT* configuration, but yielded mediocre results when using either the *Global_PT* or *Global_ST* configurations. **ETour2** was among the worst performing hyper-heuristics whenever the *island* topology was used. On the other hand, **ETour25** was consistently among the top performing hyper-heuristics in the *Global_RT* and *Global_PT* configurations, and performed substantially better than **ETour2** in all *island* topology configurations (but was not among the top hyper-heuristics).

The following observations below about the behavior of different trigger and topology configurations for **ETour2** and **ETour25** are visible in figures 6.19 and 6.20, respectively.

ETour2 and ETour25 using the *Global_RT* and *Global_PT* configurations: The HSD subplots for the *Global_RT* and *Global_PT* configurations are readily comparable (for **ETour2** and **ETour25** individually). However, the samples in the HSD plots for **ETour2** were more concentrated than in the corresponding plots for **ETour25**, depending on the type of environment. This shows that the tournament size parameter had a visible impact on the variance of $\mathcal{N}(t)$ values for entity tournament selection. Both **ETour2** and **ETour25** show comparable HSD plots for all Type I and Type III environments. However, **ETour2** and **ETour25** show completely different behavior for Type II environments. **ETour25** had a triangular variance pattern for Type II environments, where the width of the possible range of IQR spread values for **ETour25** decreased linearly as the median $\mathcal{N}(t)$ value increased. The IQR value was highest when the median $\mathcal{N}(t)$ value was approximately 0.65. On the other hand, **ETour2** had a very narrow HSD distribution compared to **ETour25**.

The entity reassignment plots for the *Global_RT* and *Global_PT* configurations are distinctly different from one another. However, the corresponding plots for the same configuration bear a degree of resemblance in shape between **ETour2** and **ETour25**.

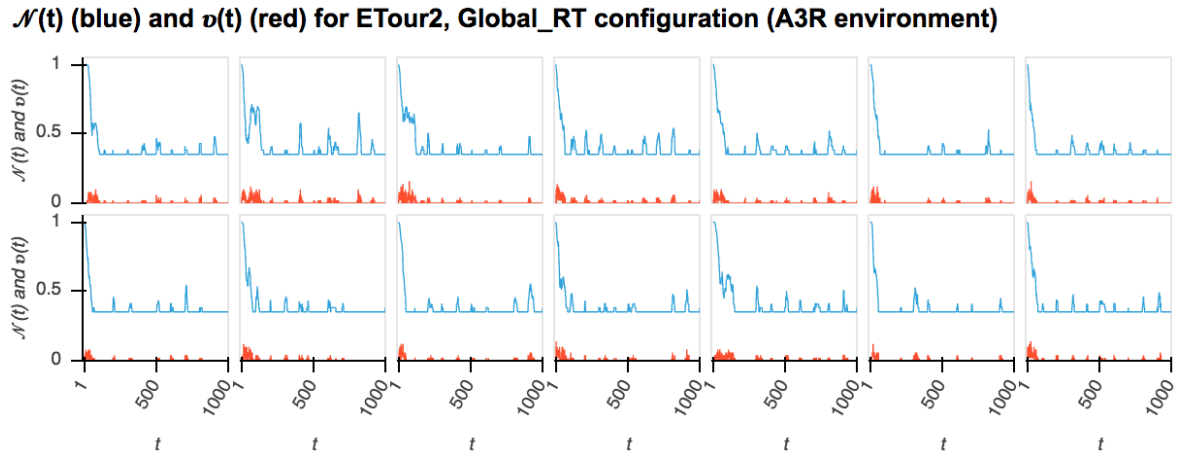
The *Global_RT* configurations showed high variance in δ values, but low φ values. This observation is visible as horizontal lines in the respective entity reassignment plots. In contrast, the *Global_PT* configurations show low δ values and high variance in φ values, which is visible as vertical lines in the respective entity reassignment plots. Values for φ and variances were noticeably higher for **ETour25** than **ETour2** (especially for *Type II* environments), but the general shape of each subplot was comparably similar.

Figure 6.21 shows illustrative examples of $\mathcal{N}(t)$ and normalized $v(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for **ETour2**. Figure 6.22 shows the corresponding plots for **ETour25**. The **ETour25** plots clearly show how *progressive* environment samples have more frequent entity reassignments with slightly higher volumes compared to *abrupt* environment samples. **ETour25** clearly had the ability to adapt to the environment and the exact problem instance under consideration. In contrast, **ETour2** showed more uniform HSD behavior over time and much smaller entity reassignment volumes, as expected.

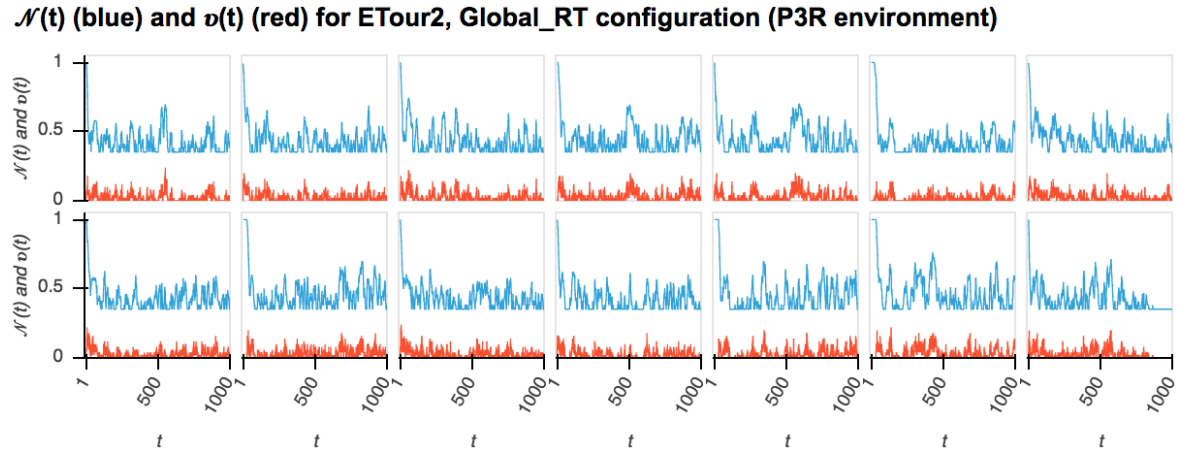
ETour2 and ETour25 using the *Island_RT* and *Island_PT* configurations: The HSD plots for both trigger configurations were almost identical for **ETour2** and **ETour25**, respectively. The **ETour2** plots show a wide range of $\mathcal{N}(t)$ values across the entire range of possible values. The **ETour25** plots indicate that most $\mathcal{N}(t)$ values were concentrated around approximately 0.9 (the minimum value for the *island* configuration).

The entity reassignment behaviors for the *Island_RT* configurations were almost identical for **ETour2** and **ETour25**, showing high variance for δ values and very low φ values. In contrast, the *Island_PT* configurations showed the same δ pattern between **ETour2** and **ETour25**, but **ETour25** (similar to the *global* topology case) had much higher sample variance for φ than **ETour2** had.

ETour2 and ETour25 using the *Global_ST* configuration: The HSD plots for the *Global_ST* configuration for both **ETour2** and **ETour25** depart from the trends visible in the RT and PT trigger configurations. Both HSD plots are visually similar to the corresponding plot of **RLFreq**, as visible in figure 6.11. Each subplot shows how HSD behavior was mainly differentiated by the presence of *abrupt*, *progressive*, or *chaotic* environment dynamics, and was relatively consistent across other types of environment



(a) A3R

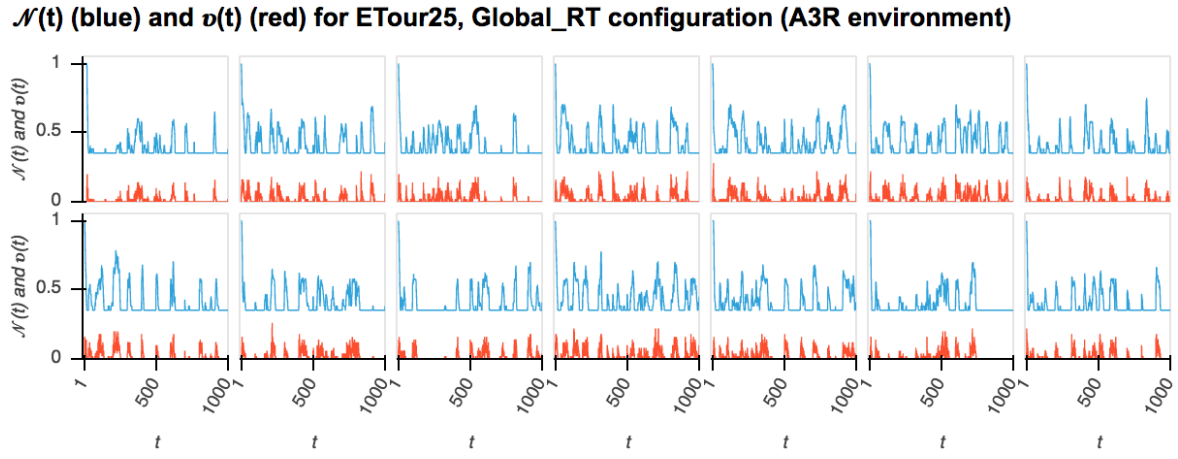


(b) P3R

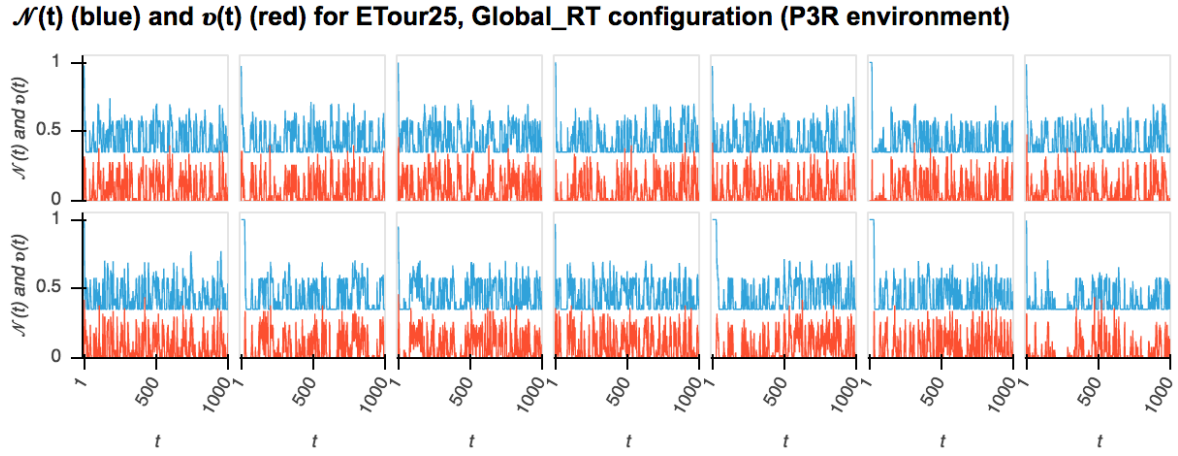
Figure 6.21: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour2** using the *global* topology and RT trigger.

dynamics. Both **ETour2** and **ETour25** clearly showed a higher propensity to yield higher median $\mathcal{N}(t)$ values for *abrupt* environments (which saw less frequent heuristic changes) than for *progressive* or *chaotic* environments. The entity reassignment plots for both **ETour2** and **ETour25** confirm that both hyper-heuristics rarely reassigned entities for *abrupt* environments (i.e. low δ values) compared to when the hyper-heuristics operated on *progressive* or *chaotic* environments.

Figures 6.23 and 6.24 show illustrative examples of $\mathcal{N}(t)$ and normalized $v(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for



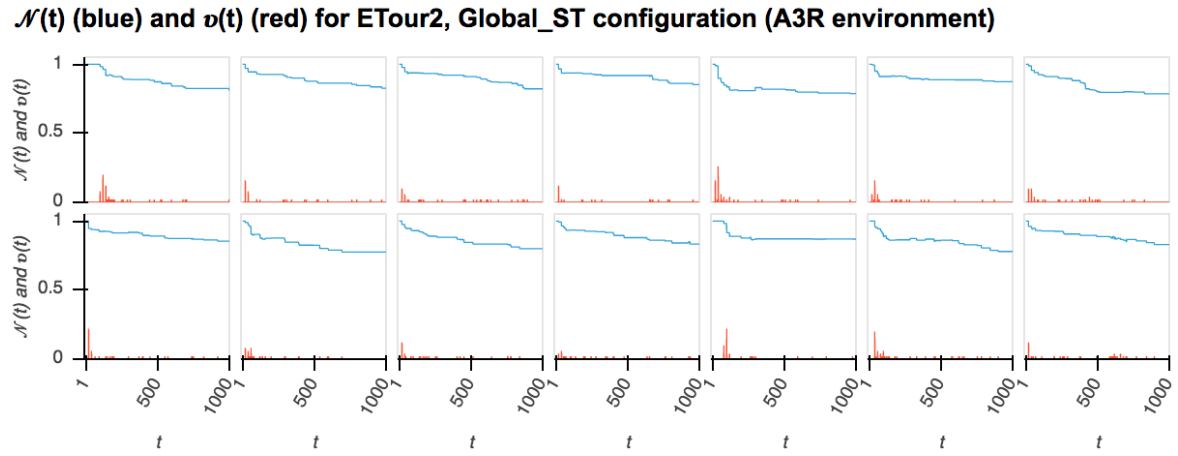
(a) A3R



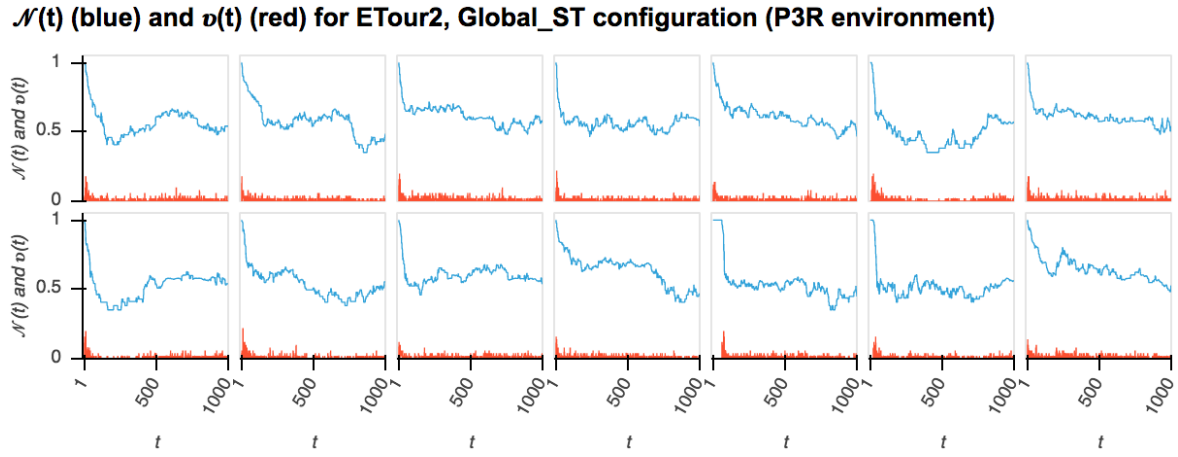
(b) P3R

Figure 6.22: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour25** using the *global* topology and RT trigger.

ETour2 and **ETour25** (both using the *Global_ST* configuration). At first glance, both **ETour2** and **ETour25** show seemingly similar behavior in each respective environment. However, closer inspection reveals that the $\mathcal{N}(t)$ values for **ETour25** were generally more responsive. This is also expressed in the HSD plots in figures 6.19 and 6.20, where the "rectangular"-like block of median $\mathcal{N}(t)$ values and IQR range values for **ETour25** are situated more towards the left of the subplot than for values of **ETour2**. The difference is subtle in figures 6.19 and 6.20, but the plots in figures 6.23 and 6.24 clearly show how individual algorithm runs for **ETour25** have decidedly varied behavior compared



(a) A3R

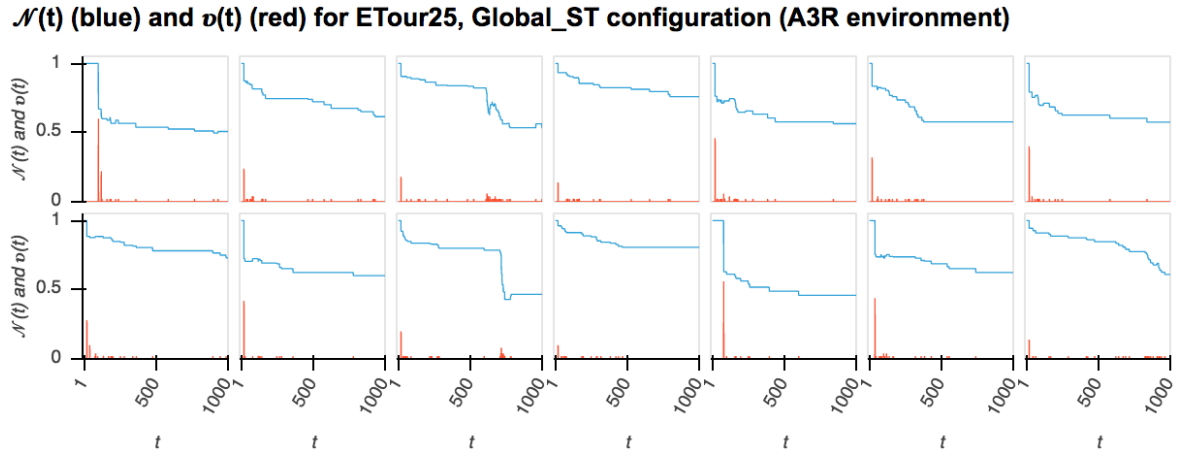


(b) P3R

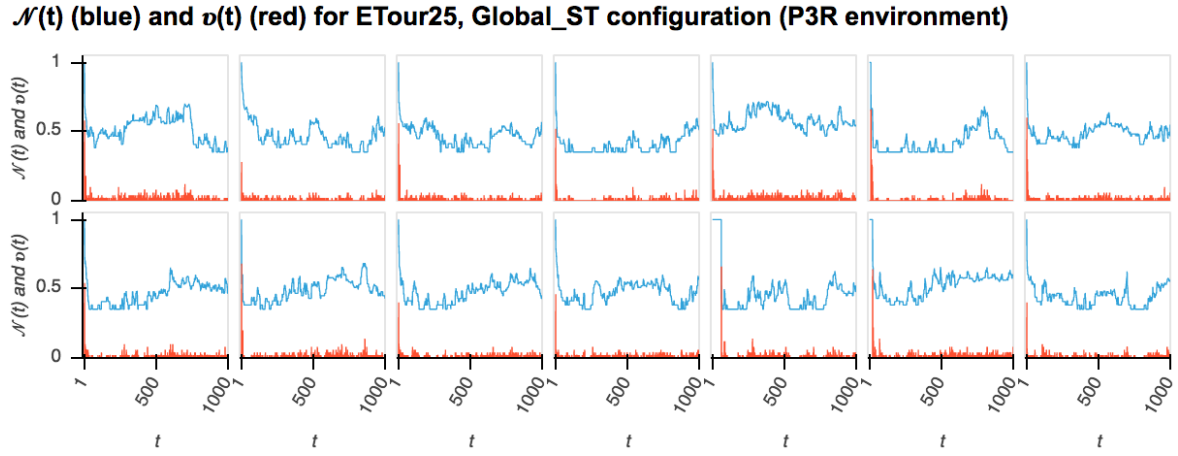
Figure 6.23: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour2** using the *global* topology and ST trigger.

to that of **ETour2**. Figure 6.7 shows that, for the *Global_ST* configuration, **ETour25** performed better than random selection and was comparable to fixed selection, while **ETour2** fared substantially worse against both fixed and random selection. Clearly, the different HSD behavior is reflected in the performance results.

ETour2 and ETour25 using the *Island_ST* configuration: The HSD plots for both **ETour2** and **ETour25** reveal similar behavior patterns to that of the **Global_ST** configuration, in that median $\mathcal{N}(t)$ values and IQR spread values for $\mathcal{N}(t)$ fluctuated across the full spectrum of possible values. The entity reassignment plots reveal that



(a) A3R

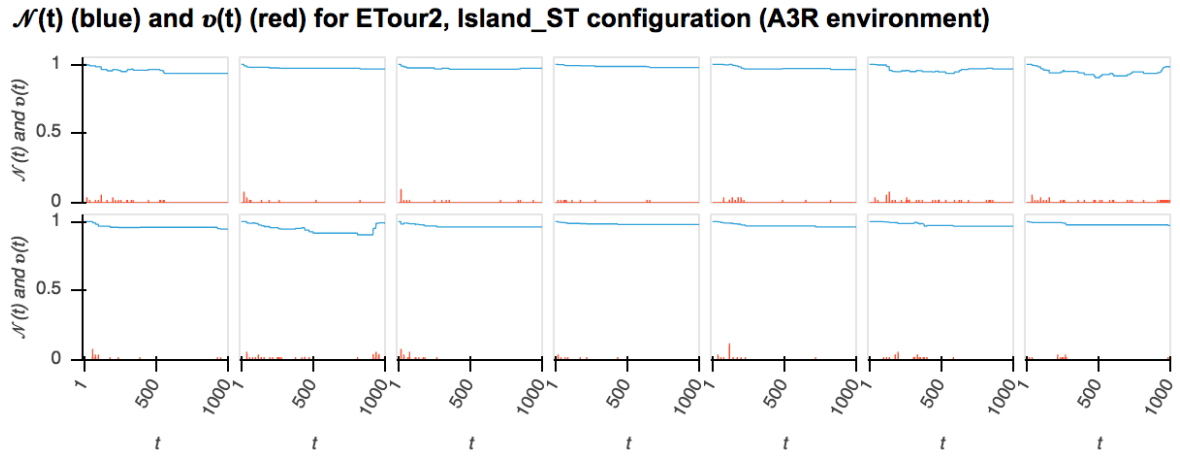


(b) P3R

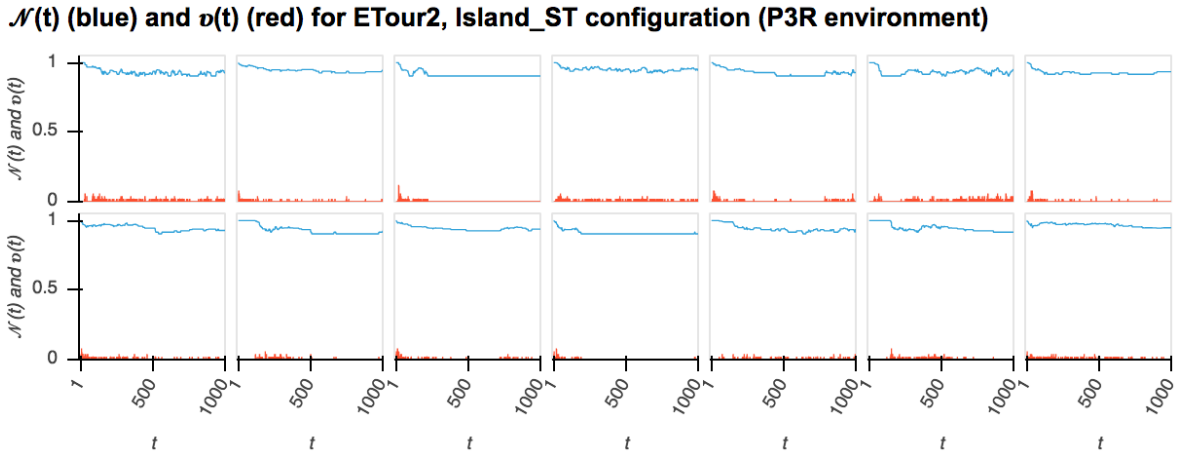
Figure 6.24: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour25** using the *global* topology and ST trigger.

δ and φ values also bear some resemblance to those for the *Global_ST* configuration. The φ sample variance is, however, higher for **ETour25**. However, both **ETour2** and **ETour25** failed to significantly outperform any other hyper-heuristics when using the *Island_ST* configuration and **ETour2**, in fact, was among the worst performing methods.

Figures 6.25 and 6.26 show illustrative examples of $\mathcal{N}(t)$ and normalized $v(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for **ETour2** and **ETour25** (both using the *Island_ST* configuration). Visually there is almost no difference between samples, environments, or between **ETour2** and **ETour25**.



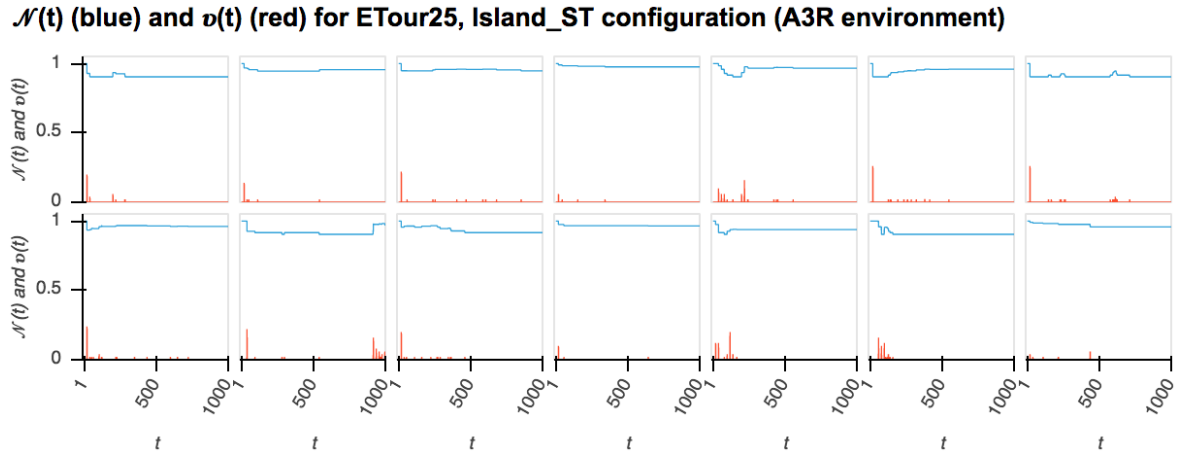
(a) A3R



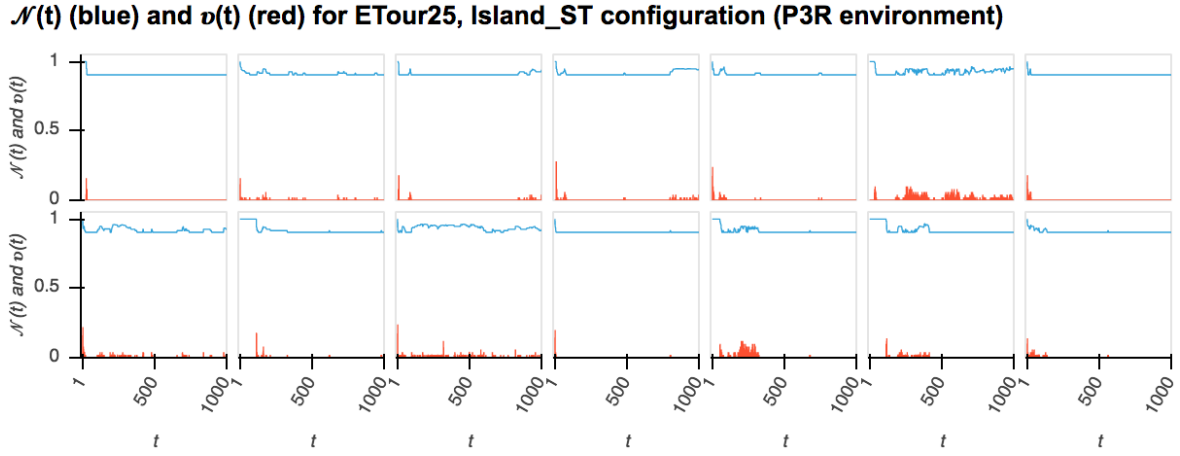
(b) P3R

Figure 6.25: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour2** using the *island* topology and ST trigger.

The use of the ST trigger and the *island* topology did not yield varied behavior across environments, problem instances, or tournament sizes when entity tournament selection was employed. Figure 5.18 confirms that neither **ETour2** nor **ETour25** showed any significant differences in performance. Further analysis is required in future studies to determine why **ETour2** and **ETour25** do not perform well under the *island* topology with an ST trigger configuration.



(a) A3R



(b) P3R

Figure 6.26: $\mathcal{N}(t)$ and normalized $v(t)$ for **ETour25** using the *island* topology and ST trigger.

Conclusion about Entity tournament selection: Entity tournament selection behaved noticeably different than the other tournament selection-based hyper-heuristics that conducted tournaments between heuristics. Tournament size directly affected the volume and frequency of entity reassignments, but the choice of topology and trigger affected the efficacy of the hyper-heuristic.

The behavior of competitive population heuristic selection

The analysis in figures 6.6 and 6.7 shows that **Comp** never outperformed **Rand** or **GFix** when using either the *Global_RT* or *Global_PT* configurations, and barely managed to outperform **GFix** in a few environments when using the *Global_ST* configuration. When configured with the *island* topology, **Comp** showed strong performance against **IFix** regardless of trigger type, but never managed to outperform **Rand**. **Comp** showed extremely weak performance against most of the other hyper-heuristics by never recording any wins and logging a number of losses against the majority of other methods (especially in the *Global_RT* or *Global_PT* configurations). The question arises whether the HSD and entity reassignment behaviors of **Comp** depart greatly from the other hyper-heuristics as well.

Figure 6.27 shows the HSD and entity reassignment plots for **Comp**. It is immediately apparent that the HSD plots for all three *global* topology variations strongly resemble each other (**Comp** is the only hyper-heuristic with this characteristic, as can be seen in appendix A). Within each HSD plot, the subplots for each environment appear similar. The shape of the HSD plots are unique when compared to the HSD plots of the other hyper-heuristics as found in appendix A.

The following observations below about the behavior of different trigger and topology configurations for **Comp** are visible in figure 6.27.

Comp using the *Global_RT* configuration: The entity reassignment behavior was nearly identical to that of **Rand**. High δ values reveal that entities were triggered for reassignment nearly every algorithm iteration, while low and consistent φ values show that each type of environment experienced similar volumes of entity reassignment. Similar to **Rand**, the variance between samples for δ was extremely low for each environment, which indicates that the average number of reassigned entities never varied, regardless of the exact problem instance under consideration. Sample variance for φ values was slightly higher than for **Rand**.

However, contrary to **Rand**, the sample variance in the HSD plot shows that the *Global_RT* configuration exhibited algorithm runs with a wider gamut of different balances of entities across heuristics. A unique feature that is visible in the HSD plot for

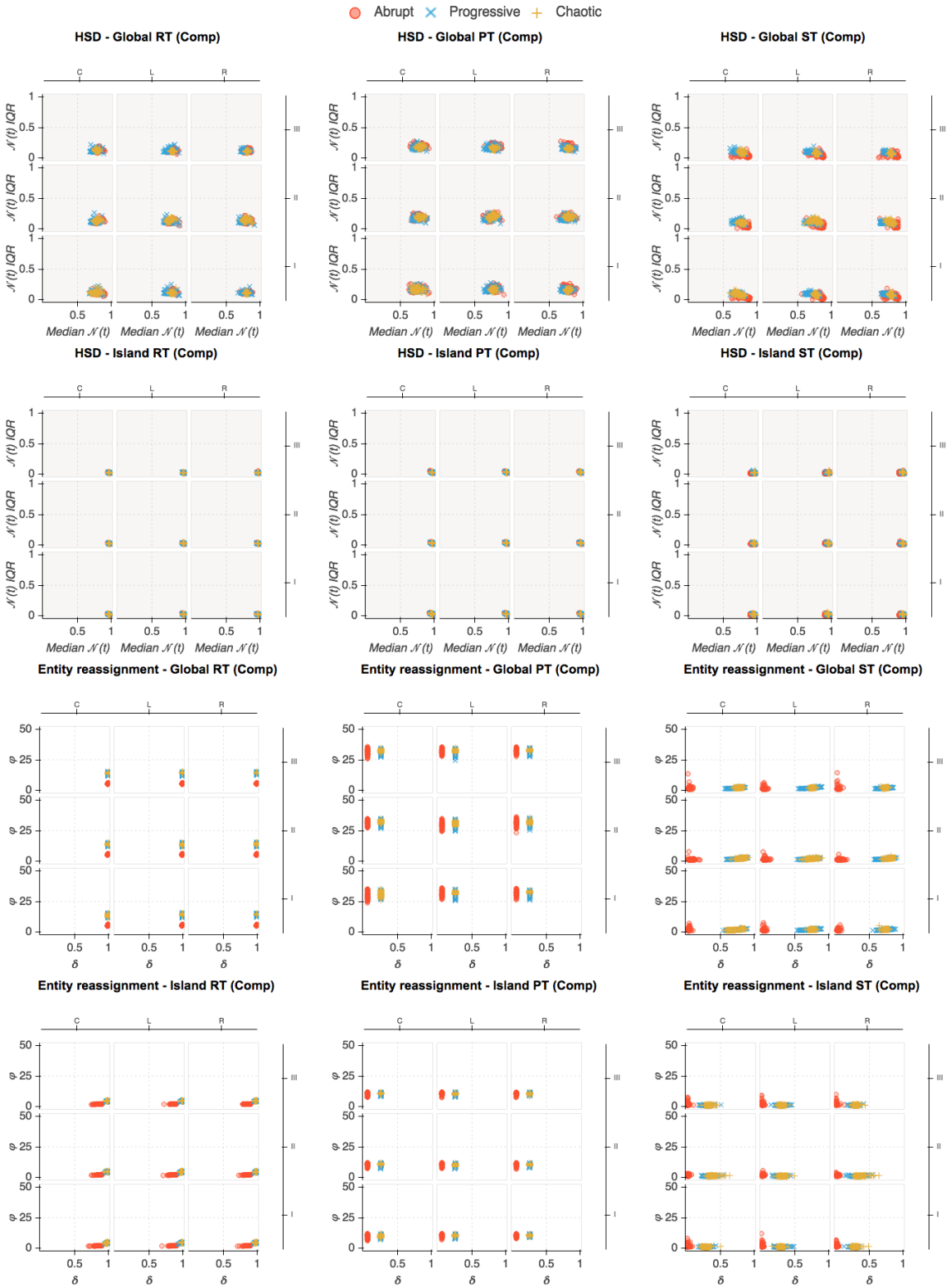


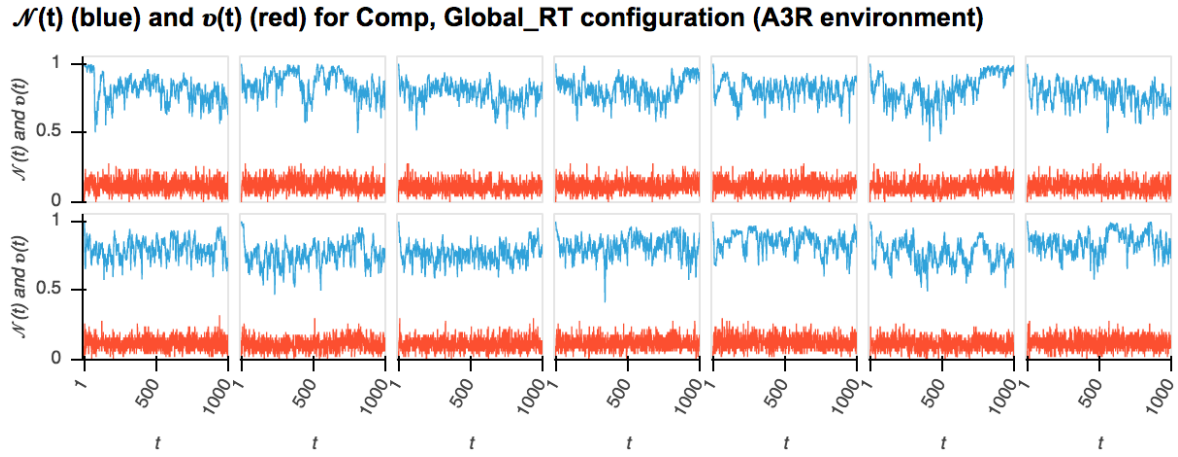
Figure 6.27: HSD and entity reassignment rate analysis for **Comp**

Comp is that none of the IQR spread values for $\mathcal{N}(t)$ ever dropped below approximately 0.15 (for any environment). This reveals that wide distributions were the norm for $\mathcal{N}(t)$ values within each individual algorithm run, and that no problem instance ever experienced constant HSD behavior throughout the run. This is not the case for the majority of the other hyper-heuristics, which all showed a number of samples with very low IQR spread values for $\mathcal{N}(t)$.

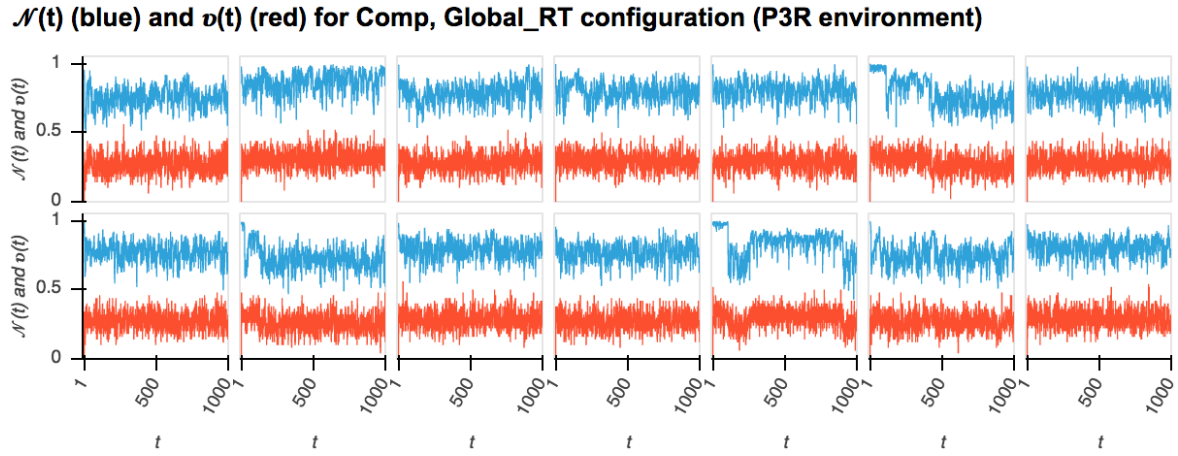
The entity reassignment plot reveals that the *Global_RT* configuration reassigned at least some entities to new heuristics every iteration. Since φ values were nearly identical to that of **Rand** (i.e. at the maximum rate of the RT trigger), the conclusion is that all triggered entities almost always got reassigned to new heuristics. In other words, the *Global_RT* configuration rarely maintained the original assignments of any triggered entities. Unlike **Rand**, entity reassignment was done in a non-uniform way across entities (as indicated by the lower $\mathcal{N}(t)$ values over time along with a wider variance). Ultimately, however, these reassignments were not fruitful, since the hyper-heuristic failed to outperform **GFix** or **Rand** in the end.

Figure 6.28 shows illustrative examples of $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for **Comp** using the *Global_RT* configuration. It is immediately apparent that the HSD behavior is different to the other hyper-heuristics discussed above: within any given sample the HSD values seem erratic, alternating unpredictably between highs and lows. The $\mathcal{N}(t)$ values rarely (if ever) hit values of 1.0 and constantly fluctuated across a range of 0.25, which illustrates why the IQR in figure 6.27 never approached 0. Outliers are present, visible as spikes in HSD values, but these outliers typically tend to return to the existing range of variance HSD values. Similarly, $\upsilon(t)$ values show that entity assignments occurred every iteration and that volumes fluctuated predictably in a well-defined range, providing visual confirmation for the δ and φ values in figure 6.27.

Comp using the *Island_RT* configuration: The HSD plot is identical to that of **Rand**, in stark contrast to the *Global_RT* configuration. Median $\mathcal{N}(t)$ values are approximately 1 and low IQR values indicate that most entities were balanced across all heuristics for the entire algorithm run. The low variance shows that this was the case throughout each sample in every environment.



(a) A3R

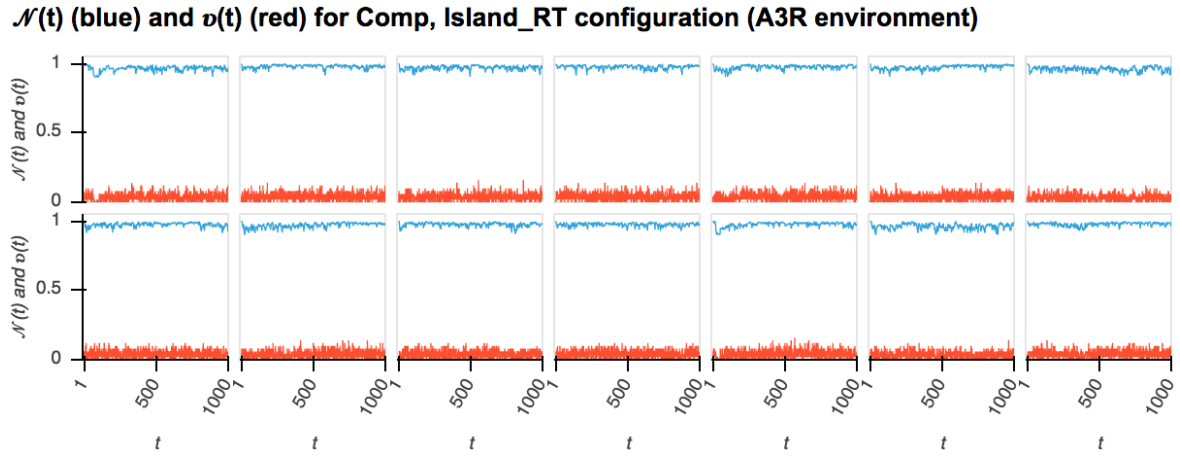


(b) P3R

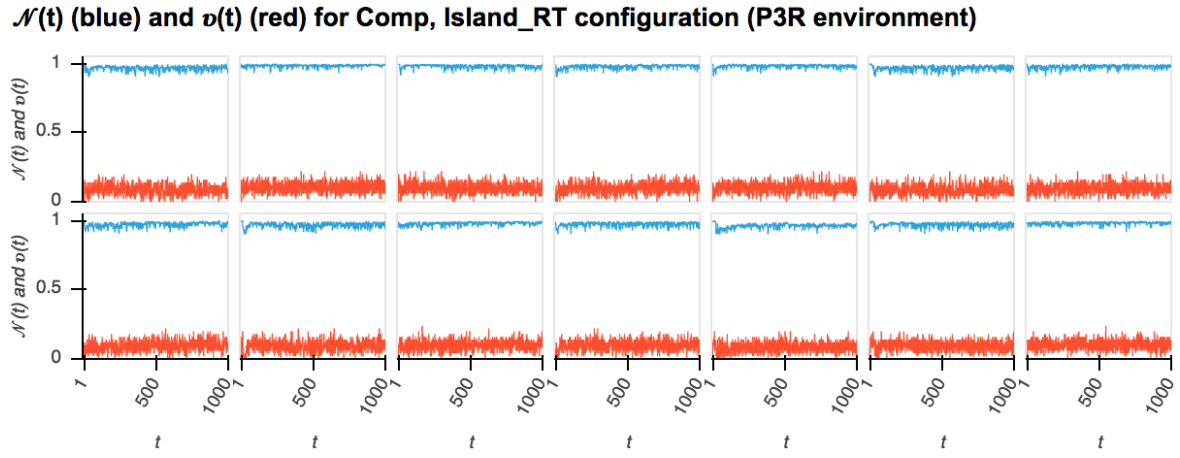
Figure 6.28: $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **Comp** using the *global* topology and RT trigger.

The entity reassignment plot suggests selective entity reassignment frequency behavior, in that different environments showed different spreads of δ values between samples (this is noticeable as a horizontal red line for *abrupt* environments). Triggered entities displayed a wide range of behavior in different algorithm runs, ranging from always being reassigned (i.e. $\delta \approx 1$) to being reassigned only two out of three times the entities were triggered (i.e. $\delta \approx 0.66$). The φ values were similar to those for the corresponding configuration for **Rand**.

Figure 6.29 shows illustrative examples of $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all



(a) A3R



(b) P3R

Figure 6.29: $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **Comp** using the *island* topology and RT trigger.

1000 iterations for 14 random samples of the A3R and P3R environments for **Comp** using the *Island_RT* configuration. Considering the smaller HSD ranges for the *island* topology compared to the *global* topology, the $\mathcal{N}(t)$ behavior in figure 6.29 is readily comparable to the values in figure 6.28 in the sense that $\mathcal{N}(t)$ fluctuated almost unpredictably across the range of possible values, and large outliers are present that always tended back towards the general pattern of variance. The $\upsilon(t)$ values in sub-figure 6.29a frequently hit values of 0, in concert with the low φ values and fluctuating δ values present in figure 6.27.

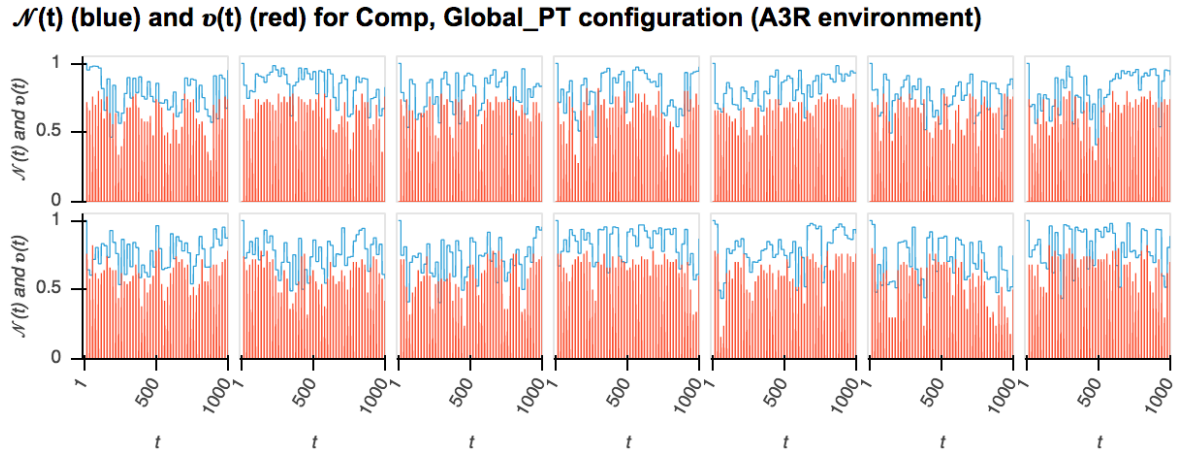
Comp using the *Global_PT* configuration: The HSD plot for the *Global_PT* configuration is similar to the plot for the *Global_RT* configuration. Similar conclusions can be made about HSD behavior. The entity reassignment plot resembles that of **Rand**, but the variance of φ is visibly higher. This behavior reveals that the *Global_PT* configuration moderated the number of entities that were reassigned based on the specific sample under consideration.

Figure 6.28 shows illustrative examples of $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for **Comp** using the *Global_PT* configuration. Similar to the *Global_RT* configuration, HSD behavior seems erratic and inconsistent (which explains the high IQR values for $\mathcal{N}(t)$ that never approach zero). Entities were reassigned every iteration that experienced a PT trigger event, but the varied normalized $\upsilon(t)$ values confirm the inconsistency of the number of reassigned entities, as highlighted by the high variance in φ values.

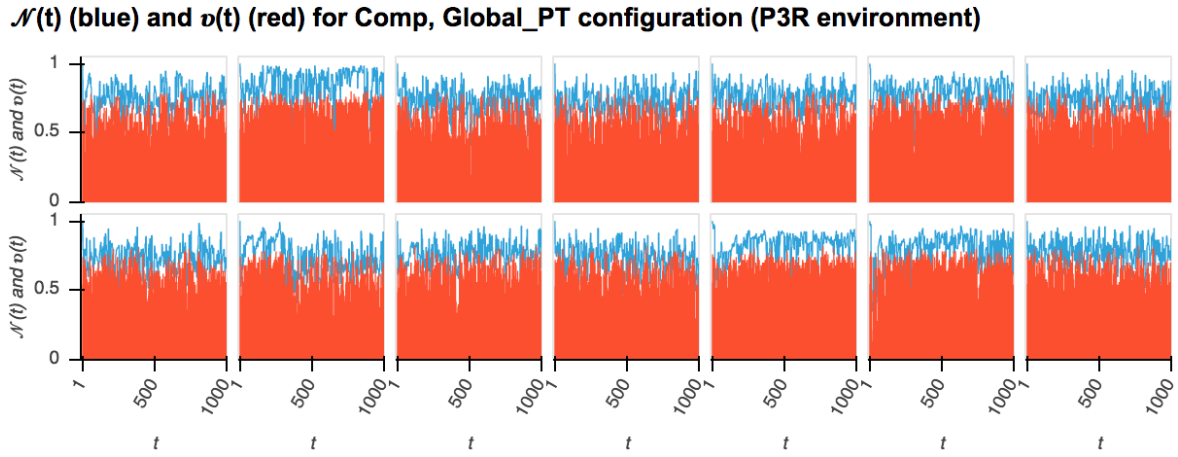
Comp using the *Island_PT* configuration: The HSD and entity reassignment plots are similar in shape to the corresponding plots for the *Global_PT* configuration, but at the smaller value ranges imposed by using the *island* topology. Similar conclusions can be made. Figure 6.29 shows illustrative examples of $\mathcal{N}(t)$ and normalized $\upsilon(t)$ values over all 1000 iterations for 14 random samples of the A3R and P3R environments for **Comp** using the *Island_PT* configuration. The observations and conclusions are similar as for the *Global_PT* configuration results in figure 6.30, albeit at reduced value scales.

Comp using the *Global_ST* configuration: The HSD plot is similar to the plots for the *Global_RT* and *Global_PT* configurations, indicating that the *Global_ST* configuration exhibited algorithm runs with a wider gamut of different balances of entities across heuristics. However, there is one major difference compared to the HSD plots of the *Global_RT* and *Global_PT* configurations: the minimum IQR spread values were noticeable lower (i.e. closer to zero) for many samples. This shows that the *Global_ST* configuration yielded tightly distributed HSD values across entire algorithm runs for a subset of samples, where the the *Global_RT* and *Global_PT* configurations failed to do so.

The entity reassignment plot bears a strong resemblance to that of **Rand**, and similar



(a) A3R

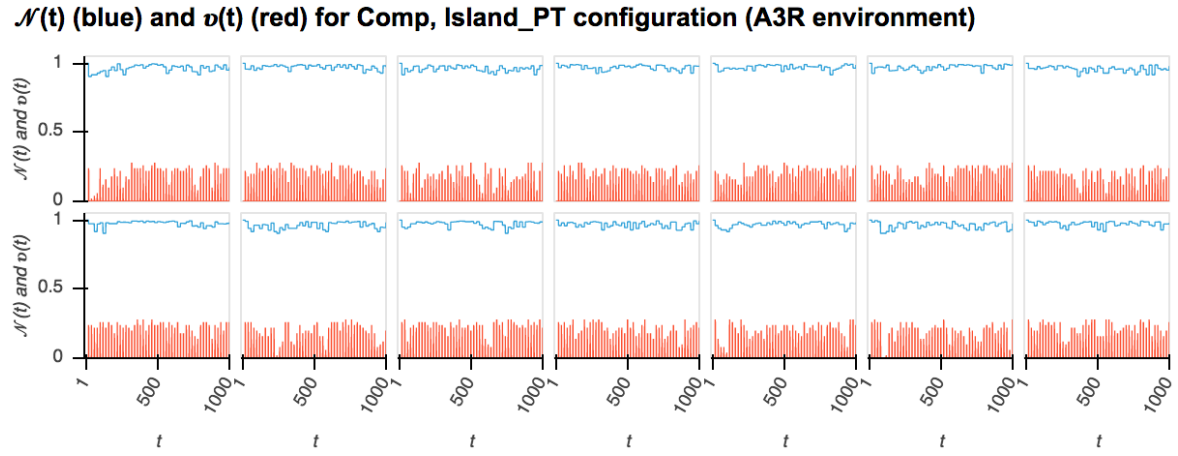


(b) P3R

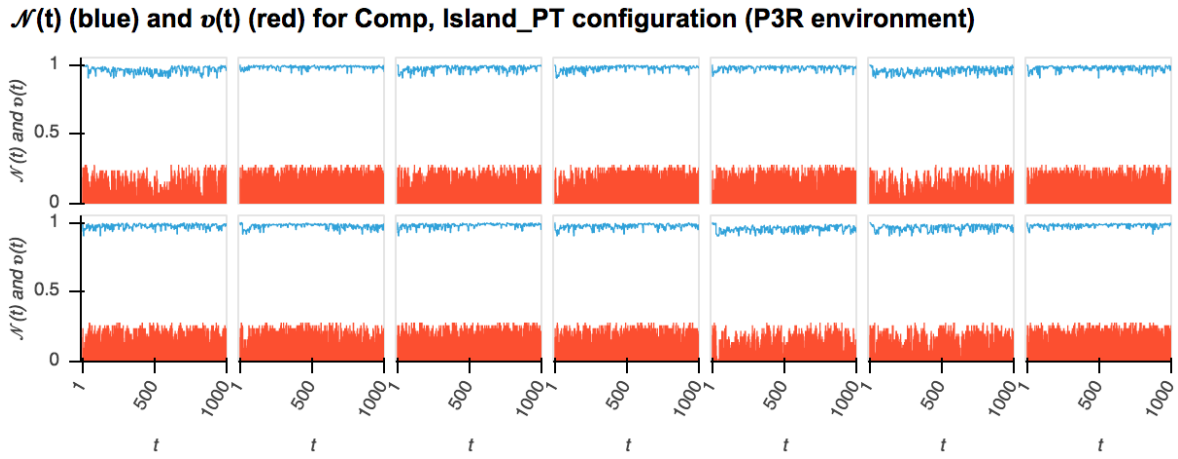
Figure 6.30: $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **Comp** using the *global* topology and PT trigger.

conclusions can be made. The variance patterns for δ were, however, wider and values were slightly lower (especially for *progressive* and *chaotic* environments).

Comp using the *Island_ST* configuration: The HSD and entity reassignment plots are similar to those for **Rand**, and similar conclusions can be made. One deviation in the entity reassignment plot is the slightly wider variance patterns, as well as the"L"-shaped variance for *abrupt* environments.



(a) A3R



(b) P3R

Figure 6.31: $\mathcal{N}(t)$ and normalized $\upsilon(t)$ for **Comp** using the *island* topology and PT trigger.

Conclusions about Comp: **Comp** showed unique behavior compared to the other hyper-heuristics, but, however, was unable to outperform random or even fixed heuristic selection for a number of configurations. Inspection of the behavior of **Comp** showed complex patterns of transitioning entities between heuristics, with the HSD values alternating widely over time within individual algorithm runs. Deeper analysis in a future study is needed to determine how the intelligent selection logic behind the **Comp** selection operator affected entity assignments, and why that intelligence did not bear any fruit.

6.4 Summary

This chapter analyzed the performance and behavior of a number of well-known hyper-heuristics that were configured to operate as part of the HMHH framework. Key questions were whether any of the hyper-heuristics could raise performance above that of simply using the individual heuristics or speciated heuristics, to establish whether any hyper-heuristics performed better than either fixed or random selection, and to understand which hyper-heuristics performed better than others.

Each of the hyper-heuristics performed either on par with or statistically significantly better than all of the standalone heuristics. None of the hyper-heuristics ever recorded a loss against any of the standalone heuristics for any environment. The choice of topology and trigger impacted the ability of each hyper-heuristic to yield a higher number of wins the standalone heuristics for more environments (as opposed to draws for those environments). Hyper-heuristics that used a *global* topology generally yielded a higher number of wins across more environments than those same hyper-heuristics that used the *island* topology. The *Global_RT* configuration showed the strongest ability to yield the highest number of wins across the most environments for nearly half of the hyper-heuristics.

All hyper-heuristics either matched or outperformed the homogeneous speciated heuristics in a statistically significant manner for the majority of environments. Isolated losses were recorded by certain hyper-heuristics for a handful of environments, in sharp contrast to the stand-alone heuristic outcome where no losses were ever recorded. No hyper-heuristic managed to record nine wins against all nine speciated heuristics, which resulted in more draws in a higher number of environments (in addition to the handful of losses). There was no clear link between topology or trigger combinations that systemically yielded increased wins by hyper-heuristics against the speciated heuristics.

The hyper-heuristics had varying degrees of success against fixed heuristic selection, depending on the exact trigger and topology configuration that was used. Hyper-heuristics that used the *island* topology had a clear tendency to outperform fixed selection across the majority of environment types, regardless of whether the RT, PT, or ST triggers were used, while yielding hardly any losses (the small remainder were draws). In contrast, hyper-heuristics that used the *global* topology were influenced by the trigger

choice. Every combination of trigger with the *global* topology resulted in the region of 2-5 losses for practically every hyper-heuristic (with isolated hyper-heuristics yielding up to 10 losses). The *Global_RT* configuration showed strong wins for those hyper-heuristics that did not rely on roulette wheel selection (those hyper-heuristics predominantly only showed losses and draws). This trend is even more pronounced for the *Global_PT* configuration, where only hyper-heuristics based on heuristic tournament-based selection showed high numbers of wins and all other hyper-heuristics only recorded draws and losses. Lastly, the *Global_ST* trigger saw mostly draws for most hyper-heuristics with a small number of wins and losses. Those hyper-heuristics based on tournament selection, when configured with larger tournament sizes, yielded a more noteworthy number of wins.

No hyper-heuristic, regardless of topology or trigger, ever recorded noteworthy losses against random heuristic selection (**ETour2** being the only exception). When the *island* topology was used, only **RLFreq** showed strong wins against random selection, and only when using the *Island_RT* configuration. All other hyper-heuristics that used the *island* topology yielded mostly draws against **Rand**. The hyper-heuristics that relied on roulette wheel selection never managed to record any wins against **Rand** when the *global* topology was used. In contrast, the remaining methods showed relatively strong wins when using the *Global_RT* configuration (with the remainder of environments being draws). For the *Global_PT* configuration, only those hyper-heuristics that relied on tournament selection yielded a higher number of wins, and more so for higher tournament sizes. The remainder of cases were, again, draws. All hyper-heuristics yielded mostly draws against **Rand** when using the *Global_ST* configuration.

To be deemed a good hyper-heuristic overall, the hyper-heuristic needed to either match or outperform both random and fixed selection, as well as the other hyper-heuristics. A number of hyper-heuristics managed to achieve this across multiple topology and trigger configurations, notably **RLFreq**, **HTour5X**, **HTour5M**, and **Freq5**. **RLFreq** was the clear top performer in the *Island_RT*, and *Island_PT* configurations, followed closely by **HTour5X** and **HTour5M**. Both **HTour5X** and **HTour5M** as well as **ETour25** were top contenders when the *Global_RT* and *Global_PT* configuration were used. Those same three hyper-heuristics, together with **RLfreq**, showed the strongest tendency to yield wins (as opposed to draws) when the *Global_ST* configuration was

used.

The heuristic space diversity and entity reassignment rates of select hyper-heuristics were also investigated to learn more about their modes of operation. The entity reassignment behavior of all hyper-heuristics (even random selection) were affected by the choice of trigger and topology. Many hyper-heuristics had different styles of entity reassignment behaviors for various environments when using different types of triggers or topologies. Hyper-heuristics with similar HSD and entity reassignment behavior to **Rand** had similar performance to **Rand**. The hyper-heuristics that relied on tournament selection between heuristic populations showed similar behavior to **Rand** by keeping very balanced entity-to-heuristic assignments at low tournament sizes. These same hyper-heuristics showed more definitive and assertive heuristic selection as tournament sizes increased. In contrast, the hyper-heuristics that relied on tournaments between individual entities tended to converge on a single heuristic when a small tournament size was used, and better adjusted the balance of entities across heuristics with a larger tournament size. Entity reassignment behavior is a very complex area, and future studies are needed to unpack and correlate different behaviors in more detail.

Chapter 7

Conclusions

This chapter presents the principal findings of this thesis. The investigation centered around hyper-heuristic approaches that manage various population-based SI and EC meta-heuristics that are specialized at solving continuous dynamic optimization problems. Section 7.1 briefly summarizes the main subject matter of each chapter. Section 7.2 discusses how the objectives of the thesis were achieved. Lastly, a body of future work arising from the research in the thesis is presented in section 7.3.

7.1 Summary

This thesis reviewed the use of a hyper-heuristic to manage which candidate solutions should be improved by which meta-heuristic over time. The overarching goals were to determine whether hyper-heuristic approaches could perform significantly better than their constituent heuristics on DOPs, to characterize how sensitive the performance of various hyper-heuristics are to hyper-heuristic framework parameter values, and to characterize the behavior of various hyper-heuristic selection operators in order to better understand their operation.

The concepts behind dynamic optimization for real-valued environments were briefly discussed in chapter 2. The DOP classification of Duhain and Engelbrecht [53] that combines the classifications of Angeline [4] and Eberhart *et al.* [56][84] with spatial and temporal change severity classes to uniquely define 27 environment types was presented. Parameter guidelines provided by Duhain and Engelbrecht were examined in concert with

the permissible parameter value ranges of the original scenario 2 settings [19][64][127] to yield 27 unique MPB environment definitions that are comparable to the majority of the DOP literature.

Comparison of the performance of stochastic CI optimization algorithms is a complex task which requires careful thought. The facets that influence the trustworthiness and quality of performance measures for DOPs were distilled from the literature. Good performance measures balance these facets to yield a faithful representation of an algorithm's performance on DOPs over time. Critique was also given on the common analysis practices used in CI studies to perform statistical significance testing of performance results. Critique was given on the common analysis practices used in CI studies to perform statistical significance testing of performance results. It was discussed how nonparametric statistical procedures are highly recommended in the literature, since these procedures have less restrictive assumptions than their parametric counterparts, and offer more resilient analysis capabilities that avoids false discoveries.

Chapter 3 discussed the field of hyper-heuristics as an approach to solve optimization problems. Hyper-heuristics were also contrasted against other existing control adaptation approaches from the fields of SI and EC. The HMHH framework by Grobler *et al.* [75][76] [74] was presented (using the terminology from the classification of hyper-heuristic approaches by Burke *et al.* [24]) as an example of a multi-point search selection hyper-heuristic that supports on-line learning and utilizes perturbative heuristics.

HMHH was extended with *neighborhood topologies* that control the extent to which the information of population members are visible across heuristics. The original HMHH scheme which allows information of all entities to be visible to all heuristics at all times was designated as the *global* topology. The proposed *island* topology restricts each heuristic to operate independently on only those entities that are assigned to the heuristic. The concept of *heuristic selection triggers* for HMHH was also proposed. The original HMHH *periodic trigger* (PT) performs heuristic selection simultaneously for all entities every k algorithm iterations. The newly proposed *stagnation trigger* (ST) changes an individual entity's heuristic if the entity's performance did not improve over k iterations. The *random trigger* (RT) probabilistically changes an individual entity's heuristic approximately every k iterations.

A critical discussion was provided that focused on the static *parameter tuning prob-*

lem and the dynamic *parameter control problem* for meta-heuristic methods. It was shown how static parameter tuning of SI and EC approaches is generally deemed to be an impossible task [46][97][103][**?**]. On the other hand, mixing together multiple self-adaptive parameter control methods in an ad hoc fashion is typically sub-optimal as well [97]. An outline was given of how hyper-heuristics address this conundrum by combining well-proven heuristics for a problem domain together with adaptable resource allocation logic to exploits the most suitable heuristic at the right time.

An overview was provided of how the necessary conditions for the *no free lunch* (NFL) theorems of Wolpert and Macready [184] are simply too restrictive to be found in practical continuous domain problems. The assessments of Schumacher *et al.* [157], Igel and Toussaint [88], Auger and Teytaud [5], Alabert *et al.* [3], Poli and Graff [149], and Kerschke *et al.* [98] imply the possibility that, in any practical situation, an intelligent selection hyper-heuristic could improve performance over using heuristics in isolation.

Various control groups were defined in chapter 4 that serve as an objective set of baselines to ground the analysis of all investigated hyper-heuristics against. The control groups addressed concerns around whether increased performance by any hyper-heuristics resulted from intelligent heuristic selection, or was simply due to the use of multiple sub-populations (speciation), simultaneously applying multiple methods using fixed entity allocations, or random heuristic selection. An experimental approach was laid out to systematically conduct all experimentation in this thesis while keeping the control groups in mind. The performance and diversity of the stand-alone and speciated versions of each heuristic were systematically explored in experiments. Rigorous nonparametric statistical tests demonstrated that the performance of the algorithms differed significantly across environments. It was shown how various heuristics excelled in different types of environments, which justified each of their inclusion into the heuristic pool.

A new performance measure called the *relative error distance*, P_r, was proposed that reports the normalized performance of an algorithm across search landscapes that result from environment changes relative to the best possible performance score. The outcome of the P_r measure is a single scalar value that faithfully reflects both the average performance over time, as well as the variance in performance values over time. The proposed measure does not assume normally distributed performance data across an algorithm

run, is resilient against fitness landscape scale changes, better incorporates performance variance across fitness landscape changes, and allows easier algorithm comparisons using established nonparametric statistical methods. An assessment was performed to illustrate that these conditions that the P_r measure aims to address were indeed present in the measured performance data of over 1 200 000 algorithm runs.

Chapter 5 explored the effect that different combinations of heuristic change triggers, heuristic change frequencies (k), and HMHH topologies had on the performance of various hyper-heuristics. The majority of investigated hyper-heuristic topology and trigger combinations generally performed on par with or better when $k = 20\%$ compared to configurations that used larger values for k. A strong correlation was found across most hyper-heuristic configurations where more frequent heuristic selection yielded a higher number of statistically significant wins than less frequent heuristic selection yielded. The newly proposed ST and RT triggers statistically either matched or exceeded the performance of the original PT trigger for nearly every investigated hyper-heuristic. However, per topology, certain hyper-heuristics performed better with one type of trigger than other triggers. Even under a random trigger, more frequent heuristic changes are beneficial, implying that simple algorithmic diversity is effective at improving performance.

Correlations were presented that show how certain hyper-heuristics, when operating on specific environments, performed better with one of the topologies than the other. The exact DOP at hand also had a strong influence on which topology was more suitable, regardless of the exact hyper-heuristic that was used. The *island* topology showed dramatically increased wins for 9 of the 27 environments, while the *global* topology was more suitable for 15 environments.

Chapter 6 directly compared the performance of various hyper-heuristics against each other and the control groups identified in chapter 4. The heuristic space diversity and entity reassignment rates of select hyper-heuristics were also investigated to learn more about their modes of operation. The following observations and conclusions were drawn:

- Every hyper-heuristic either matched or exceeded the performance of all of the standalone heuristics in every environment, never recording a single loss against any heuristic. Trigger and topology choices impacted the ability of each hyper-heuristic to yield a higher number of wins than draws against the standalone heuristics across more environments. The *global* topology showed the highest win count

across the most environments for half of the hyper-heuristics, while yielding similar performance to the *island* topology for the remainder of the hyper-heuristics.

- The hyper-heuristics recorded a high number of wins or draws against the homogeneous speciated heuristics across the majority of environments. No hyper-heuristic managed to record nine wins against all nine speciated heuristics, which resulted in a larger number of draws for a higher number of environments. Half of the hyper-heuristics recorded isolated losses against one or two speciated heuristics in a handful of environments, but the overwhelming majority of environments saw the hyper-heuristics either match or exceed the performance of the speciated heuristics.

- Most hyper-heuristics that used the *island* topology had a clear tendency to outperform fixed selection across the majority of environments, regardless of trigger choice, while yielding hardly any losses (the small remainder were draws).

- In contrast, every combination of trigger with the *global* topology resulted in losses against fixed heuristic selection for approximately one to five environments (depending on the exact hyper-heuristic), with isolated hyper-heuristics recording losses for up to 10 environments. The *Global_RT* and *Global_PT* configurations showed strong wins against fixed selection for those hyper-heuristics that did not rely on roulette wheel selection. Those hyper-heuristics based on tournament selection, when configured with larger tournament sizes, yielded wins for a more noteworthy number of environments.

- Almost no hyper-heuristic ever recorded losses against random heuristic selection in any noteworthy numbers of environments, regardless of topology or trigger choice. However, only **RLFreq** showed wins against random selection for a large number of environments, and that only when using the *Island_RT* configuration.

- In contrast, half of the hyper-heuristics recorded wins across roughly half of the environments when using either the *Global_RT* or *Global_PT* configurations. All hyper-heuristics yielded draws for most environments against random heuristic selection when using the *Global_ST* configuration.

- The entity reassignment behavior of all hyper-heuristics (even random selection) were affected by the choice of trigger and topology. Hyper-heuristics with similar HSD and entity reassignment behavior to **Rand** had similar performance to **Rand**. The hyper-heuristics that relied on tournament selection between heuristic populations showed similar behavior to **Rand** by keeping very balanced entity-to-heuristic assignments at low tournament sizes. These same hyper-heuristics showed more definitive and assertive heuristic selection as tournament sizes increased.

- In contrast, the hyper-heuristics that relied on tournaments between individual entities tended to converge on a single heuristic when a small tournament size was used, and better adjusted the balance of entities across heuristics with a larger tournament size.

7.2 Achievement of Objectives

Section 1.3 outlined the objectives of this thesis. The main goal of this thesis was to investigate how well various hyper-heuristic selection operators could continually balance computational resources across different population-based meta-heuristics in order to solve a DOP better than the individual meta-heuristics could. This overall goal and the listed sub-objectives in section 1.3 were achieved as follows:

- Existing hyper-heuristic algorithms, frameworks, and approaches that aim to solve DOPs were identified and reviewed. Emphasis was placed on population-based approaches and selection hyper-heuristics.

- The *moving peaks benchmark* (MPB) [19] was extended to support all 27 environment types as proposed by Duhain and Engelbrecht [53], and suitable parameters were identified that ensure the resulting environments complied with the generally accepted scenario 2 settings for the MPB as found in literature.

- The HMHH framework was applied to manage meta-heuristic algorithms that are specialized to solve DOPs, and was found to perform well against control group algorithms.

- The HMHH framework was extended with multiple neighborhood topologies, and their effectiveness was investigated across multiple hyper-heuristics and multiple types of DOPs.
- The HMHH framework was extended with different types of heuristic change triggers, and their effectiveness was investigated across multiple hyper-heuristics and multiple types of DOPs.
- A new measure called the *relative error distance*, or P_r, was proposed that does not assume normally distributed performance data, is resilient against fitness landscape scale changes, better incorporates performance variance across multiple fitness landscape changes, and allows easier algorithm comparisons using established nonparametric statistical methods.
- An extensive sensitivity analysis was performed that shows how sensitive different hyper-heuristics were to changes in the neighborhood topology and heuristic trigger (including different values for the heuristic change frequency parameter, namely k).
- The performance of various hyper-heuristics was compared against each other as well as various control groups comprising of individual heuristic, random heuristic selection, and fixed heuristic allocations.
- The heuristic space diversity and heuristic allocation behavior of various hyper-heuristics were explored to better understand the modes of operation of each method.

7.3 Future Work

A number of areas of future work were identified as part of the investigation of this thesis. The sections below outline a number of promising areas of further research.

7.3.1 (Self-)Adaptive Heuristic Pools

A control adaptation approach such as HMHH fundamentally operates by modifying the allocation of entities in the population to specific meta-heuristics. The majority

of investigated meta-heuristics were not necessarily designed to have members added or removed from their populations during their normal modes of operation. While the experiments in this thesis showed how HMHH was able to raise performance above using the meta-heuristics in isolation, additional performance gains may be possible if HMHH were able to "smooth over" heuristic reassignments in a more streamlined manner.

One promising avenue of research, as discussed in section 4.3.1, is to better integrate (self-)adaptive meta-heuristics into the heuristic pool. The goal would be to integrate (self-)adaptive methods in such a way that simultaneously maximizes the effectiveness of each heuristic after entities have been added or removed, while preventing (or managing) the (self-)adaptive mechanisms of the heuristics from "competing with" or counteracting the operation of the hyper-heuristic. In the case of PSO, a large body of theoretical studies have investigated the relationship between PSO parameter values (including population sizes) and convergence. Harrison *et al.* [78] investigated a number of self-adaptive PSO algorithms and whether algorithms with modified parameters respected certain well-known convergence criteria. They showed that over half of the algorithms exhibited divergent behavior while many others converged too soon. Cleghorn and Engelbrecht [33] showed how particle instability nearly always yielded performance that was worse than random. Numerous different types of meta-heuristics exist [60], and various methods will have their own specifics in how their control parameters could be adapted as the population size increases or decreases dynamically over time. Future studies should investigate the trade-off and synergies between the two levels of hyper-heuristics and meta-heuristic (self-)adaptation, especially in the management of which parameters are adapted, and in what manner.

Different types of meta-heuristics may also be sensitive to disturbances to their populations in different ways and at different times during the search. As an anecdotal example, it may possibly be better to reassign a particle that has approximately zero velocity than a particle with non-zero velocity, or to reassign a particle that has similar particles in its immediate area than a particle out on its own in a unique part of the search space. Future work should investigate different types of heuristic change triggers that are, perhaps, more cognizant of the inner working of each heuristic. Such insight could subsequently be valuable to help determine the most appropriate moment to trigger heuristic changes for entities.

7.3.2 Heuristic Allocation Behavior versus Performance Improvements

Chapter 6 investigated the heuristic space diversity, entity reassignment frequency, and entity reassignment rate behavior of a number of example hyper-heuristics. The entity assignment behavior of a hyper-heuristic is a complex time-dependent interaction between the optimization logic of each heuristic, the actual performance achieved by each entity using a specific heuristic, the (estimated) potential performance that an entity *could* potentially achieve using a heuristic, the detrimental and turbulent effect of heuristic reassignment, among many other factors. Different types of search landscapes and landscape change dynamics may also deeply affect all of these aspects. Much more future research should be devoted to understanding the interactions between these and other aspects to allow performance to be increased even further. The paragraphs below outline a number of possible ideas.

The trajectories of entities over time, i.e., a multi-dimensional vector that captures the behavioral aspects outlined in the previous paragraph, may be useful to design better heuristic change triggers. Future research should explore triggers that use such nuanced information, compared to simpler triggers which rely only on, for example, a fixed duration of time or stagnation of performance. Such multivariate trajectory information could also be helpful as another type of richer feedback that the hyper-heuristic could reason about (yielding a more elaborate version of $Q_{\delta_m}(t)$ if you will, as outlined in section 3.4.6). Research into such areas would be aligned to more recent trends in the hyper-heuristic field as identified by Swan *et al.* [168].

Another body of work should address the correlation between increased performance and certain types of behaviors. The incorporation of search landscape analysis features may play a role in the analysis, where the goal would be to jointly correlate the presence of certain search landscape characteristics at time t, the internal state of heuristics, the quality of the entities in the population, and the affect that different heuristic selection actions have on increased performance. Similar to late acceptance strategies in single-point search hyper-heuristics [24], insight into such correlations in population based selection hyper-heuristics would allow the creation of new hyper-heuristic methods that can react better to detected correlations between the problem space and heuristic space

states.

As discussed by Swan *et al.* [168], modern hyper-heuristics are starting to incorporate more of what the authors call "analytic information" to allow better entity-to-heuristic allocation. Change detection (both in temporal and spatial terms) was out of scope in this thesis to avoid biased results, as explained in section 4.3.1. Future work should focus on the incorporation of this information into the hyper-heuristic layer to allow the hyper-heuristic to make better informed decisions about entity allocation. For example, a hyper-heuristic could pertinently assign more entities to exploratory heuristics after a change is detected, and might do so in different ways depending on how severe those changes are.

The computational complexity of HMHH as a whole is influenced by the computational complexity of the underlying pool of heuristics. Allocating more entities to computationally complex heuristics could heavily influence the efficacy of HMHH as time goes on. For example, it is well-known that CPSO suffers from quadratic complexity [16], and continually allocating more entities to CPSO could result in HMHH becoming quadratically more expensive to run. Future work should focus on ways to make the hyper-heuristic layer more aware of the complexity of the underlying heuristics, both in the general sense as well as in combination with the current entity allocations at any given point in time of algorithm execution. Entity allocation decisions can then be made while taking the computational burden of the decision into account.

Lastly, the selection pressure of a hyper-heuristic and the consequent effect on heuristic space convergence (where all entities are operated upon by a single heuristic) and heuristic space diversity is complex. In this thesis, each heuristic maintained a fixed number of minimum entities (essentially, *sentinels* in heuristic space) to ensure that each heuristic always maintained a valid feedback score. The results in chapter 6 showed an intuitive correlation between higher selection pressure and increased performance. For example, almost every hyper-heuristic based on tournament selection showed better performance at higher tournament sizes (which is known to yield stronger selection pressure in these types of selection operators [60]). Future work should investigate this link more thoroughly, and perhaps even define new ways of enforcing the right level of selection pressure. Intuitive examples could include using more complex sources of feedback (as discussed above) to more accurately trigger entities, or to select the most

appropriate heuristics that would optimize selection pressure, or to use more deterministic approaches of allocating entities that enforce known selection pressure quotients by using the gradient of heuristic reassignment rate.

7.3.3 Complementary Heuristics

Chapter 3 outlined what the informal definition of "complementary heuristics" is, and how each heuristic in the heuristic pool should compensate for each other's weaknesses. A number of studies were highlighted that show how a complementary set of heuristics can be built to solve static optimization problems.

DOPs are more nuanced than static optimization problems in that the characteristics of the problem may change over time. Consequently, the complementary nature of the pool of heuristics may also change over time. Methods that worked well together for certain types of search landscapes and/or dynamics (in the same instance of a DOP) may not work well in subsequent search landscapes. Further research should be performed to formally define what a set of "complementary" heuristics looks like for hyper-heuristics that solve DOPs, and ways to automatically (or at least through a rigorous manual process) find the right number and set of complementary heuristics to include in a heuristic pool of a hyper-heuristic.

7.3.4 Comparative studies of P_r and existing DOP measures

Section 2.7.2 discussed the shortcomings and challenges faced by the most often-used DOP-focused performance measures. Section 4.4.2 highlighted how many of the listed shortcomings were present in the measurement data of over 1 200 000 experiment runs. The newly proposed P_r measure improves on existing measures by not assuming normally distributed performance data across an algorithm run, being resilient against fitness landscape scale changes, better incorporating performance variance across multiple fitness landscape changes, and allowing easier algorithm comparisons using established nonparametric statistical methods.

Future studies should focus on comparing P_r against the most-often used existing DOP performance measures. Specifically, comparative studies should explore the correlation between existing measures and P_r, and investigate where and how discrepancies

occur. Secondly, studies should be performed to determine if the outcome of performance assessments of various DOP-focused algorithms would be significantly impacted if P_r were to be used instead of other existing measures.

Bibliography

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[2] Antonio Aguilera and Ricardo Pérez-Aguila. General n-dimensional rotations. In *WSCG '2004: Short Communications: the 12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2004*, volume 12, No. 1-3, pages p. 1–8. UNION Agency - Science Press, Plzen, Czech Republic.

[3] Aureli Alabert, Alessandro Berti, Ricard Caballero, and Marco Ferrante. No-free-lunch theorems in the continuum. *Theoretical Computer Science*, 600:98 – 106, 2015.

[4] Peter J. Angeline. Tracking extrema in dynamic environments. In Peter J. Angeline, Robert G. Reynolds, John R. McDonnell, and Russell C. Eberhart, editors, *Evolutionary Programming VI*, pages 335–345, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[5] Anne Auger and Olivier Teytaud. Continuous lunches are free! In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 916–922, New York, NY, USA, 2007. ACM.

[6] Thomas Bäck. On the behavior of evolutionary algorithms in dynamic environments. In *1998 IEEE International Conference on Evolutionary Compu-*

tation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pages 446–451, May 1998.

[7] Ruibin Bai and Graham Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In Toshihide Ibaraki, Koji Nonobe, and Mutsunori Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 87–108, Boston, MA, 2005. Springer US.

[8] B. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In P. Bauer, G. Hommel, and E. Sonnemann, editors, *Multiple Hypothesenprüfung / Multiple Hypotheses Testing*, pages 100–115, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[9] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In Catriel Beeri and Peter Buneman, editors, *Database Theory — ICDT'99*, pages 217–235, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[10] Marco Biazzini, Balazs Banhelyi, Alberto Montresor, and Mark Jelasity. Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 1339–1346, New York, NY, USA, 2009. ACM.

[11] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[12] Tim M. Blackwell. Swarms in dynamic environments. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska, and Julian Miller, editors, *Genetic and Evolutionary Computation — GECCO 2003*, pages 1–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[13] Tim M. Blackwell and Peter J. Bentley. Don't push me! collision-avoiding swarms. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1691–1696 vol.2. IEEE, May 2002.

[14] Tim M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 19–26, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[15] Tim M. Blackwell and Jürgen Branke. Multi-swarm optimization in dynamic environments. In Günther R. Raidl, Stefano Cagnoni, Jürgen Branke, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Colin G. Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing*, pages 489–500, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[16] Tim M. Blackwell and Jürgen Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, Aug 2006.

[17] Tim M. Blackwell, Jürgen Branke, and Xiaodong Li. Particle swarms for dynamic optimization problems. In Christian Blum and Daniel Merkle, editors, *Swarm Intelligence: Introduction and Applications*, pages 193–217, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[18] Ron Bond, Andries P. Engelbrecht, and Beatrice Ombuki-Berman. Evaluating landscape characteristics of dynamic benchmark functions. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1343–1350. IEEE, May 2015.

[19] Jürgen Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1875–1882 Vol. 3. IEEE, July 1999.

[20] Jürgen Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2001.

[21] Jürgen Branke and Hartmut Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In Ashish Ghosh and Shigeyoshi Tsutsui, editors, *Advances in Evolutionary Computing: Theory and Applications*, pages 239–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[22] Janez Brest, Ales Zamuda, Borko Boskovic, Mirjam S. Maucec, and Viljem Zumer. Dynamic optimization using self-adaptive differential evolution. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pages 415–422. IEEE, May 2009.

[23] David V. Budescu and Mia Budescu. How to measure diversity when you must. *Psychological Methods*, 17(2):215–227, 2012.

[24] Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[25] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham*, 2010.

[26] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. A classification of hyper-heuristic approaches. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 449–468. Springer US, Boston, MA, 2010.

[27] Edmund K. Burke, Graham Kendall, Dario Landa Silva, Ross O'Brien, and Eric Soubeiga. An ant algorithm hyperheuristic for the project presentation scheduling problem. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2263–2270 Vol. 3. IEEE, Sep. 2005.

[28] Edmund K. Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, Dec 2003.

[29] Konstantin Chakhlevitch and Peter Cowling. Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In Günther R. Raidl and Jens Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 23–33, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[30] Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: Recent developments. In Carlos Cotta, Marc Sevaux, and Kenneth Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, pages 3–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[31] Pai-Chun Chen, Graham Kendall, and Greet Vanden Berghe. An ant based hyper-heuristic for the travelling tournament problem. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 19–26, April 2007.

[32] Christopher W. Cleghorn and Andries P. Engelbrecht. A generalized theoretical deterministic particle swarm model. *Swarm Intelligence*, 8(1):35–59, Mar 2014.

[33] Christopher W. Cleghorn and Andries P. Engelbrecht. Particle swarm optimizer: The impact of unstable particles on performance. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, Dec 2016.

[34] Christopher W. Cleghorn and Andries P. Engelbrecht. Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. *Swarm Intelligence*, 12(1):1–22, Mar 2018.

[35] Helen G. Cobb and John J. Grefenstette. Genetic algorithms for tracking changing environments. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 523–530, San Francisco, CA, USA, 1993. ACM, Morgan Kaufmann Publishers Inc.

[36] Carlos Cobos, Martha Mendoza, and Elizabeth León. A hyper-heuristic approach to design and tuning heuristic methods for web document clustering. In *Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1350–1358. IEEE, June 2011.

[37] P. Cowling, G. Kendall, and Limin Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1185–1190 vol.2. IEEE, May 2002.

[38] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 176–190, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[39] Teodor Gabriel Crainic and Michel Toulouse. Parallel strategies for meta-heuristics. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 475–513. Springer US, Boston, MA, 2003.

[40] Carlos Cruz, Juan R. González, and David A. Pelta. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, 15(7):1427–1448, Jul 2011.

[41] Swagatam Das, Ankush Mandal, and Rohan Mukherjee. An adaptive differential evolution algorithm for global optimization in dynamic environments. *IEEE Transactions on Cybernetics*, 44(6):966–978, June 2014.

[42] Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1 – 30, 2016.

[43] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, Feb 2011.

[44] Kenneth A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, 1975.

[45] Kenneth A. De Jong. Evolving in a changing world. In Zbigniew W. Raś and Andrzej Skowron, editors, *Foundations of Intelligent Systems*, pages 512–519, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[46] Kenneth A. De Jong. *Parameter Setting in EAs: a 30 Year Perspective*, pages 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[47] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[48] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[49] Marco Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[50] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477 Vol. 2. IEEE, July 1999.

[51] Mathys C. du Plessis and Andries P. Engelbrecht. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, 218(1):7 – 20, 2012.

[52] Mathys Cornelius Du Plessis. *Adaptive multi-population differential evolution for dynamic environments*. PhD thesis, University of Pretoria, South Africa, 2012.

[53] J.G.O.L. Duhain and A.P. Engelbrecht. Towards a more complete classification system for dynamically changing environments. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, June 2012.

[54] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[55] Russell C. Eberhart and Yuhui Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 84–88 vol.1. IEEE, July 2000.

[56] Russell C. Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 94–100 vol. 1. IEEE, May 2001.

[57] Agoston E. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and James E. Smith. Parameter control in evolutionary algorithms. In Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[58] Agoston E. Eiben and Selmar K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19 – 31, 2011.

[59] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Inc., USA, 2006.

[60] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, USA, 2nd edition, 2007.

[61] Wenyuan Feng, Torsten Brune, Lipton Chan, Munir Chowdhury, C Kuek, and Yun Li. Benchmarks for testing evolutionary algorithms. In *3rd Asia-Pacific Conference on Control and Measurement*, pages 134–138, 1998.

[62] Helmut Finner. On a monotonicity problem in step-down multiple test procedures. *Journal of the American Statistical Association*, 88(423):920–923, 1993.

[63] Henry Fisher and Gerald L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial Scheduling*, pages 225–251, 1963.

[64] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[65] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[66] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[67] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[68] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044 – 2064, 2010. Special Issue on Intelligent Distributed Information Systems.

[69] Salvador García and Francisco Herrera. An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677–2694, 2008.

[70] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617, May 2008.

[71] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300, 2015.

[72] John J Grefenstette. Genetic algorithms for changing environments. In *2nd International Conference on Parallel Problem Solving from Nature*, volume 2, pages 137–144, 1992.

[73] Jacomine Grobler. *The heterogeneous meta-hyper-heuristic: from low level heuristics to low level meta-heuristics*. PhD thesis, University of Pretoria, South Africa, 2015.

[74] Jacomine Grobler, Andries P. Engelbrecht, Graham Kendall, and Venkata S. S. Yadavalli. Alternative hyper-heuristic strategies for multi-method global optimization. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[75] Jacomine Grobler, Andries P. Engelbrecht, Graham Kendall, and Venkata S. S. Yadavalli. Multi-method algorithms: Investigating the entity-to-algorithm allocation problem. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 570–577. IEEE, June 2013.

[76] Jacomine Grobler, Andries P. Engelbrecht, Graham Kendall, and Venkata S. S. Yadavalli. Heuristic space diversity control for improved meta-hyper-heuristic performance. *Information Sciences*, 300:49–62, 2015.

[77] Limin Han and Graham Kendall. Guided operators for a hyper-heuristic genetic algorithm. In *Australasian Joint Conference on Artificial Intelligence*, pages 807–820. Springer, 2003.

[78] Kyle R. Harrison, Andries P. Engelbrecht, and Beatrice M. Ombuki-Berman. Self-adaptive particle swarm optimization: a review and analysis of convergence. *Swarm Intelligence*, 12(3):187–226, 2018.

[79] Kyle R. Harrison, Beatrice M. Ombuki-Berman, and Andries P. Engelbrecht. The effect of probability distributions on the performance of quantum particle swarm optimization for solving dynamic optimization problems. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 242–250, Dec 2015.

[80] Yosef Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.

[81] Burt S. Holland and Margaret DiPonzio Copenhaver. An improved sequentially rejective bonferroni test procedure. *Biometrics*, 43(2):417–423, 1987.

[82] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[83] Gerhard Hommel. A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, 75(2):383–386, 1988.

[84] Xiaohui Hu and Russell C. Eberhart. Tracking dynamic systems with PSO: where's the cheese. In *Proceedings of the Workshop on Particle Swarm Optimization, Purdue School of Engineering and Technology*, pages 80–83. 2001.

[85] Xiaohui Hu and Russell C. Eberhart. Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1666–1670 vol.2. IEEE, May 2002.

[86] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997.

[87] Sheldon Hui and Ponnuthurai N. Suganthan. Niching-based self-adaptive ensemble de with mmts for solving dynamic optimization problems. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1536–1541. IEEE, July 2014.

[88] Christian Igel and Marc Toussaint. On classes of functions for which no free lunch results hold. *Information Processing Letters*, 6(86):317–321, 2003.

[89] Ronald L. Iman and James M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1980.

[90] Sk. Minhazul Islam, Swagatam Das, Saurav Ghosh, Subhrajit Roy, and Ponnuthurai N. Suganthan. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):482–500, April 2012.

[91] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer for dynamic optimization problems. In Günther R. Raidl, Stefano Cagnoni, Jürgen Branke, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Colin G. Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni

Squillero, editors, *Applications of Evolutionary Computing*, pages 513–524, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[92] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):329–354, Dec 2006.

[93] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, June 2005.

[94] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, http://www.scipy.org/, 2001 – 2019.

[95] Ahmad R. Jordehi. Particle swarm optimisation for dynamic optimisation problems: a review. *Neural Computing and Applications*, 25(7-8):1507–1516, 2014. Springer.

[96] Masoud Kamosi, Ali B. Hashemi, and Mohammad R. Meybodi. A new particle swarm optimization algorithm for dynamic environments. In Bijaya Ketan Panigrahi, Swagatam Das, Ponnuthurai N. Suganthan, and Subhransu S. Dash, editors, *Swarm, Evolutionary, and Memetic Computing*, pages 129–138, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[97] Giorgos Karafotias, Mark Hoogendoorn, and Agoston E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, April 2015.

[98] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. MIT Press.

[99] Berna Kiraz, A. Şima Etaner-Uyar, and Ender Özcan. An ant-based selection hyper-heuristic for dynamic environments. In Anna I. Esparcia-Alcázar, editor, *Applications of Evolutionary Computation*, pages 626–635, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[100] Berna Kiraz, A. Şima Uyar, and Ender Özcan. An investigation of selection hyper-heuristics in dynamic environments. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcázar, Juan J. Merelo, Ferrante Neri, Mike Preuss, Hendrik Richter, Julian Togelius, and Georgios N. Yannakakis, editors, *Applications of Evolutionary Computation*, pages 314–323, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[101] Natalio Krasnogor and Jim Smith. A memetic algorithm with self-adaptive local search: Tsp as a case study. In *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*, GECCO'00, pages 987–994, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[102] Thiemo Krink, Jakob S. Vesterstrøm, and Jacques Riget. Particle swarm optimisation with spatial particle extension. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1474–1479 vol.2, May 2002.

[103] Barend J. Leonard and Andries P. Engelbrecht. On the optimality of particle swarm parameters in dynamic environments. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 1564–1569. IEEE, June 2013.

[104] Changhe Li, Trung Thanh Nguyen, Ming Yang, Michalis Mavrovouniotis, and Shengxiang Yang. An adaptive multipopulation framework for locating and tracking multiple optima. *IEEE Transactions on Evolutionary Computation*, 20(4):590–605, 2016.

[105] Changhe Li and Shengxiang Yang. Fast multi-swarm optimization for dynamic optimization problems. In *2008 Fourth International Conference on Natural Computation*, volume 7, pages 624–628. IEEE, Oct 2008.

[106] Changhe Li and Shengxiang Yang. A clustering particle swarm optimizer for dynamic optimization. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pages 439–446, May 2009.

[107] Changhe Li, Shengxiang Yang, and Trung Thanh Nguyen. A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):627–646, 2012.

[108] Changhe Li, Shengxiang Yang, TT Nguyen, E Ling Yu, Xin Yao, Yaochu Jin, HG Beyer, and PN Suganthan. Benchmark generator for CEC 2009 competition on dynamic optimization. Technical report, Department of Computer Science, University of Leicester, 2008.

[109] Jianjun (David) Li. A two-step rejection procedure for testing multiple hypotheses. *Journal of Statistical Planning and Inference*, 138(6):1521 – 1527, 2008.

[110] Xiaodong Li. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In Kalyanmoy Deb, editor, *Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation – GECCO 2004*, pages 105–116, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[111] Xiaodong Li, Jürgen Branke, and Tim Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 51–58, New York, NY, USA, 2006. ACM.

[112] Xiaodong Li and Khanh Hoa Dam. Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the 2003 Congress on Evolutionary Computation, CEC '03.*, volume 3, pages 1772–1779 Vol.3. IEEE, Dec 2003.

[113] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz. *Parameter Setting in Evolutionary Algorithms*, volume 54. Springer Publishing Company, Incorporated, 1st edition, 2007.

[114] Robert Duncan Luce. *Individual Choice Behavior a Theoretical Analysis*. John Wiley and sons, Oxford, England, 1959.

[115] Rodica Ioana Lung and Dumitru Dumitrescu. A collaborative model for tracking optima in dynamic environments. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pages 564–567. IEEE, Sep. 2007.

[116] Wenjian Luo, Juan Sun, Chenyang Bu, and Houjun Liang. Species-based particle swarm optimizer enhanced by memory for dynamic optimization. *Applied Soft Computing*, 47:130 – 140, 2016.

[117] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[118] Jorge Maturana, Frédéric Lardeux, and Frédéric Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 16(6):881–909, Dec 2010.

[119] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1 – 17, 2017.

[120] Michalis Mavrovouniotis and Shengxiang Yang. Population-based incremental learning with immigrants schemes in changing environments. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1444–1451. IEEE, Dec 2015.

[121] David Meignan, Abderrafiaa Koukam, and Jean-Charles Créput. Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics*, 16(6):859–879, Dec 2010.

[122] Rui Mendes and Arvind S Mohais. DynDE: a differential evolution for dynamic optimization problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2808–2815 Vol. 3. IEEE, Sep. 2005.

[123] Efrñn Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 485–492, New York, NY, USA, 2006. ACM.

[124] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK, 1996.

[125] Ronald W. Morrison. Performance measurement in dynamic environments. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, number 5-8. Citeseer, 2003.

[126] Ronald W. Morrison and Kenneth A. De Jong. Measurement of population diversity. In Pierre Collet, Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, and Marc Schoenauer, editors, *Artificial Evolution*, pages 31–41, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[127] Irene Moser and Raymond Chiong. Dynamic function optimization: The moving peaks benchmark. In Enrique Alba, Amir Nakib, and Patrick Siarry, editors, *Metaheuristics for Dynamic Optimization*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[128] Irene Moser and Tim Hendtlass. A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pages 252–259. IEEE, Sep. 2007.

[129] Rohan Mukherjee, Gyana Ranjan Patra, Rupam Kundu, and Swagatam Das. Cluster-based differential evolution with crowding archive for niching in dynamic environments. *Information Sciences*, 267:58 – 82, 2014.

[130] Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer Decision-Making*, pages 523–544. Springer US, Boston, MA, 2004.

[131] Peter Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

[132] Filipe V. Nepomuceno and Andries P. Engelbrecht. A self-adaptive heterogeneous PSO inspired by ants. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders Lyhne Christensen, Andries P. Engelbrecht, Roderich Groß, and Thomas

Stützle, editors, *Swarm Intelligence*, volume 7461 of *Lecture Notes in Computer Science*, pages 188–195, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[133] Filipe V. Nepomuceno and Andries P. Engelbrecht. Behavior changing schedules for heterogeneous particle swarms. In *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pages 112–118, Sep. 2013.

[134] Filipe V. Nepomuceno and Andries P. Engelbrecht. A self-adaptive heterogeneous PSO for real-parameter optimization. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 361–368. IEEE, June 2013.

[135] Trung Thanh Nguyen. *Continuous dynamic optimisation using evolutionary algorithms*. PhD thesis, University of Birmingham, 2011.

[136] Trung Thanh Nguyen, Shengxiang Yang, and Jürgen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

[137] Trung Thanh Nguyen and Xin Yao. Continuous dynamic constrained optimization – the challenges. *IEEE Transactions on Evolutionary Computation*, 16(6):769–786, Dec 2012.

[138] Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh, and Reza Safabakhsh. Evaluating the performance of DNPSO in dynamic environments. In *Proceedings of the 2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 2640–2645. IEEE, Oct 2008.

[139] Olusegun Olorunda and Andries P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1128–1134. IEEE, June 2008.

[140] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):141–152, Feb 2006.

[141] Ender Özcan, Yuri Bykov, Murat Birben, and Edmund K Burke. Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pages 997–1004. IEEE, May 2009.

[142] Ender Özcan, Mustafa Misir, Gabriela Ochoa, and Edmund K. Burke. A reinforcement learning-great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 1(1):39–59, January 2010.

[143] Ender Ozcan, Sima Etaner Uyar, and Edmund Burke. A greedy hyper-heuristic in dynamic environments. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2201–2204, New York, NY, USA, 2009. ACM.

[144] Daniel Parrott and Xiaodong Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10(4):440–458, Aug 2006.

[145] K. E. Parsopoulos and M. N. Vrahatis. Unified particle swarm optimization in dynamic environments. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing*, pages 590–599, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[146] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[147] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800, Oct 2010.

[148] Christian Perwass, Herbert Edelsbrunner, Leif Kobbelt, and Konrad Polthier. Algebra. In *Geometric Algebra with Applications in Engineering*, pages 51–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[149] Riccardo Poli and Mario Graff. There is a free lunch for hyper-heuristics, genetic programming and computer scientists. In Leonardo Vanneschi, Steven Gustafson, Alberto Moraglio, Ivanoe De Falco, and Marc Ebner, editors, *Genetic Programming*, pages 195–207, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[150] A. Kai Qin and Ponnuthurai N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791 Vol. 2. IEEE, Sep. 2005.

[151] William Rand and Rick Riolo. Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions. In *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, GECCO '05, pages 32–38, New York, NY, USA, 2005. ACM.

[152] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33, 2011.

[153] Zhilei Ren, He Jiang, Jifeng Xuan, and Zhongxuan Luo. Ant based hyper heuristics with space reduction: A case study of the p-median problem. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 546–555, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[154] Ismael Rodríguez-Fdez, Adrián Canosa, Manuel Mucientes, and Alberto Bugarín. STAC: A web platform for the comparison of algorithms using statistical tests. In *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, Aug 2015.

[155] Dror M Rom. A sequentially rejective test procedure based on a modified bonferroni inequality. *Biometrika*, 77(3):663–665, 1990.

[156] Matthew H. P. R. Sankey. Introductory note on the thermal efficiency of steam-engines. In *Minutes of Proceedings of the Institution of Civil Engineers*, volume 134, 1898.

[157] Chris Schumacher, Michael D Vose, and L Darrell Whitley. The no free lunch and problem description length. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 565–570. Morgan Kaufmann Publishers Inc., 2001.

[158] Carlos Segura, Carlos A. Coello Coello, Eduardo Segredo, and Coromoto León. On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, 9(1):189–198, Jan 2015.

[159] Juliet P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

[160] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[161] Samuel S. Shapiro and Martin B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[162] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC Press, 4 edition, 2007.

[163] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015. Wiley Online Library.

[164] Kenneth Sörensen and Fred W. Glover. Metaheuristics. In Saul I. Gass and Michael C. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 960–970. Springer US, Boston, MA, 2013.

[165] Paolo Spanevello and Marco A. Montes de Oca. Experiments on adaptive heterogeneous PSO algorithms. In *Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, pages 36–40, 2009.

[166] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[167] Jerry Swan, Steven Adriaensen, Mohamed Bishr, Edmund K. Burke, John A. Clark, Patrick De Causmaecker, Juanjo Durillo, Kevin Hammond, Emma Hart, Colin G. Johnson, Zoltan A. Kocsis, Ben Kovitz, Krzysztof Krawiec, Simon Martin, J. J. Merelo, Leandro L. Minku, Ender Özcan, Gisele L. Pappa, Erwin Pesch, Pablo Garcia-Sànchez, Andrea Schaerf, Kevin Sim, Jim Smith, Thomas Stützle, Vo Stefan, Stefan Wagner, and Xin Yao. A research agenda for metaheuristic standardization. In *Proceedings of the XI Metaheuristics International Conference*, pages 1–3, Agadir, United Kingdom, 2015.

[168] Jerry Swan, Patrick De Causmaecker, Simon Martin, and Ender Özcan. A re-characterization of hyper-heuristics. In Lionel Amodeo, El-Ghazali Talbi, and Farouk Yalaoui, editors, *Recent Developments in Metaheuristics*, pages 75–89. Springer International Publishing, Cham, 2018.

[169] Ke Tang, Fei Peng, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94 – 104, 2014.

[170] Jay D. Teachman. Analysis of population diversity: Measures of qualitative variation. *Sociological Methods & Research*, 8(3):341–362, 1980.

[171] Haluk R. Topcuoglu, Abdulvahid Ucar, and Lokman Altin. A hyper-heuristic based framework for dynamic optimization problems. *Applied Soft Computing*, 19(0):236 – 251, 2014.

[172] Krzysztof Trojanowski and Zbigniew Michalewicz. Searching for optima in non-stationary environments. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3. IEEE, 1999.

[173] Gönül Uludağ, Berna Kiraz, A. Şima Etaner-Uyar, and Ender Özcan. A hybrid multi-population framework for dynamic environments combining online and offline learning. *Soft Computing*, 17(12):2327–2348, Dec 2013.

[174] Stefan A.G. Van der Stockt and Andries P. Engelbrecht. Analysis of hyper-heuristic performance in different dynamic environments. In *Proceedings of the 2014 IEEE*

Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), pages 1–8, Dec 2014.

[175] Stefan A.G. Van der Stockt and Andries P. Engelbrecht. Analysis of global information sharing in hyper-heuristics for different dynamic environments. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 822–829, Sendai, Japan, 2015.

[176] Stefan A.G. van der Stockt and Andries P. Engelbrecht. Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization. *Swarm and Evolutionary Computation*, 43:127 – 146, 2018.

[177] Jasper A. Vrugt and Bruce A. Robinson. Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711, 2007.

[178] Hongfeng Wang, Na Wang, and Dingwei Wang. Multi-swarm optimization algorithm for dynamic optimization problems using forking. In *2008 Chinese Control and Decision Conference*, pages 2415–2419. IEEE, 2008.

[179] Yu Wang, Bin Li, Thomas Weise, Jianyu Wang, Bo Yuan, and Qiongjie Tian. Self-adaptive learning based particle swarm optimization. *Information Sciences*, 181(20):4515 – 4538, 2011. Special Issue on Interpretable Fuzzy Systems.

[180] Karsten Weicker. An analysis of dynamic severity and population size. In Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, pages 159–168, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[181] Karsten Weicker. Performance measures for dynamic environments. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, Hans-Paul Schwefel, and José-Luis Fernández-Villacañas, editors, *Parallel Problem Solving from Nature — PPSN VII*, pages 64–73, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[182] Karsten Weicker and Nicole Weicker. On evolution strategy optimization in dynamic environments. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 2039–2046 Vol. 3. IEEE, July 1999.

[183] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[184] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[185] John Woodward, Jerry Swan, and Simon Martin. The 'composite' design pattern in metaheuristics. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pages 1439–1444, New York, NY, USA, 2014. ACM.

[186] Guohua Wu, Rammohan Mallipeddi, Ponnuthurai N. Suganthan, Rui Wang, and Huangke Chen. Differential evolution with multi-population based ensemble of mutation strategies. *Information Sciences*, 329:329 – 345, 2016. Special issue on Discovery Science.

[187] Guohua Wu, Xin Shen, Haifeng Li, Huangke Chen, Anping Lin, and Ponnuthurai N Suganthan. Ensemble of differential evolution variants. *Information Sciences*, 423:172–186, 2018.

[188] Shengxiang Yang and Changhe Li. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 14(6):959–974, Dec 2010.

[189] Xiangwei Zheng and Hong Liu. A different topology multi-swarm PSO in dynamic environment. In *Proceedings of the 2009 IEEE International Symposium on IT in Medicine Education*, volume 1, pages 790–795, Aug 2009.

Appendix A

Additional Results – Chapter 6

This chapter presents additional results that were omitted from chapter 6 due to space constraints. The following tables are listed:

- Research question 1 focused on how each hyper-heuristic performs relative to the pool of stand-alone heuristics. Tables A.1, A.2, A.3, A.4, A.5, and A.6 show the detailed results for each hyper-heuristic for each environment and for each configuration.
- Research question 2 performed the same analysis as used in research question 1 to compare the hyper-heuristics against the nine homogeneous speciated heuristics Tables A.7, A.8, A.9, A.10, A.11, and A.12 in appendix A show the detailed results for each environment.
- Research question 3 compared how well the hyper-heuristics perform relative to each other, fixed heuristic selection, and random heuristic selection. Tables A.14, A.16, A.18, A.20, A.22, and A.24 which show the detailed wins, draws, and losses of each of the six comparisons in each environment, while tables A.13, A.15, A.17, A.19, A.21, and A.23 show the associated Friedman ranks and p-values.
- Research question 4 explored the entity reassignment characteristics of each hyper-heuristic. Figures A.1 through to figure A.21 present the detailed results for the measures introduced in the research question.

Table A.1: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *global* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	5-4-0	5-4-0	5-4-0	5-4-0	8-1-0	6-3-0	9-0-0	5-4-0	9-0-0	5-4-0	5-4-0	4-5-0	5-4-0	9-0-0	9-0-0	7-2-0	8-1-0	8-1-0	5-4-0	5-4-0	5-4-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	8-1-0	6-3-0	9-0-0	6-3-0	8-1-0	6-3-0	6-3-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	7-2-0	9-0-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	8-1-0	6-3-0	9-0-0	6-3-0	9-0-0	6-3-0	6-3-0	5-4-0	6-3-0	9-0-0	8-1-0	6-3-0	8-1-0	9-0-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A2L	5-4-0	4-5-0	4-5-0	5-4-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	5-4-0	5-4-0	3-6-0	4-5-0	9-0-0	8-1-0	7-2-0	8-1-0	8-1-0	5-4-0	5-4-0	5-4-0
A2R	6-3-0	6-3-0	6-3-0	6-3-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	5-4-0	6-3-0	6-3-0	5-4-0	9-0-0	8-1-0	8-1-0	8-1-0	9-0-0	6-3-0	9-0-0	6-3-0
A3C	8-1-0	6-3-0	6-3-0	6-3-0	8-1-0	8-1-0	8-1-0	6-3-0	8-1-0	6-3-0	6-3-0	6-3-0	6-3-0	9-0-0	8-1-0	8-1-0	9-0-0	9-0-0	6-3-0	6-3-0	7-2-0
A3L	8-1-0	6-3-0	6-3-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A3R	6-3-0	6-3-0	6-3-0	6-3-0	8-1-0	8-1-0	8-1-0	7-2-0	9-0-0	7-2-0	7-2-0	7-2-0	6-3-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	7-2-0	7-2-0
C1C	7-2-0	7-2-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	7-2-0	7-2-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	7-2-0	6-3-0
C1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0
C2C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	7-2-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0
C2L	7-2-0	8-1-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	7-2-0	7-2-0	8-1-0	9-0-0	9-0-0	7-2-0	7-2-0	9-0-0	7-2-0	7-2-0	8-1-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	7-2-0	7-2-0	5-4-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	7-2-0	6-3-0	5-4-0	6-3-0	9-0-0	7-2-0	7-2-0	7-2-0	8-1-0	5-4-0	6-3-0	6-3-0
P1L	7-2-0	5-4-0	5-4-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	7-2-0	4-5-0	5-4-0	9-0-0	9-0-0	9-0-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0
P1R	7-2-0	5-4-0	4-5-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	6-3-0	4-5-0	4-5-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	6-3-0
P2C	8-1-0	5-4-0	5-4-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	5-4-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0
P2L	6-3-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	5-4-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	6-3-0
P2R	6-3-0	5-4-0	5-4-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	6-3-0	5-4-0	5-4-0	8-1-0	8-1-0	7-2-0	6-3-0	5-4-0	6-3-0	8-1-0	8-1-0
P3C	7-2-0	7-2-0	5-4-0	7-2-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	7-2-0	7-2-0	5-4-0	6-3-0	7-2-0	8-1-0	7-2-0	7-2-0	9-0-0	5-4-0	7-2-0	7-2-0
P3L	6-3-0	6-3-0	5-4-0	7-2-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	7-2-0	5-4-0	5-4-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0	7-2-0	7-2-0
P3R	7-2-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	4-5-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	7-2-0
μ_{wins}	7.04	6.44	6.11	7.44	8.15	7.96	8.26	7.96	8.33	7.07	6.93	6	6.19	8.15	7.93	7.63	7.7	8	6.85	7.22	7.04
σ_{wins}	1.16	1.4	1.48	1.34	0.72	0.94	0.76	1.19	0.73	1.21	1.21	1.69	1.57	0.77	0.68	0.74	0.82	0.96	1.29	1.15	1.19
μ_{draws}	1.96	2.56	2.89	1.56	0.85	1.04	0.74	1.04	0.67	1.93	2.07	3	2.81	0.85	1.07	1.37	1.3	1	2.15	1.78	1.96
σ_{draws}	1.16	1.4	1.48	1.34	0.72	0.94	0.76	1.19	0.73	1.21	1.21	1.69	1.57	0.77	0.68	0.74	0.82	0.96	1.29	1.15	1.19
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.2: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *global* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	8-1-0	6-3-0	8-1-0	6-3-0	6-3-0	5-4-0	5-4-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	5-4-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	9-0-0	6-3-0	9-0-0	6-3-0	6-3-0	6-3-0	6-3-0	8-1-0	8-1-0	7-2-0	7-2-0	8-1-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	9-0-0	6-3-0	9-0-0	6-3-0	6-3-0	6-3-0	6-3-0	8-1-0	9-0-0	6-3-0	6-3-0	8-1-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A2L	5-4-0	8-1-0	5-4-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	6-3-0	5-4-0	5-4-0	6-3-0	8-1-0	7-2-0	8-1-0	8-1-0	5-4-0	5-4-0	5-4-0
A2R	9-0-0	6-3-0	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	9-0-0	6-3-0	6-3-0	6-3-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	6-3-0
A3C	9-0-0	6-3-0	9-0-0	8-1-0	9-0-0	8-1-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0
A3L	8-1-0	6-3-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A3R	7-2-0	7-2-0	6-3-0	7-2-0	8-1-0	6-3-0	9-0-0	7-2-0	8-1-0	8-1-0	8-1-0	5-4-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0
C1C	6-3-0	7-2-0	6-3-0	5-4-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0	6-3-0	6-3-0	5-4-0	7-2-0	7-2-0	8-1-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0
C1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0
C2C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	7-2-0	9-0-0	9-0-0	9-0-0
C2L	7-2-0	6-3-0	6-3-0	8-1-0	9-0-0	7-2-0	9-0-0	8-1-0	8-1-0	7-2-0	7-2-0	5-4-0	5-4-0	7-2-0	9-0-0	9-0-0	5-4-0	7-2-0	7-2-0	7-2-0	9-0-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	7-2-0	7-2-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0
C3L	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	6-3-0	6-3-0	5-4-0	5-4-0	7-2-0	7-2-0	7-2-0	7-2-0	5-4-0	5-4-0	7-2-0	5-4-0	5-4-0	5-4-0	7-2-0	9-0-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0
P1L	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	4-5-0	4-5-0	5-4-0	6-3-0	8-1-0	5-4-0	4-5-0	5-4-0	5-4-0	6-3-0
P1R	6-3-0	6-3-0	5-4-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	6-3-0	4-5-0	4-5-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	5-4-0	7-2-0	7-2-0
P2C	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	7-2-0	7-2-0	7-2-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0
P2L	6-3-0	5-4-0	5-4-0	5-4-0	7-2-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	9-0-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0
P2R	6-3-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	8-1-0	5-4-0	5-4-0	5-4-0	7-2-0	5-4-0
P3C	6-3-0	7-2-0	6-3-0	6-3-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	7-2-0	9-0-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0
P3L	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0
P3R	7-2-0	6-3-0	6-3-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	4-5-0	5-4-0	7-2-0	7-2-0	8-1-0	7-2-0	7-2-0	6-3-0	7-2-0	7-2-0
μ_{wins}	7	6.7	6.59	7.04	7.7	7.22	7.74	7.26	7.52	7.04	7.19	6.22	6.33	6.93	7.56	7.93	6.78	6.96	6.78	7.19	7
σ_{wins}	1.36	1.27	1.45	1.48	1.1	1.25	1.26	1.29	1.25	1.19	1.33	1.58	1.54	1.3	1.09	0.92	1.25	1.22	1.55	1.33	1.49
μ_{draws}	2	2.3	2.41	1.96	1.3	1.78	1.26	1.74	1.48	1.96	1.81	2.78	2.67	2.07	1.44	1.07	2.22	2.04	2.22	1.81	2
σ_{draws}	1.36	1.27	1.45	1.48	1.1	1.25	1.26	1.29	1.25	1.19	1.33	1.58	1.54	1.3	1.09	0.92	1.25	1.22	1.55	1.33	1.49
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.3: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *global* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A2L	8-1-0	8-1-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	6-3-0	6-3-0	5-4-0	5-4-0	7-2-0	5-4-0	5-4-0	6-3-0	5-4-0	5-4-0
A2R	9-0-0	6-3-0	5-4-0	9-0-0	9-0-0	9-0-0	6-3-0	9-0-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	8-1-0	6-3-0	9-0-0	6-3-0	9-0-0	6-3-0	9-0-0
A3C	9-0-0	8-1-0	6-3-0	8-1-0	9-0-0	9-0-0	8-1-0	9-0-0	8-1-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	6-3-0	9-0-0	8-1-0
A3L	8-1-0	7-2-0	6-3-0	7-2-0	7-2-0	8-1-0	7-2-0	6-3-0	6-3-0	8-1-0	8-1-0	6-3-0	8-1-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0
A3R	5-4-0	7-2-0	6-3-0	7-2-0	7-2-0	6-3-0	7-2-0	7-2-0	7-2-0	6-3-0	6-3-0	6-3-0	5-4-0	7-2-0	8-1-0	6-3-0	8-1-0	6-3-0	7-2-0	7-2-0	8-1-0
C1C	7-2-0	6-3-0	6-3-0	7-2-0	7-2-0	6-3-0	7-2-0	6-3-0	7-2-0	7-2-0	7-2-0	6-3-0	7-2-0	7-2-0	7-2-0	8-1-0	7-2-0	6-3-0	7-2-0	7-2-0	8-1-0
C1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
C2C	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0
C2L	9-0-0	6-3-0	5-4-0	9-0-0	8-1-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0	5-4-0	5-4-0	7-2-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	8-1-0	9-0-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0
C3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	7-2-0	8-1-0	7-2-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	7-2-0	5-4-0	6-3-0	6-3-0	6-3-0	5-4-0	7-2-0	6-3-0	7-2-0	6-3-0	7-2-0	5-4-0	5-4-0	6-3-0	6-3-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	9-0-0
P1L	6-3-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	4-5-0	4-5-0	5-4-0	6-3-0	8-1-0	5-4-0	5-4-0	6-3-0	5-4-0	7-2-0
P1R	8-1-0	7-2-0	6-3-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	4-5-0	4-5-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0
P2C	8-1-0	6-3-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	5-4-0	7-2-0	8-1-0	7-2-0	5-4-0	8-1-0	8-1-0	8-1-0
P2L	8-1-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	8-1-0	8-1-0	7-2-0	5-4-0	5-4-0	5-4-0	7-2-0	9-0-0	6-3-0	5-4-0	7-2-0	8-1-0	8-1-0
P2R	6-3-0	6-3-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	7-2-0	7-2-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	8-1-0	5-4-0	5-4-0	6-3-0	6-3-0	5-4-0
P3C	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0	7-2-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	9-0-0
P3L	7-2-0	6-3-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	7-2-0	6-3-0	5-4-0	6-3-0	5-4-0	6-3-0	7-2-0	6-3-0	6-3-0	7-2-0	7-2-0	7-2-0
P3R	8-1-0	7-2-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0	8-1-0	7-2-0	8-1-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0
μ_{wins}	7.52	6.78	6.41	7.22	7.22	7.19	7.04	7.26	7.22	7.37	7.37	6.37	6.56	6.52	7.19	7.63	7.04	6.74	7.26	7.37	7.67
σ_{wins}	1.25	1.34	1.34	1.31	1.31	1.55	1.19	1.38	1.01	1.21	1.24	1.5	1.53	1.25	1.18	1.15	1.26	1.29	1.16	1.11	1.27
μ_{draws}	1.48	2.22	2.59	1.78	1.78	1.81	1.96	1.74	1.78	1.63	1.63	2.63	2.44	2.48	1.81	1.37	1.96	2.26	1.74	1.63	1.33
σ_{draws}	1.25	1.34	1.34	1.31	1.31	1.55	1.19	1.38	1.01	1.21	1.24	1.5	1.53	1.25	1.18	1.15	1.26	1.29	1.16	1.11	1.27
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.4: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *island* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	7-2-0	7-2-0	7-2-0	7-2-0	5-4-0	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A2L	5-4-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	5-4-0	5-4-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0	5-4-0	5-4-0	6-3-0
A2R	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	9-0-0
A3C	9-0-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	9-0-0
A3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0
A3R	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	9-0-0	8-1-0	7-2-0	8-1-0	7-2-0	8-1-0
C1C	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0
C1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0
C2C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C2L	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	3-6-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	7-2-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3L	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	9-0-0	8-1-0	8-1-0	9-0-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0
P1L	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	7-2-0	5-4-0	4-5-0	5-4-0	5-4-0	4-5-0
P1R	6-3-0	4-5-0	4-5-0	6-3-0	7-2-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	6-3-0	4-5-0	5-4-0	4-5-0	4-5-0	7-2-0	7-2-0	4-5-0	6-3-0	6-3-0	5-4-0
P2C	7-2-0	7-2-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	5-4-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	6-3-0	7-2-0
P2L	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0	5-4-0	4-5-0	4-5-0	7-2-0	5-4-0	4-5-0	5-4-0	4-5-0	5-4-0
P2R	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	8-1-0	5-4-0	4-5-0	4-5-0	4-5-0	5-4-0
P3C	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0	7-2-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0
P3L	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	6-3-0	5-4-0	4-5-0	5-4-0	4-5-0	4-5-0
P3R	6-3-0	4-5-0	5-4-0	6-3-0	7-2-0	6-3-0	7-2-0	7-2-0	7-2-0	6-3-0	7-2-0	4-5-0	4-5-0	4-5-0	4-5-0	7-2-0	7-2-0	6-3-0	6-3-0	6-3-0	7-2-0
μ_{wins}	6.48	6.26	6.26	6.89	7.11	6.96	7.19	6.93	7.11	6.74	6.7	6.07	6.22	6.19	6.41	7.59	7.04	6.52	6.63	6.37	6.7
σ_{wins}	1.65	1.89	1.83	1.5	1.48	1.51	1.47	1.52	1.48	1.58	1.61	1.82	1.8	2	1.97	1.25	1.53	1.89	1.67	1.6	1.73
μ_{draws}	2.52	2.74	2.74	2.11	1.89	2.04	1.81	2.07	1.89	2.26	2.3	2.93	2.78	2.81	2.59	1.41	1.96	2.48	2.37	2.63	2.3
σ_{draws}	1.65	1.89	1.83	1.5	1.48	1.51	1.47	1.52	1.48	1.58	1.61	1.82	1.8	2	1.97	1.25	1.53	1.89	1.67	1.6	1.73
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.5: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *island* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
A2L	5-4-0	5-4-0	5-4-0	5-4-0	8-1-0	7-2-0	5-4-0	8-1-0	8-1-0	6-3-0	6-3-0	5-4-0	5-4-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	6-3-0	6-3-0
A2R	9-0-0	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
A3C	9-0-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	6-3-0	8-1-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
A3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0
A3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
C1C	5-4-0	5-4-0	4-5-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	7-2-0	6-3-0	4-5-0	5-4-0	5-4-0	5-4-0
C1L	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0
C2C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C2L	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	3-6-0	4-5-0	5-4-0	5-4-0	4-5-0	5-4-0	4-5-0	5-4-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	6-3-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P1L	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	5-4-0	4-5-0	4-5-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	7-2-0	5-4-0	4-5-0	5-4-0	4-5-0	5-4-0
P1R	5-4-0	4-5-0	4-5-0	5-4-0	6-3-0	5-4-0	7-2-0	5-4-0	6-3-0	6-3-0	6-3-0	5-4-0	4-5-0	4-5-0	4-5-0	7-2-0	5-4-0	4-5-0	6-3-0	6-3-0	6-3-0
P2C	7-2-0	5-4-0	5-4-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	5-4-0	5-4-0	7-2-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0
P2L	5-4-0	4-5-0	4-5-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	7-2-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P2R	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	4-5-0	6-3-0	5-4-0	4-5-0	5-4-0	4-5-0	4-5-0
P3C	6-3-0	4-5-0	4-5-0	5-4-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	7-2-0	6-3-0	5-4-0	5-4-0	5-4-0	5-4-0
P3L	5-4-0	4-5-0	4-5-0	4-5-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P3R	7-2-0	6-3-0	5-4-0	5-4-0	7-2-0	5-4-0	7-2-0	5-4-0	7-2-0	5-4-0	7-2-0	6-3-0	5-4-0	4-5-0	4-5-0	7-2-0	6-3-0	4-5-0	6-3-0	5-4-0	5-4-0
μ_{wins}	6.67	6.04	6.11	6.56	6.89	6.63	6.81	6.7	6.93	6.74	6.7	6.07	6.15	6.22	6.37	7.3	6.81	6.41	6.78	6.67	6.59
σ_{wins}	1.69	1.85	1.95	1.74	1.63	1.71	1.59	1.68	1.71	1.65	1.66	1.66	1.88	2.06	2.06	1.27	1.59	2.02	1.58	1.73	1.67
μ_{draws}	2.33	2.96	2.89	2.44	2.11	2.37	2.19	2.3	2.07	2.26	2.3	2.93	2.85	2.78	2.63	1.7	2.19	2.59	2.22	2.33	2.41
σ_{draws}	1.69	1.85	1.95	1.74	1.63	1.71	1.59	1.68	1.71	1.65	1.66	1.66	1.88	2.06	2.06	1.27	1.59	2.02	1.58	1.73	1.67
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.6: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of nine heuristics. Each hyper-heuristic used the *island* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	5-4-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	6-3-0
A1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A1R	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
A2C	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0
A2L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	6-3-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	7-2-0	6-3-0	8-1-0	8-1-0	5-4-0	8-1-0	8-1-0	8-1-0
A2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
A3C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0
A3L	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	8-1-0
A3R	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	9-0-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
C1C	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	5-4-0	4-5-0	5-4-0	4-5-0	4-5-0	6-3-0	6-3-0	5-4-0	5-4-0	6-3-0	5-4-0
C1L	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0	6-3-0
C1R	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	7-2-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0
C2C	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	3-6-0	9-0-0	9-0-0	9-0-0
C2L	4-5-0	5-4-0	4-5-0	5-4-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	3-6-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0
C2R	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0
C3C	9-0-0	9-0-0	8-1-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	9-0-0	8-1-0
C3L	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0	8-1-0	8-1-0	9-0-0	8-1-0
C3R	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0	8-1-0
P1C	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	6-3-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P1L	4-5-0	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	5-4-0	4-5-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0
P1R	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	6-3-0	7-2-0	6-3-0	5-4-0	5-4-0	4-5-0	4-5-0	7-2-0	6-3-0	4-5-0	6-3-0	6-3-0	6-3-0
P2C	7-2-0	7-2-0	7-2-0	8-1-0	8-1-0	7-2-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	8-1-0	8-1-0	8-1-0	7-2-0	7-2-0	8-1-0	8-1-0
P2L	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0
P2R	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	4-5-0	6-3-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P3C	5-4-0	5-4-0	5-4-0	6-3-0	5-4-0	6-3-0	6-3-0	5-4-0	6-3-0	6-3-0	5-4-0	5-4-0	5-4-0	4-5-0	5-4-0	4-5-0	6-3-0	4-5-0	6-3-0	6-3-0	5-4-0
P3L	5-4-0	4-5-0	4-5-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	5-4-0	4-5-0	4-5-0	4-5-0	5-4-0	5-4-0	4-5-0	5-4-0	5-4-0	5-4-0
P3R	7-2-0	7-2-0	6-3-0	6-3-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	7-2-0	6-3-0	5-4-0	4-5-0	4-5-0	7-2-0	7-2-0	4-5-0	7-2-0	7-2-0	7-2-0
μ_{wins}	6.85	6.7	6.56	6.89	6.93	6.93	6.89	7	7	6.96	6.85	6.56	6.52	6.19	6.3	7.07	6.93	6.22	6.89	7	6.85
σ_{wins}	1.75	1.79	1.83	1.65	1.62	1.66	1.53	1.62	1.54	1.6	1.77	1.83	1.85	2.04	2.02	1.49	1.49	2.03	1.63	1.73	1.66
μ_{draws}	2.15	2.3	2.44	2.11	2.07	2.07	2.11	2	2	2.04	2.15	2.44	2.48	2.81	2.7	1.93	2.07	2.78	2.11	2	2.15
σ_{draws}	1.75	1.79	1.83	1.65	1.62	1.66	1.53	1.62	1.54	1.6	1.77	1.83	1.85	2.04	2.02	1.49	1.49	2.03	1.63	1.73	1.66
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.7: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *global* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	4-7-0	4-7-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	8-3-0	6-5-0	5-6-0	4-5-2	4-6-1	8-3-0	8-3-0	6-5-0	6-5-0	7-4-0	4-7-0	6-5-0	4-7-0
A1L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0
A1R	6-5-0	6-5-0	5-6-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	7-4-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0
A2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0	4-7-0	3-8-0	4-7-0	3-8-0	2-7-2	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A2L	4-7-0	4-7-0	4-7-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	4-5-2	4-7-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	5-6-0
A2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A3C	4-7-0	4-5-2	4-6-1	4-7-0	4-6-1	4-7-0	4-6-1	4-7-0	4-7-0	4-7-0	4-7-0	4-5-2	4-6-1	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-6-1	4-7-0	4-7-0
A3L	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
A3R	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0
C1C	8-3-0	7-4-0	5-6-0	8-3-0	7-4-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
C1L	7-4-0	7-4-0	7-4-0	8-3-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	8-3-0	7-4-0	8-3-0	8-3-0	7-4-0	7-4-0	8-3-0	7-4-0	8-3-0	7-4-0
C1R	9-2-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
C2C	4-6-1	4-7-0	4-4-3	4-3-4	4-3-4	4-4-3	4-3-4	4-4-3	4-3-4	4-7-0	4-4-3	4-7-0	4-6-1	4-3-4	4-3-4	4-3-4	4-3-4	4-3-4	4-7-0	4-6-1	4-4-3
C2L	8-3-0	8-3-0	8-3-0	9-2-0	9-2-0	9-2-0	9-2-0	9-2-0	9-2-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	9-2-0	8-3-0	8-3-0	8-3-0	8-3-0	9-2-0	9-2-0
C2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-5-2	4-7-0	4-7-0	4-7-0	4-7-0	4-5-2	4-5-2	4-7-0	4-7-0	4-5-2	4-7-0	4-7-0	4-7-0
C3C	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	7-4-0
P1C	8-3-0	8-3-0	4-7-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	4-7-0	6-5-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P1L	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	4-7-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P1R	8-3-0	5-6-0	5-6-0	8-3-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P2C	4-7-0	3-8-0	3-7-1	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	3-8-0	4-7-0	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
P2L	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	8-3-0	4-7-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P2R	8-3-0	7-4-0	4-7-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	6-5-0	4-7-0	4-7-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0
P3C	8-3-0	7-4-0	4-7-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	4-7-0	5-6-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P3L	8-3-0	5-6-0	4-7-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	8-3-0	5-6-0	4-7-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0
P3R	6-5-0	5-6-0	5-6-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	8-3-0
μ_{wins}	6.22	5.44	4.89	6.37	6.15	6.37	6.33	6.33	6.22	6.3	6.07	4.89	5.04	6.33	6.22	6.15	6.11	6.3	6.04	6.26	6.26
σ_{wins}	1.72	1.4	1.22	1.71	1.56	1.67	1.71	1.71	1.6	1.64	1.57	1.34	1.13	1.75	1.95	1.66	1.63	1.66	1.7	1.72	1.72
μ_{draws}	4.74	5.48	5.93	4.48	4.67	4.52	4.44	4.56	4.56	4.7	4.81	5.89	5.85	4.44	4.44	4.7	4.74	4.48	4.93	4.7	4.63
σ_{draws}	1.68	1.37	1.21	1.67	1.49	1.6	1.55	1.65	1.5	1.64	1.52	1.34	1.06	1.65	1.65	1.64	1.61	1.55	1.66	1.68	1.67
μ_{loss}	0.04	0.07	0.19	0.15	0.19	0.11	0.22	0.11	0.22	0	0.11	0.22	0.11	0.22	0.33	0.15	0.15	0.22	0.04	0.04	0.11
σ_{loss}	0.19	0.38	0.62	0.77	0.79	0.58	0.8	0.58	0.85	0	0.58	0.64	0.32	0.85	0.92	0.77	0.77	0.85	0.19	0.19	0.58

Table A.8: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *global* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	4-6-1	4-7-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	4-5-2	6-5-0	6-5-0
A1L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	7-4-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0
A1R	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0
A2C	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A2L	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0
A2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A3C	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0
A3L	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
A3R	6-5-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0
C1C	8-3-0	6-5-0	5-6-0	5-6-0	6-5-0	7-4-0	7-4-0	7-4-0	5-6-0	7-4-0	8-3-0	5-6-0	5-6-0	5-6-0	7-4-0	8-3-0	7-4-0	5-6-0	6-5-0	8-3-0	6-5-0
C1L	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	8-3-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	8-3-0
C1R	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0
C2C	4-7-0	4-4-3	4-4-3	4-6-1	4-6-1	4-7-0	4-3-4	4-4-3	4-3-4	4-7-0	5-6-0	4-7-0	4-7-0	4-3-4	4-3-4	4-3-4	4-3-4	4-3-4	4-7-0	4-7-0	4-7-0
C2L	8-3-0	8-3-0	7-4-0	9-2-0	9-2-0	9-2-0	9-2-0	9-2-0	9-2-0	8-3-0	8-3-0	7-4-0	8-3-0	8-3-0	10-1-0	10-1-0	8-3-0	9-2-0	8-3-0	8-3-0	9-2-0
C2R	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-5-2	4-7-0	4-5-2	4-7-0	5-6-0	4-7-0	4-7-0	4-5-2	4-5-2	4-7-0	4-5-2	4-5-2	5-6-0	4-7-0	4-7-0
C3C	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-6-1	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
P1C	8-3-0	4-7-0	4-7-0	4-7-0	6-5-0	7-4-0	6-5-0	6-5-0	4-7-0	7-4-0	8-3-0	5-6-0	4-7-0	4-7-0	6-5-0	8-3-0	6-5-0	4-7-0	6-5-0	8-3-0	8-3-0
P1L	8-3-0	5-6-0	4-7-0	5-6-0	5-6-0	8-3-0	8-3-0	5-6-0	5-6-0	8-3-0	5-6-0	4-7-0	4-7-0	5-6-0	5-6-0	8-3-0	7-4-0	4-7-0	6-5-0	6-5-0	8-3-0
P1R	5-6-0	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	5-6-0	8-3-0	5-6-0	8-3-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	8-3-0	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0
P2C	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
P2L	8-3-0	4-7-0	4-7-0	8-3-0	7-4-0	6-5-0	6-5-0	5-6-0	5-6-0	8-3-0	8-3-0	4-7-0	4-7-0	5-6-0	7-4-0	8-3-0	8-3-0	4-7-0	5-6-0	8-3-0	8-3-0
P2R	7-4-0	4-7-0	4-7-0	4-7-0	5-6-0	8-3-0	4-7-0	4-7-0	5-6-0	7-4-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	4-7-0	4-7-0	4-7-0	7-4-0	8-3-0
P3C	8-3-0	5-6-0	4-7-0	5-6-0	7-4-0	8-3-0	8-3-0	8-3-0	4-7-0	8-3-0	8-3-0	5-6-0	5-6-0	4-7-0	5-6-0	8-3-0	8-3-0	5-6-0	7-4-0	8-3-0	8-3-0
P3L	8-3-0	6-5-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	8-3-0	5-6-0	6-5-0	6-5-0	6-5-0	8-3-0	8-3-0	5-6-0	6-5-0	8-3-0	8-3-0
P3R	5-6-0	5-6-0	5-6-0	8-3-0	7-4-0	7-4-0	6-5-0	8-3-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	5-6-0	5-6-0	6-5-0	7-4-0
μ_{wins}	6.19	5.19	5.11	5.74	5.96	6.26	5.93	6	5.41	6.22	6.07	5	5	5.22	5.48	6.3	5.85	5.07	5.33	6.22	6.33
σ_{wins}	1.39	1.18	1.09	1.53	1.4	1.58	1.47	1.57	1.37	1.42	1.36	1.04	1.11	1.31	1.4	1.75	1.51	1.24	1.18	1.4	1.59
μ_{draws}	4.81	5.67	5.78	5.22	5	4.74	4.85	4.89	5.37	4.78	4.93	5.96	6	5.52	5.26	4.56	4.93	5.7	5.59	4.78	4.67
σ_{draws}	1.39	1.14	1.12	1.5	1.36	1.58	1.41	1.53	1.39	1.42	1.36	1.02	1.11	1.31	1.38	1.72	1.47	1.32	1.15	1.4	1.59
μ_{loss}	0	0.15	0.11	0.04	0.04	0	0.22	0.11	0.22	0	0	0.04	0	0.26	0.26	0.15	0.22	0.22	0.07	0	0
σ_{loss}	0	0.6	0.58	0.19	0.19	0	0.85	0.58	0.85	0	0	0.19	0	0.86	0.86	0.77	0.85	0.85	0.38	0	0

Table A.9: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *global* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0
A1L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
A1R	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0
A2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A2L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0
A2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A3C	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	4-7-0	5-6-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0
A3L	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
A3R	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C1C	7-4-0	5-6-0	5-6-0	7-4-0	7-4-0	5-6-0	7-4-0	5-6-0	7-4-0	7-4-0	8-3-0	5-6-0	5-6-0	6-5-0	7-4-0	8-3-0	7-4-0	4-7-0	8-3-0	8-3-0	8-3-0
C1L	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	8-3-0	8-3-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	9-2-0
C1R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
C2C	4-7-0	4-4-3	4-3-4	4-5-2	4-4-3	4-3-4	4-3-4	4-4-3	4-5-2	4-4-3	4-7-0	4-5-2	4-7-0	4-3-4	4-3-4	4-4-3	4-3-4	4-3-4	5-6-0	4-7-0	4-7-0
C2L	9-2-0	8-3-0	7-4-0	9-2-0	8-3-0	9-2-0	8-3-0	9-2-0	8-3-0	9-2-0	9-2-0	7-4-0	7-4-0	9-2-0	9-2-0	10-1-0	8-3-0	9-2-0	9-2-0	8-3-0	9-2-0
C2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-3-4	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-6-1	4-5-2	4-7-0	4-7-0	4-7-0
C3C	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0	4-7-0	4-7-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
P1C	7-4-0	4-7-0	5-6-0	6-5-0	6-5-0	4-7-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	4-7-0	4-7-0	6-5-0	5-6-0	8-3-0	7-4-0	4-7-0	8-3-0	8-3-0	8-3-0
P1L	7-4-0	4-7-0	4-7-0	7-4-0	5-6-0	6-5-0	8-3-0	6-5-0	5-6-0	8-3-0	8-3-0	4-7-0	4-7-0	5-6-0	5-6-0	8-3-0	7-4-0	5-6-0	8-3-0	6-5-0	8-3-0
P1R	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	6-5-0	8-3-0	8-3-0	5-6-0	5-6-0	5-6-0	7-4-0	8-3-0	8-3-0	5-6-0	8-3-0	8-3-0	8-3-0
P2C	4-7-0	3-7-1	3-7-1	4-7-0	4-7-0	4-7-0	3-8-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0	3-7-1	4-7-0	4-7-0	4-7-0
P2L	8-3-0	5-6-0	5-6-0	5-6-0	7-4-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	5-6-0	4-7-0	5-6-0	8-3-0	8-3-0	8-3-0	5-6-0	8-3-0	8-3-0	8-3-0
P2R	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	5-6-0	4-7-0	8-3-0	8-3-0	6-5-0	7-4-0	4-7-0	4-7-0	6-5-0	4-7-0	8-3-0	6-5-0	4-7-0	7-4-0	4-7-0	6-5-0
P3C	8-3-0	5-6-0	7-4-0	8-3-0	8-3-0	5-6-0	7-4-0	5-6-0	7-4-0	8-3-0	8-3-0	4-7-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	4-7-0	8-3-0	8-3-0	8-3-0
P3L	8-3-0	5-6-0	4-7-0	8-3-0	7-4-0	8-3-0	8-3-0	6-5-0	7-4-0	8-3-0	8-3-0	4-7-0	4-7-0	6-5-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	8-3-0	8-3-0
P3R	8-3-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	7-4-0	6-5-0	8-3-0	8-3-0	5-6-0	5-6-0	7-4-0	6-5-0	8-3-0	6-5-0	5-6-0	7-4-0	6-5-0	7-4-0
μ_{wins}	6.11	5.11	5.15	5.93	5.89	5.78	5.7	5.93	5.96	6.3	6.3	5.04	5.07	5.56	5.7	6.33	5.85	5.15	6.3	6.07	6.44
σ_{wins}	1.6	1.19	1.13	1.47	1.37	1.42	1.44	1.41	1.4	1.56	1.59	1.13	1.11	1.28	1.46	1.75	1.38	1.26	1.59	1.52	1.63
μ_{draws}	4.89	5.74	5.67	5	5	5.07	5	4.96	4.96	4.59	4.7	5.89	5.93	5.26	5.15	4.56	4.96	5.59	4.7	4.93	4.56
σ_{draws}	1.6	1.16	1.18	1.41	1.33	1.44	1.47	1.37	1.34	1.5	1.59	1.12	1.11	1.26	1.49	1.69	1.34	1.28	1.59	1.52	1.63
μ_{loss}	0	0.15	0.19	0.07	0.11	0.15	0.3	0.11	0.07	0.11	0	0.07	0	0.19	0.15	0.11	0.19	0.26	0	0	0
σ_{loss}	0	0.6	0.79	0.38	0.58	0.77	1.07	0.58	0.38	0.58	0	0.38	0	0.79	0.77	0.58	0.79	0.86	0	0	0

Table A.10: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *island* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	5-6-0	5-6-0	8-3-0	7-4-0	8-3-0	7-4-0	8-3-0	8-3-0	6-5-0	6-5-0	5-6-0	6-5-0	5-6-0	4-7-0	7-4-0	6-5-0	6-5-0	5-6-0	7-4-0	6-5-0
A1L	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0
A1R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0
A2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A2L	5-6-0	5-6-0	4-7-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	5-6-0	4-7-0	4-7-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	4-7-0	5-6-0	5-6-0
A2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	6-5-0	4-7-0	6-5-0	4-7-0	4-7-0
A3C	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0
A3L	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
A3R	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0
C1C	5-6-0	5-6-0	5-6-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	7-4-0	5-6-0	5-6-0	5-6-0	5-6-0	8-3-0	7-4-0	5-6-0	6-5-0	7-4-0	6-5-0
C1L	6-5-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0
C1R	6-5-0	6-5-0	6-5-0	7-4-0	9-2-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0
C2C	4-6-1	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	4-3-4	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	5-6-0
C2L	7-4-0	7-4-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	7-4-0	7-4-0	4-6-1	7-4-0	8-3-0	7-4-0	6-5-0	8-3-0	8-3-0	8-3-0
C2R	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	4-7-0	4-5-2	4-7-0	5-6-0	6-5-0	6-5-0	5-6-0	4-7-0	4-7-0	4-7-0
C3C	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-4-1	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
P1C	6-5-0	6-5-0	6-5-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	7-4-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	6-5-0	6-5-0	5-6-0
P1L	7-4-0	4-7-0	4-7-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	4-7-0	5-6-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	8-3-0	8-3-0
P1R	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	5-6-0	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	5-6-0	8-3-0	6-5-0	5-6-0
P2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	6-5-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0
P2L	5-6-0	5-6-0	4-7-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	8-3-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0
P2R	7-4-0	4-7-0	4-7-0	6-5-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	7-4-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	7-4-0	4-7-0	4-7-0	4-7-0	4-7-0
P3C	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	6-5-0	4-7-0	5-6-0	6-5-0	4-7-0
P3L	8-3-0	5-6-0	6-5-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	6-5-0	5-6-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	5-6-0	5-6-0
P3R	6-5-0	5-6-0	5-6-0	6-5-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	5-6-0	5-6-0	4-7-0	5-6-0	8-3-0	8-3-0	6-5-0	6-5-0	5-6-0	6-5-0
μ_{wins}	5.52	5.07	4.96	6.26	6.56	6.48	6.67	6.48	6.78	5.7	5.81	4.96	4.96	4.74	4.96	6.74	6.22	5	5.52	5.52	5.3
σ_{wins}	1.12	0.87	0.9	1.46	1.53	1.5	1.33	1.42	1.22	1.3	1.42	0.94	0.94	0.81	0.94	1.1	1.15	0.83	1.37	1.22	1.07
μ_{draws}	5.44	5.93	6.04	4.74	4.44	4.52	4.33	4.52	4.22	5.3	5.19	6.04	5.81	6.19	6.04	4.26	4.78	6	5.48	5.48	5.7
σ_{draws}	1.09	0.87	0.9	1.46	1.53	1.5	1.33	1.42	1.22	1.3	1.42	0.94	1.08	0.88	0.94	1.1	1.15	0.83	1.37	1.22	1.07
μ_{loss}	0.04	0	0	0	0	0	0	0	0	0	0	0	0.22	0.07	0	0	0	0	0	0	0
σ_{loss}	0.19	0	0	0	0	0	0	0	0	0	0	0	0.85	0.27	0	0	0	0	0	0	0

Table A.11: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *island* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	5-6-0	4-6-1	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	4-7-0	4-7-0	4-7-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	6-5-0	6-5-0
A1L	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
A1R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
A2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0
A2L	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	4-7-0	4-7-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0
A2R	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
A3C	5-6-0	4-7-0	4-7-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	5-6-0
A3L	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0
A3R	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0
C1C	7-4-0	5-6-0	5-6-0	8-3-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	5-6-0	4-6-1	4-7-0	8-3-0	6-5-0	5-6-0	6-5-0	7-4-0	7-4-0
C1L	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0
C1R	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C2C	5-6-0	4-7-0	4-4-3	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-6-1	5-6-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	4-5-2	4-7-0
C2L	7-4-0	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0	7-4-0	4-6-1	5-6-0	8-3-0	7-4-0	7-4-0	8-3-0	7-4-0	7-4-0
C2R	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	4-7-0	4-7-0	5-6-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	4-7-0	4-7-0	4-7-0
C3C	5-6-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-4-1	6-5-0	6-4-1	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	7-4-0	6-5-0
P1C	7-4-0	4-7-0	4-7-0	8-3-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	6-5-0	8-3-0	5-6-0	4-7-0	4-7-0	4-7-0	8-3-0	6-5-0	4-7-0	8-3-0	7-4-0	6-5-0
P1L	5-6-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	8-3-0	7-4-0	7-4-0	8-3-0	5-6-0	5-6-0	4-7-0	4-7-0	4-7-0	8-3-0	7-4-0	4-7-0	8-3-0	8-3-0	8-3-0
P1R	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	8-3-0	5-6-0	8-3-0	7-4-0	8-3-0	5-6-0	5-6-0	5-6-0	5-6-0	8-3-0	5-6-0	4-7-0	8-3-0	6-5-0	6-5-0
P2C	4-7-0	4-7-0	3-8-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	3-8-0	4-7-0	4-7-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
P2L	5-6-0	4-7-0	4-7-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	4-7-0	4-7-0	8-3-0	5-6-0	4-7-0	5-6-0	5-6-0	5-6-0
P2R	4-7-0	4-7-0	4-7-0	7-4-0	4-7-0	4-7-0	7-4-0	7-4-0	7-4-0	4-7-0	4-7-0	4-7-0	4-7-0	4-5-2	4-7-0	8-3-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0
P3C	7-4-0	4-7-0	4-7-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	7-4-0	4-7-0	4-7-0	4-6-1	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	5-6-0	5-6-0
P3L	8-3-0	5-6-0	4-7-0	6-5-0	6-5-0	6-5-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	5-6-0	7-4-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	6-5-0	8-3-0
P3R	8-3-0	6-5-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	8-3-0	5-6-0	8-3-0	6-5-0	6-5-0	4-7-0	5-6-0	8-3-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0
μ_{wins}	5.74	4.93	4.67	5.7	5.56	5.41	5.96	5.74	6.15	5.59	5.81	4.93	4.93	4.67	5	6.56	5.7	5.04	5.93	5.48	5.56
σ_{wins}	1.23	0.92	0.96	1.17	1.15	0.97	1.26	1.1	1.1	1.28	1.33	0.96	1	0.78	0.83	1.25	1.03	0.94	1.3	1.12	1.19
μ_{draws}	5.26	6.07	6.19	5.3	5.44	5.59	5.04	5.26	4.85	5.37	5.19	6.04	6.07	6.11	6	4.44	5.3	5.96	5.07	5.44	5.44
σ_{draws}	1.23	0.92	1.04	1.17	1.15	0.97	1.26	1.1	1.1	1.24	1.33	1.02	1	0.85	0.83	1.25	1.03	0.94	1.3	1.09	1.19
μ_{loss}	0	0	0.15	0	0	0	0	0	0	0.04	0	0.04	0	0.22	0	0	0	0	0	0.07	0
σ_{loss}	0	0	0.6	0	0	0	0	0	0	0.19	0	0.19	0	0.51	0	0	0	0	0	0.38	0

Table A.12: Wins, draws and losses per environment for each hyper-heuristic as compared individually against the pool of eleven speciated heuristics. Each hyper-heuristic used the *island* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses against each heuristic.

Env	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	4-7-0	5-6-0	6-5-0	5-6-0	4-7-0	5-6-0	6-5-0	6-5-0
A1L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0
A1R	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
A2C	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	6-5-0	4-7-0	6-5-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	6-5-0
A2L	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	4-7-0	6-5-0	6-5-0	5-6-0
A2R	6-5-0	6-5-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	6-5-0	6-5-0
A3C	6-5-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0
A3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	6-5-0	5-6-0
A3R	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0
C1C	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	7-4-0	8-3-0	8-3-0	8-3-0	8-3-0	6-5-0	5-6-0	5-6-0	4-7-0	5-6-0	8-3-0	8-3-0	5-6-0	8-3-0	7-4-0	7-4-0
C1L	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	7-4-0	7-4-0	7-4-0
C1R	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C2C	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0
C2L	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	7-4-0	8-3-0	8-3-0	8-3-0	7-4-0	7-4-0	6-5-0	6-5-0	4-6-1	5-6-0	8-3-0	6-5-0	5-6-0	8-3-0	7-4-0	7-4-0
C2R	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	4-7-0	5-6-0	5-6-0	4-7-0	5-6-0	6-5-0	6-5-0	6-5-0	5-6-0	4-7-0	5-6-0
C3C	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	5-6-0	6-5-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	5-6-0	6-5-0	6-5-0	5-6-0	6-5-0	6-5-0	5-6-0
C3L	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0
C3R	6-5-0	6-5-0	6-5-0	6-5-0	7-4-0	6-5-0	7-4-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-5-0	6-4-1	6-5-0	7-4-0	6-5-0	5-6-0	6-5-0	6-5-0	6-5-0
P1C	6-5-0	4-7-0	6-5-0	7-4-0	8-3-0	7-4-0	8-3-0	7-4-0	7-4-0	8-3-0	7-4-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	6-5-0	6-5-0
P1L	5-6-0	4-7-0	5-6-0	6-5-0	8-3-0	8-3-0	8-3-0	8-3-0	7-4-0	6-5-0	6-5-0	5-6-0	4-7-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	7-4-0	7-4-0
P1R	8-3-0	6-5-0	5-6-0	8-3-0	8-3-0	5-6-0	8-3-0	6-5-0	8-3-0	8-3-0	6-5-0	5-6-0	5-6-0	4-6-1	4-7-0	8-3-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0
P2C	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0
P2L	5-6-0	4-7-0	5-6-0	5-6-0	6-5-0	5-6-0	8-3-0	5-6-0	5-6-0	8-3-0	5-6-0	4-7-0	4-7-0	4-7-0	4-7-0	8-3-0	5-6-0	4-7-0	6-5-0	8-3-0	6-5-0
P2R	4-7-0	4-7-0	4-7-0	5-6-0	8-3-0	4-7-0	7-4-0	4-7-0	8-3-0	8-3-0	6-5-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0	4-7-0
P3C	6-5-0	5-6-0	4-7-0	8-3-0	8-3-0	7-4-0	7-4-0	6-5-0	7-4-0	7-4-0	5-6-0	4-7-0	5-6-0	4-7-0	4-7-0	8-3-0	7-4-0	4-7-0	7-4-0	5-6-0	5-6-0
P3L	5-6-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	8-3-0	7-4-0	8-3-0	5-6-0	4-7-0	4-7-0	8-3-0	8-3-0	4-7-0	8-3-0	8-3-0	8-3-0
P3R	8-3-0	8-3-0	6-5-0	6-5-0	8-3-0	8-3-0	8-3-0	6-5-0	8-3-0	8-3-0	8-3-0	6-5-0	5-6-0	4-7-0	4-7-0	6-5-0	5-6-0	5-6-0	8-3-0	8-3-0	8-3-0
μ_{wins}	5.7	5.41	5.19	6.04	6.37	6	6.33	6.07	6.37	6.41	5.85	5.22	5.07	4.67	4.85	6.44	5.81	4.81	6.22	6.15	6.15
σ_{wins}	1.1	1.01	0.79	1.19	1.36	1.18	1.41	1.11	1.21	1.19	0.99	0.97	0.78	0.83	0.82	1.37	1.18	0.79	1.34	1.17	1.03
μ_{draws}	5.3	5.59	5.81	4.96	4.63	5	4.67	4.93	4.63	4.59	5.15	5.78	5.93	6.22	6.15	4.56	5.19	6.19	4.78	4.85	4.85
σ_{draws}	1.1	1.01	0.79	1.19	1.36	1.18	1.41	1.11	1.21	1.19	0.99	0.97	0.78	0.89	0.82	1.37	1.18	0.79	1.34	1.17	1.03
μ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0.11	0	0	0	0	0	0	0
σ_{loss}	0	0	0	0	0	0	0	0	0	0	0	0	0	0.32	0	0	0	0	0	0	0

Table A.13: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *global* topology and the RT trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm	p values
A1C	11.66	14.83	11.80	13.56	12.24	8.34	10.68	6.39	11.35	6.56	14.97	15.21	16.75	16.03	**4.62**	6.92	8.94	8.46	7.85	16.44	14.01	15.38	1.110223e-16
A1L	10.87	14.97	11.33	12.84	12.03	9.14	10.63	7.23	9.86	8.21	13.51	13.83	14.61	15.36	**6.17**	8.85	9.72	10.32	7.92	17.07	14.18	14.35	1.110223e-16
A1R	9.61	14.27	11.58	13.76	11.37	8.93	10.42	7.34	11.10	8.56	14.52	14.04	15.44	14.79	**6.40**	9.56	10.93	9.32	9.07	15.42	13.62	12.96	1.110223e-16
A2C	**7.93**	12.00	12.08	11.79	11.96	9.52	9.55	10.97	9.56	10.44	11.73	11.79	13.77	13.35	11.11	12.34	11.56	13.37	11.65	12.83	11.46	12.23	1.842536e-07
A2L	10.75	14.73	13.00	13.41	12.39	8.30	10.28	6.65	10.70	7.56	13.10	14.44	15.69	15.04	**5.72**	8.51	9.44	9.24	9.80	15.59	14.39	14.27	1.110223e-16
A2R	10.20	14.06	11.68	11.86	12.96	9.18	10.83	**7.82**	9.92	9.69	13.90	12.54	14.34	14.32	8.51	10.31	10.93	10.54	9.00	14.55	12.46	13.42	1.110223e-16
A3C	**8.20**	11.72	12.18	12.54	11.68	10.34	10.25	10.73	11.17	10.51	12.48	13.45	14.21	13.13	8.94	10.28	11.27	10.37	9.42	14.82	12.41	12.92	9.614531e-13
A3L	**9.28**	12.30	11.70	12.88	12.62	10.01	10.61	9.69	10.49	10.49	12.44	13.06	13.64	13.40	9.83	10.85	10.86	10.68	10.69	13.92	11.75	11.83	8.606779e-07
A3R	9.80	13.56	11.28	12.42	12.20	10.37	10.08	10.03	11.10	9.79	11.66	13.07	12.37	13.35	**8.45**	11.15	11.44	12.24	10.21	14.55	12.15	11.72	2.172813e-08
C1C	15.00	13.77	11.07	13.86	10.82	8.68	10.65	8.45	8.66	**7.46**	12.45	13.45	14.23	14.66	8.34	8.56	10.20	10.08	10.24	14.90	13.76	13.70	1.110223e-16
C1L	12.61	12.94	11.77	11.48	11.65	**8.76**	11.23	9.87	10.48	9.87	11.89	14.06	11.54	12.89	9.06	9.59	11.15	12.49	10.62	13.01	12.54	13.51	2.707149e-08
C1R	11.68	10.72	10.82	11.13	**9.96**	10.37	11.08	10.70	12.28	11.79	11.63	12.14	11.55	11.49	10.62	12.23	11.27	13.27	12.17	11.62	13.13	11.37	2.768665e-01
C2C	**5.21**	10.45	10.42	10.79	12.35	12.55	10.83	14.25	11.14	13.86	9.63	10.68	10.01	10.34	14.48	12.39	13.58	13.55	14.01	9.69	11.11	11.66	1.110223e-16
C2L	13.48	11.00	12.20	10.32	9.54	9.92	10.44	8.94	**8.24**	12.97	12.99	11.15	11.90	10.30	12.77	9.06	12.77	12.25	12.11	12.68	13.07	12.55	4.718448e-14
C2R	**6.76**	11.37	10.54	11.76	11.23	12.18	11.72	12.20	11.06	12.73	11.73	11.45	11.38	10.73	13.17	14.31	11.70	11.99	13.66	10.73	10.58	10.03	1.089068e-08
C3C	**7.77**	11.11	12.00	12.66	11.61	10.44	10.48	11.20	9.54	12.27	10.61	10.06	12.66	11.35	14.49	12.89	13.54	14.01	12.15	10.70	11.18	10.28	5.172333e-10
C3L	**8.89**	10.30	10.06	12.93	10.87	10.61	11.24	12.10	10.94	11.28	11.23	11.49	11.77	10.63	12.24	11.65	12.35	13.41	14.41	11.96	11.30	11.35	4.741931e-04
C3R	**9.06**	11.82	11.32	11.03	11.17	9.99	9.90	11.24	10.39	11.55	12.04	11.44	10.82	10.96	11.90	13.85	12.31	12.96	13.45	12.06	12.77	10.99	7.613893e-04
P1C	15.37	13.01	10.73	13.55	10.73	8.65	10.27	8.62	8.55	**7.86**	12.48	13.58	13.69	13.87	8.77	9.14	10.31	10.45	10.83	15.11	13.87	13.55	1.110223e-16
P1L	16.28	13.04	12.29	12.61	9.44	**8.58**	9.77	9.27	9.15	8.61	12.10	13.06	12.23	13.46	8.96	9.72	10.73	12.45	12.28	12.68	13.06	13.24	1.110223e-16
P1R	13.62	14.54	11.75	13.20	10.79	9.14	11.03	8.23	9.49	**6.63**	13.11	13.87	13.82	14.72	8.35	7.58	9.89	10.45	10.30	14.18	13.92	14.41	1.110223e-16
P2C	10.87	11.63	12.58	13.08	**9.59**	11.80	10.70	10.77	10.49	9.62	11.61	12.11	11.39	12.76	10.89	11.79	10.96	12.32	12.80	12.39	11.56	11.25	4.673348e-02
P2L	15.65	12.42	11.69	11.21	11.18	8.96	10.61	9.80	9.38	**8.56**	11.61	12.94	12.66	12.76	9.96	9.52	11.37	11.34	12.58	13.03	12.34	13.44	3.238521e-13
P2R	14.42	13.31	11.22	12.37	10.04	9.37	10.10	9.52	9.79	**8.45**	11.93	13.21	12.09	12.70	10.42	10.18	11.28	11.89	12.87	12.49	12.94	12.39	3.434444e-10
P3C	13.75	12.77	10.73	13.58	11.75	9.04	10.75	8.32	9.00	**8.28**	12.00	13.35	13.49	13.94	9.08	9.52	10.80	10.45	10.41	15.24	13.49	13.25	1.110223e-16
P3L	14.99	12.73	12.23	12.70	10.39	**9.14**	9.80	9.70	10.66	9.93	11.92	12.56	11.78	12.53	9.75	9.75	11.35	11.96	11.24	13.63	12.15	12.10	1.279662e-08
P3R	12.96	15.15	11.89	12.49	11.08	8.80	10.48	**7.79**	9.94	7.86	12.82	14.31	13.49	13.75	8.89	8.31	10.11	10.08	10.35	15.25	13.70	13.48	1.110223e-16
Mean	11.42	12.85	11.51	12.51	11.27	9.66	10.51	9.55	10.25	**9.51**	12.30	12.90	12.98	13.12	9.61	10.33	11.14	11.48	11.19	13.58	12.70	12.65	1.203099e-02

Table A.14: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *global* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations..

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	3-14-4	0-12-9	3-13-5	0-13-8	2-13-6	10-11-0	7-11-3	14-7-0	5-12-4	14-7-0	0-12-9	0-11-10	0-8-13	0-9-12	**16-5-0**	13-8-0	9-11-1	9-11-1	11-10-0	0-8-13	0-13-8	0-11-10
A1L	3-17-1	0-10-11	2-17-2	1-15-5	1-17-3	8-13-0	4-16-1	11-10-0	7-14-0	9-12-0	0-15-6	0-13-8	0-11-10	0-9-12	**14-7-0**	9-12-0	7-14-0	5-15-1	10-11-0	0-7-14	0-13-8	0-12-9
A1R	8-13-0	0-13-8	0-19-2	0-13-8	2-17-2	9-12-0	4-16-1	11-10-0	2-18-1	9-12-0	0-12-9	0-13-8	0-9-12	0-12-9	**14-7-0**	8-13-0	2-18-1	8-13-0	8-13-0	0-9-12	0-13-8	0-17-4
A2C	**9-12-0**	0-20-1	0-20-1	0-21-0	0-20-1	1-20-0	1-20-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-17-4	0-20-1	0-21-0	0-20-1	0-21-0	0-20-1	0-21-0	0-20-1	0-21-0	0-20-1
A2L	4-15-2	0-10-11	0-16-5	0-14-7	0-16-5	11-10-0	7-13-1	13-8-0	4-15-2	11-10-0	0-16-5	0-12-9	0-10-11	0-10-11	**15-6-0**	11-10-0	8-13-0	8-13-0	7-13-1	0-10-11	0-12-9	0-12-9
A2R	3-18-0	0-15-6	0-21-0	0-20-1	0-18-3	6-15-0	0-21-0	**10-11-0**	5-16-0	5-16-0	0-15-6	0-19-2	0-13-8	0-13-8	9-12-0	3-18-0	0-21-0	1-20-0	7-14-0	0-12-9	0-19-2	0-17-4
A3C	**8-13-0**	0-21-0	0-21-0	0-20-1	0-21-0	1-20-0	1-20-0	1-20-0	0-21-0	1-20-0	0-20-1	0-18-3	0-18-3	0-19-2	4-17-0	1-20-0	0-21-0	1-20-0	3-18-0	0-12-9	0-20-1	0-20-1
A3L	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-20-1	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-18-3	0-21-0	0-21-0
A3R	1-20-0	0-20-1	0-21-0	0-21-0	0-21-0	1-20-0	1-20-0	1-20-0	0-21-0	1-20-0	0-21-0	0-20-1	0-21-0	0-20-1	**4-17-0**	0-21-0	0-21-0	0-21-0	1-20-0	0-14-7	0-21-0	0-21-0
C1C	0-9-12	0-15-6	1-20-0	0-15-6	2-19-0	9-12-0	3-18-0	10-11-0	9-12-0	10-11-0	0-18-3	0-15-6	0-12-9	0-11-10	10-11-0	9-12-0	4-17-0	4-17-0	4-17-0	0-10-11	0-15-6	0-15-6
C1L	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	**5-16-0**	0-21-0	1-20-0	0-21-0	1-20-0	0-21-0	0-16-5	0-21-0	0-20-1	2-19-0	1-20-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-19-2
C1R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C2C	**21-0-0**	1-19-1	1-19-1	0-20-1	0-20-1	0-20-1	0-20-1	0-17-4	0-20-1	0-18-3	4-16-1	0-20-1	3-17-1	1-19-1	0-14-7	0-20-1	0-20-1	0-20-1	0-17-4	4-16-1	0-20-1	0-20-1
C2L	0-12-9	0-18-3	1-20-0	0-21-0	1-20-0	1-20-0	1-20-0	5-16-0	1-20-0	**9-12-0**	0-19-2	0-19-2	0-21-0	0-21-0	1-20-0	3-18-0	0-20-1	0-20-1	0-21-0	0-20-1	0-18-3	0-20-1
C2R	**16-5-0**	0-20-1	0-21-0	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-21-0	0-20-1	0-19-2	0-20-1	0-20-1	0-20-1	0-21-0	0-21-0	1-20-0
C3C	**9-12-0**	0-21-0	0-20-1	0-20-1	0-21-0	1-20-0	1-20-0	0-21-0	2-19-0	0-20-1	0-21-0	1-20-0	0-20-1	0-21-0	0-15-6	0-20-1	0-20-1	0-20-1	0-20-1	0-21-0	0-21-0	1-20-0
C3L	**3-18-0**	1-20-0	1-20-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-18-3	0-21-0	0-21-0	0-21-0
C3R	**2-19-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0
P1C	0-9-12	0-16-5	2-19-0	0-15-6	2-19-0	9-12-0	2-19-0	9-12-0	10-11-0	10-11-0	0-19-2	0-15-6	0-15-6	0-15-6	9-12-0	8-13-0	2-19-0	2-19-0	2-19-0	0-9-12	0-15-6	0-15-6
P1L	0-8-13	0-18-3	1-20-0	0-19-2	2-19-0	8-13-0	1-20-0	2-19-0	3-18-0	8-13-0	1-20-0	0-18-3	1-20-0	0-15-6	6-15-0	1-20-0	1-20-0	0-21-0	1-20-0	0-19-2	0-18-3	0-17-4
P1R	0-15-6	0-12-9	0-19-2	0-16-5	1-19-1	10-11-0	0-20-1	10-11-0	8-13-0	**13-8-0**	0-16-5	0-14-7	0-14-7	0-11-10	10-11-0	11-10-0	7-14-0	3-18-0	3-18-0	0-14-7	0-14-7	0-12-9
P2C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P2L	0-9-12	0-21-0	0-21-0	1-20-0	1-20-0	3-18-0	1-20-0	1-20-0	2-19-0	**7-14-0**	1-20-0	0-20-1	0-20-1	0-20-1	1-20-0	1-20-0	1-20-0	1-20-0	0-20-1	0-19-2	0-21-0	0-18-3
P2R	0-14-7	0-20-1	0-21-0	0-21-0	1-20-0	1-20-0	1-20-0	1-20-0	1-20-0	**7-14-0**	0-21-0	0-20-1	0-21-0	0-20-1	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-20-1	0-20-1	0-21-0
P3C	0-15-6	0-19-2	1-20-0	0-15-6	0-21-0	8-13-0	1-20-0	9-12-0	8-13-0	9-12-0	0-21-0	0-16-5	0-15-6	0-15-6	8-13-0	6-15-0	1-20-0	1-20-0	1-20-0	0-10-11	0-15-6	0-16-5
P3L	0-13-8	0-21-0	0-21-0	0-21-0	1-20-0	2-19-0	1-20-0	1-20-0	1-20-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	1-20-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
P3R	0-16-5	0-10-11	0-19-2	0-18-3	2-19-0	9-12-0	2-19-0	11-10-0	3-18-0	11-10-0	0-16-5	0-12-9	0-16-5	0-16-5	9-12-0	10-11-0	3-18-0	3-18-0	3-18-0	0-10-11	0-16-5	0-16-5
μ_{wins}	3.44	0.07	0.48	0.07	0.67	4.19	1.44	4.52	2.67	**5.04**	0.22	0.04	0.15	0.04	4.96	3.59	1.67	1.7	2.26	0.15	0	0.07
σ_{wins}	5.31	0.27	0.8	0.27	0.83	4.13	1.97	5.09	3.2	4.95	0.8	0.19	0.6	0.19	5.52	4.44	2.8	2.76	3.37	0.77	0	0.27
μ_{draws}	13.96	17.59	19.74	18.63	19.48	16.74	19.22	16.3	18	15.78	18.74	17.7	17.19	17.11	15.52	17.19	19.11	18.96	18.26	15.67	18.26	17.96
σ_{draws}	5.1	3.94	1.85	2.87	1.91	4.06	2.47	4.98	3.39	4.79	2.89	3.39	4.29	4.35	5.3	4.28	2.85	2.79	3.25	5.1	3.27	3.3
μ_{loss}	3.59	3.33	0.78	2.3	0.85	**0.07**	0.33	0.19	0.33	0.19	2.04	3.26	3.67	3.85	0.52	0.22	0.22	0.33	0.48	5.19	2.74	2.96
σ_{loss}	4.66	3.99	1.42	2.85	1.61	**0.27**	0.68	0.79	0.88	0.62	2.89	3.43	4.35	4.37	1.74	0.51	0.42	0.55	0.98	5.17	3.27	3.36

Table A.15: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *global* topology and the PT trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
p values																						
A1C	11.54	14.11	10.42	12.47	12.56	7.59	9.75	**5.03**	10.34	5.86	13.92	14.13	17.53	16.46	9.34	7.14	9.72	10.93	7.70	19.14	14.13	13.21
A1L	11.58	13.85	10.51	11.63	11.23	9.55	10.72	**6.92**	10.06	7.15	13.58	14.45	16.34	15.01	8.62	7.25	10.69	10.34	8.07	17.41	13.96	14.10
A1R	10.79	14.00	11.31	11.59	11.44	9.65	9.63	**7.17**	9.92	7.56	12.69	13.21	15.70	14.63	10.20	8.24	10.52	10.97	10.37	15.35	14.30	13.76
A2C	**8.99**	9.93	11.21	11.32	12.30	10.63	10.24	11.31	11.44	10.94	10.66	10.45	13.24	13.07	13.65	11.61	13.20	12.00	12.56	12.28	10.38	11.59
A2L	11.35	14.56	10.38	12.11	11.86	8.70	10.97	**6.01**	10.59	7.20	13.07	14.24	14.90	13.99	10.99	8.32	11.61	9.23	8.82	17.08	13.75	13.27
A2R	11.08	11.92	11.15	12.11	11.45	10.35	9.44	8.99	10.24	**8.93**	12.61	12.65	14.79	14.54	11.80	9.82	11.61	10.28	10.35	14.76	11.44	12.70
A3C	9.65	11.14	10.51	10.66	11.52	9.28	11.93	**8.48**	10.32	10.30	11.20	11.15	14.79	13.56	11.72	11.59	13.06	13.63	11.99	13.85	11.39	11.28
A3L	**9.42**	11.83	11.72	11.46	12.42	10.82	11.52	10.94	10.52	10.83	11.10	10.99	14.11	13.77	11.58	10.28	10.63	10.41	10.97	13.93	11.93	11.80
A3R	**9.75**	12.10	10.08	12.72	11.59	10.63	11.38	9.90	11.18	10.68	12.56	11.39	15.10	12.04	11.18	9.83	11.45	11.75	10.35	14.62	11.41	11.30
C1C	13.24	13.00	9.97	11.92	12.30	8.37	10.56	8.68	10.27	8.54	13.49	13.51	13.62	13.85	9.92	**6.92**	9.45	11.11	10.42	15.86	13.21	14.82
C1L	12.44	13.10	11.82	11.34	10.90	**9.48**	10.18	10.00	9.86	9.58	12.77	12.30	12.79	12.34	9.80	9.93	10.79	12.72	11.38	13.77	13.27	12.45
C1R	11.25	10.75	12.04	11.11	11.51	**9.58**	11.34	12.10	11.49	10.28	12.56	10.68	12.24	12.17	10.80	11.63	13.03	13.24	11.44	10.24	11.93	11.59
C2C	**5.49**	10.62	10.62	11.73	11.93	12.45	11.06	13.63	11.27	13.69	10.49	9.69	10.18	10.11	13.07	14.69	14.00	14.01	15.08	10.18	8.92	10.07
C2L	15.28	13.75	11.73	12.90	10.28	8.85	11.54	9.66	11.08	8.61	14.45	12.92	13.24	13.04	11.13	**7.59**	7.92	11.59	10.92	12.25	12.72	11.56
C2R	**6.58**	9.93	12.10	11.10	10.73	11.39	11.32	14.17	11.13	13.27	10.90	10.58	10.49	11.68	13.90	13.28	11.63	12.92	14.70	9.99	10.08	11.13
C3C	**7.92**	9.94	11.62	12.31	9.85	12.17	11.41	12.55	10.55	13.38	11.06	9.49	11.24	12.08	12.21	14.32	13.99	12.34	13.72	10.79	9.56	10.51
C3L	8.76	10.94	11.38	11.69	10.63	11.89	10.66	13.07	11.69	12.87	10.68	9.37	12.23	12.63	13.48	12.68	12.45	12.42	13.96	10.93	**8.65**	9.94
C3R	**9.35**	10.51	12.24	12.38	10.56	12.20	10.07	11.38	10.37	10.80	10.55	11.76	11.73	13.14	12.03	13.27	11.58	13.35	13.42	10.70	9.90	11.70
P1C	13.93	12.54	10.32	12.15	11.77	7.97	10.04	8.69	10.10	8.70	13.15	12.63	13.26	14.35	10.82	**7.32**	9.32	11.59	11.14	16.08	13.06	14.06
P1L	15.49	13.51	10.89	11.44	11.08	8.49	10.44	9.90	10.54	8.86	11.08	13.85	11.82	12.62	11.66	**7.86**	9.39	12.44	12.31	13.94	12.34	13.06
P1R	12.37	14.48	11.49	13.02	11.00	**7.45**	9.82	7.82	9.89	7.66	13.38	13.59	15.19	14.81	10.11	8.10	8.94	11.45	10.87	15.31	12.97	13.28
P2C	10.76	11.63	12.15	11.73	10.76	10.31	11.44	11.69	11.38	10.48	11.80	10.96	10.71	12.35	14.07	10.79	**10.13**	11.83	11.83	12.20	11.82	12.20
P2L	14.61	12.51	11.32	12.69	10.20	8.15	11.38	10.54	11.14	9.52	11.90	12.24	12.49	13.07	11.56	**7.25**	9.51	11.31	12.13	13.75	12.03	13.70
P2R	13.46	12.72	10.59	11.94	10.72	8.15	9.97	10.76	11.08	**8.01**	12.55	13.30	13.01	12.01	12.61	8.79	9.52	12.01	11.94	14.06	12.39	13.38
P3C	12.41	13.00	10.16	12.06	11.63	8.54	10.69	9.14	10.41	9.70	12.34	12.72	13.09	13.82	11.48	**8.18**	9.24	11.18	10.99	14.87	13.14	14.20
P3L	14.48	13.14	11.20	11.87	11.46	8.75	10.51	10.37	10.79	9.62	11.61	12.83	12.07	12.30	11.30	**8.72**	9.72	11.93	11.20	13.83	12.73	12.59
P3R	11.65	14.42	12.33	12.87	10.55	7.69	10.23	8.56	9.70	**7.52**	13.31	13.15	15.44	13.50	10.21	8.77	8.73	10.54	11.03	15.63	13.44	13.72
Mean	11.24	12.37	11.16	11.94	11.27	**9.60**	10.68	9.91	10.64	9.65	12.20	12.16	13.38	13.22	11.45	9.78	10.81	11.76	11.47	13.81	12.03	12.48

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
p values																						
A1C	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A1L	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A1R	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A2C	2.268528e-04	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A2L	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A2R	2.292611e-13	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A3C	5.859887e-10	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A3L	2.925837e-04	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
A3R	8.955383e-07	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C1C	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C1L	8.479256e-07	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C1R	1.179694e-01	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C2C	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C2L	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C2R	6.217249e-15	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C3C	2.590328e-11	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C3L	2.602115e-08	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
C3R	3.316123e-04	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P1C	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P1L	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P1R	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P2C	1.655972e-01	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P2L	3.885781e-15	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P2R	2.953193e-14	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P3C	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P3L	3.570501e-09	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
P3R	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16
Mean	1.053405e-02	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16	1.110223e-16

Table A.16: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *global* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations.

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	3-14-4	1-12-8	3-16-2	3-13-5	3-13-5	11-10-0	7-12-2	**17-4-0**	3-16-2	16-5-0	1-12-8	1-12-8	0-6-15	0-7-14	8-12-1	11-10-0	7-12-2	3-16-2	10-11-0	0-2-19	1-12-8	2-13-6
A1L	2-16-3	0-15-6	4-17-0	2-16-3	2-16-3	8-13-0	3-18-0	11-10-0	5-16-0	11-10-0	0-15-6	0-12-9	0-7-14	0-10-11	8-13-0	11-10-0	3-18-0	4-17-0	8-13-0	0-7-14	0-15-6	0-14-7
A1R	2-19-0	0-15-6	2-18-1	1-18-2	1-19-1	6-15-0	6-15-0	**11-10-0**	5-16-0	9-12-0	0-18-3	0-18-3	0-7-14	0-12-9	4-17-0	8-13-0	3-18-0	2-19-0	4-17-0	0-9-12	0-13-8	0-16-5
A2C	**4-17-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-20-1	0-20-1	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2L	1-18-2	0-13-8	3-17-1	1-18-2	1-18-2	8-13-0	2-18-1	**16-5-0**	3-17-1	12-9-0	1-15-5	0-15-6	0-12-9	0-15-6	1-19-1	8-13-0	1-18-2	7-14-0	8-13-0	6-6-15	5-16-0	0-16-5
A2R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	3-18-0	3-18-0	3-18-0	3-18-0	3-18-0	0-21-0	0-21-0	0-13-8	0-13-8	0-21-0	3-18-0	0-21-0	3-18-0	3-18-0	0-13-8	0-21-0	0-21-0
A3C	2-19-0	0-21-0	1-20-0	1-20-0	0-21-0	4-17-0	0-21-0	**5-16-0**	1-20-0	1-20-0	0-21-0	0-21-0	0-14-7	0-19-2	0-21-0	0-21-0	0-20-1	0-19-2	0-21-0	0-18-3	0-21-0	0-21-0
A3L	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
A3R	2-19-0	0-21-0	2-19-0	0-21-0	0-21-0	1-20-0	0-21-0	2-19-0	0-21-0	1-20-0	0-21-0	0-21-0	0-14-7	0-21-0	0-21-0	2-19-0	0-21-0	0-21-0	2-19-0	0-16-5	0-21-0	0-21-0
C1C	0-17-4	0-17-4	2-19-0	0-20-1	0-20-1	9-12-0	2-19-0	9-12-0	2-19-0	9-12-0	0-16-5	0-16-5	0-16-5	0-16-5	2-19-0	12-9-0	6-15-0	1-19-1	2-19-0	0-10-11	0-17-4	0-11-10
C1L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	0-21-0
C1R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C2C	**20-1-0**	2-18-1	2-18-1	0-20-1	0-20-1	0-20-1	1-19-1	0-19-2	0-20-1	0-18-3	2-18-1	5-15-1	2-18-1	2-18-1	0-19-2	0-11-10	0-18-3	0-18-3	0-10-11	2-18-1	7-14-0	2-18-1
C2L	0-12-9	0-16-5	0-20-1	0-17-4	2-19-0	7-14-0	0-21-0	3-18-0	1-20-0	8-13-0	0-15-6	0-17-4	0-17-4	0-17-4	1-20-0	**12-9-0**	9-12-0	0-20-1	1-20-0	0-18-3	0-18-3	0-20-1
C2R	**17-4-0**	2-19-0	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-17-4	0-20-1	0-20-1	0-20-1	1-19-1	1-20-0	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-15-6	2-19-0	2-19-0	0-20-1
C3C	**10-11-0**	2-19-0	0-21-0	0-20-1	2-19-0	0-20-1	0-21-0	0-20-1	0-21-0	0-20-1	0-21-0	3-18-0	0-21-0	0-20-1	0-20-1	0-16-5	0-16-5	0-20-1	0-18-3	0-21-0	3-18-0	0-21-0
C3L	4-17-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	0-19-2	0-21-0	2-19-0	0-21-0	0-21-0	0-18-3	0-20-1	0-21-0	0-21-0	0-17-4	0-21-0	**5-16-0**	1-20-0
C3R	**1-20-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0
P1C	0-16-5	0-19-2	2-19-0	0-19-2	1-19-1	10-11-0	3-18-0	7-14-0	3-18-0	7-14-0	0-17-4	0-19-2	0-17-4	0-13-8	1-20-0	**12-9-0**	4-17-0	1-19-1	1-20-0	0-9-12	0-17-4	0-14-7
P1L	0-10-11	0-17-4	1-20-0	1-20-0	1-20-0	**6-15-0**	1-20-0	2-19-0	1-20-0	5-16-0	1-20-0	0-17-4	0-21-0	0-19-2	0-21-0	9-12-0	4-17-0	0-20-1	0-20-1	0-16-5	0-20-1	0-18-3
P1R	0-17-4	0-13-8	0-20-1	0-16-5	2-19-0	**12-9-0**	4-17-0	10-11-0	4-17-0	10-11-0	0-16-5	0-16-5	0-11-10	0-12-9	4-17-0	10-11-0	9-12-0	0-20-1	3-18-0	0-11-10	0-16-5	0-16-5
P2C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P2L	0-15-6	0-19-2	0-20-1	0-19-2	1-20-0	8-13-0	0-20-1	1-20-0	0-21-0	3-18-0	0-20-1	0-19-2	0-19-2	0-19-2	0-20-1	**15-6-0**	3-18-0	0-20-1	0-20-1	0-17-4	0-20-1	0-17-4
P2R	0-18-3	0-19-2	0-21-0	0-21-0	0-21-0	9-12-0	1-20-0	0-21-0	0-21-0	**11-10-0**	0-19-2	0-18-3	0-18-3	0-20-1	0-19-2	5-16-0	1-20-0	0-20-1	0-21-0	0-16-5	0-19-2	0-18-3
P3C	0-20-1	0-19-2	2-19-0	0-21-0	0-21-0	7-14-0	1-20-0	4-17-0	1-20-0	3-18-0	0-20-1	0-19-2	0-19-2	0-16-5	0-21-0	**9-12-0**	3-18-0	0-21-0	0-21-0	0-13-8	0-18-3	0-15-6
P3L	0-16-5	0-19-2	0-21-0	0-21-0	0-21-0	4-17-0	0-21-0	1-20-0	0-21-0	2-19-0	0-21-0	0-19-2	0-21-0	0-21-0	0-21-0	**5-16-0**	2-19-0	0-21-0	0-21-0	0-17-4	0-20-1	0-21-0
P3R	1-18-2	0-13-8	0-19-2	0-16-5	2-19-0	11-10-0	3-18-0	9-12-0	4-17-0	11-10-0	0-16-5	0-16-5	0-10-11	0-16-5	3-18-0	9-12-0	9-12-0	2-19-0	2-19-0	9-9-12	0-16-5	0-15-6
μ_{wins}	2.67	0.26	0.89	0.33	0.67	4.63	1.37	4.11	1.33	4.56	0.19	0.44	0.11	0.07	1.19	**5.22**	2.37	0.85	1.63	0.15	0.67	0.19
σ_{wins}	5.06	0.66	1.22	0.73	0.92	4.27	1.94	5.23	1.73	4.95	0.48	1.15	0.42	**0.38**	2.3	5.18	3.1	1.68	2.8	0.53	1.71	0.56
μ_{draws}	16.15	18.22	19.7	19.41	19.78	16.26	19.41	16.56	19.48	16.19	18.85	18.26	16.52	17.37	19.33	15.15	18.07	19.56	18.37	15.19	18.22	18.19
σ_{draws}	4.92	2.98	1.46	2.14	1.83	4.16	2.17	5.03	1.85	4.75	2.64	2.7	4.9	3.94	2.34	4.94	3.09	1.74	3.25	5.51	2.79	3.11
μ_{loss}	2.19	2.52	0.41	1.26	0.56	**0.11**	0.22	0.33	0.19	0.26	1.96	2.3	4.37	3.56	0.48	0.63	0.56	0.59	1	5.67	2.11	2.63
σ_{loss}	3	3	0.64	1.72	1.15	**0.32**	0.51	0.92	0.48	0.71	2.52	2.64	4.97	3.97	0.8	2.11	1.19	0.8	2.47	5.62	2.68	3.05

Table A.17: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *global* topology and the ST trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm	p values
A1C	13.04	13.89	10.42	10.85	10.68	10.61	11.91	11.01	10.82	**8.92**	11.97	12.13	12.01	12.39	14.07	11.17	10.20	10.75	9.86	11.82	12.56	11.92	3.553294e-05
A1L	13.96	13.04	11.49	12.39	11.37	9.52	12.56	10.25	11.35	**8.72**	12.28	12.35	11.88	12.09	12.77	9.58	11.58	11.77	9.39	11.00	12.61	11.04	4.451015e-06
A1R	12.94	12.28	11.06	11.28	11.23	10.73	11.49	10.34	11.20	10.79	12.73	11.72	11.66	11.63	12.94	**10.25**	11.23	11.20	10.42	12.01	12.75	11.11	3.057280e-01
A2C	**9.13**	11.19	10.00	12.39	12.16	11.87	10.03	12.10	11.20	10.93	11.01	11.22	12.73	12.15	12.48	11.72	11.96	12.21	12.01	11.11	11.39	12.01	1.241195e-01
A2L	12.42	12.60	10.35	11.62	11.97	9.82	11.65	10.25	11.95	**9.31**	10.96	12.45	11.87	12.05	12.89	9.97	10.41	11.79	11.93	13.42	11.73	11.59	5.806087e-03
A2R	11.31	10.32	12.39	13.24	10.15	10.18	10.67	10.63	11.16	10.68	13.14	11.49	14.32	12.43	12.59	10.24	12.30	**9.68**	12.21	11.21	11.36	11.30	2.769600e-04
A3C	10.76	12.59	10.87	11.35	11.84	10.79	11.20	11.77	10.96	**10.45**	10.99	10.85	11.99	10.87	12.13	11.80	11.83	12.90	12.10	11.62	12.00	11.34	8.293063e-01
A3L	11.20	12.12	10.94	12.38	12.07	11.27	12.15	11.13	13.03	11.48	10.54	11.46	11.80	12.53	13.11	9.87	9.89	**9.29**	10.80	11.59	12.46	11.89	2.275349e-02
A3R	11.65	13.67	10.37	12.14	12.30	10.34	12.13	10.42	10.97	10.56	12.61	12.30	12.48	13.11	11.42	**9.77**	11.13	10.20	11.30	12.38	11.54	10.23	6.841995e-03
C1C	14.23	12.70	11.32	11.27	11.30	10.69	11.94	8.76	11.87	8.86	11.56	11.82	12.96	13.11	12.77	**8.24**	8.69	11.24	11.80	14.10	11.66	12.10	6.518341e-12
C1L	12.93	12.18	12.28	12.17	10.83	11.14	11.56	9.92	10.13	10.08	10.66	12.39	12.30	12.14	13.31	**9.86**	10.31	12.56	10.55	12.59	12.28	10.82	4.209421e-03
C1R	11.44	10.56	13.07	13.21	10.58	11.73	12.04	12.34	11.13	10.08	10.48	11.99	12.30	12.37	12.77	10.28	10.32	12.27	12.24	10.35	11.48	**9.97**	1.406981e-02
C2C	**5.63**	10.65	12.18	11.93	11.83	12.07	12.58	14.23	11.85	11.42	11.86	10.41	11.25	10.23	13.41	13.32	12.63	12.14	13.30	8.63	10.70	10.75	5.551115e-16
C2L	15.48	12.00	11.48	13.00	10.24	10.23	10.63	10.17	11.34	11.17	11.96	11.52	13.41	13.28	13.68	8.86	**8.30**	11.25	10.63	10.18	12.48	11.72	6.499690e-12
C2R	**6.92**	10.97	11.75	12.24	10.24	12.58	12.03	14.41	12.08	11.83	10.48	11.38	10.31	11.79	12.92	11.73	11.75	12.87	13.39	10.17	10.34	10.83	4.224842e-09
C3C	**7.73**	8.38	12.34	11.54	11.83	12.55	11.38	13.03	12.03	12.45	12.37	10.63	10.97	11.07	12.83	13.15	11.75	13.85	12.41	10.17	10.11	10.44	5.948355e-09
C3L	**7.80**	9.49	13.38	13.46	11.30	12.35	10.63	10.97	11.82	11.44	11.54	9.31	13.82	12.13	14.15	13.18	10.69	11.62	11.99	10.75	10.89	10.30	4.716555e-10
C3R	**8.97**	10.80	12.96	12.49	10.32	11.87	10.97	10.54	11.27	12.34	11.38	10.51	11.58	12.18	12.59	13.46	12.32	12.35	13.69	11.14	9.32	9.93	3.454058e-05
P1C	14.39	12.14	11.95	11.53	11.20	10.69	11.93	9.27	11.37	8.99	12.14	12.25	12.57	12.70	13.13	**8.54**	8.90	11.17	11.54	13.73	11.32	11.56	2.879410e-09
P1L	15.59	11.21	10.75	12.06	10.97	10.17	11.42	9.55	11.83	9.85	11.28	12.76	13.31	11.86	13.07	**9.07**	9.66	11.06	11.80	11.45	12.79	11.49	2.161850e-08
P1R	13.68	12.07	11.04	12.40	11.82	9.54	9.31	9.17	10.08	10.10	11.69	12.99	15.61	14.75	12.65	**7.89**	8.87	10.25	10.00	13.85	12.76	12.49	1.110223e-16
P2C	10.54	**9.24**	11.88	12.89	11.82	12.14	11.15	12.38	11.89	9.90	12.10	11.30	11.77	11.85	14.23	10.20	10.38	12.24	13.20	10.21	10.51	11.20	3.742082e-04
P2L	14.70	11.51	11.30	11.75	11.28	10.24	11.06	11.00	11.75	10.49	11.04	11.52	12.28	12.65	14.08	**9.07**	8.54	11.93	12.15	11.17	11.34	12.15	1.971988e-06
P2R	13.51	12.69	10.63	12.37	10.08	10.99	11.35	10.23	10.99	**8.94**	11.97	11.56	12.11	11.24	13.38	9.31	10.08	10.96	12.76	12.08	12.11	13.65	8.416760e-06
P3C	13.07	11.75	11.09	11.20	11.58	11.38	12.30	9.39	11.86	9.83	12.73	10.51	11.87	12.47	13.27	**8.54**	10.10	11.07	11.93	13.80	11.14	12.13	2.013685e-05
P3L	14.45	11.69	11.08	11.70	11.56	10.41	11.17	10.76	12.62	10.97	10.77	11.85	11.96	11.42	13.75	9.58	**9.56**	10.97	10.62	12.13	12.48	11.49	7.528087e-04
P3R	12.96	12.55	12.18	11.94	12.01	9.42	9.61	9.41	10.79	10.15	11.45	11.61	14.92	14.19	11.69	**8.25**	9.06	10.10	10.30	14.55	12.94	12.93	1.110223e-16
Mean	11.87	11.64	11.50	12.10	11.28	10.94	11.37	10.87	11.46	10.40	11.62	11.57	12.44	12.25	13.04	**10.26**	10.46	11.47	11.64	11.75	11.67	11.42	4.867940e-02

Table A.18: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *global* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations.

Env	GFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	0-20-1	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	0-21-0	0-21-0	2-19-0	0-21-0	0-21-0	0-21-0
A1L	0-17-4	0-20-1	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	1-20-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0
A1R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	**1-20-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
A2R	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-17-4	0-21-0	0-21-0	1-20-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0
A3C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C1C	0-17-4	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	5-16-0	0-21-0	4-17-0	0-21-0	0-21-0	0-17-4	0-17-4	0-18-3	6-15-0	6-15-0	0-21-0	0-21-0	0-17-4	0-21-0	0-21-0
C1L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C1R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C2C	**20-1-0**	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-18-3	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	1-19-1	0-19-2	0-19-2	0-19-2	0-20-1	0-19-2	5-16-0	0-20-1	0-20-1
C2L	0-10-11	0-21-0	0-21-0	0-19-2	1-20-0	1-20-0	1-20-0	1-20-0	1-20-0	1-20-0	0-21-0	0-21-0	0-19-2	0-19-2	0-19-2	5-16-0	**6-15-0**	1-20-0	1-20-0	1-20-0	0-20-1	0-21-0
C2R	**15-6-0**	0-20-1	0-20-1	0-20-1	1-20-0	0-20-1	0-20-1	0-16-5	0-20-1	0-20-1	0-21-0	0-20-1	1-20-0	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	0-20-1	1-20-0	1-20-0	0-21-0
C3C	**12-9-0**	7-14-0	0-20-1	0-21-0	0-20-1	0-19-2	0-21-0	0-19-2	0-20-1	0-19-2	0-20-1	0-21-0	0-21-0	0-21-0	0-19-2	0-19-2	0-20-1	0-19-2	0-19-2	0-21-0	0-21-0	0-21-0
C3L	**9-12-0**	2-19-0	0-19-2	0-19-2	0-21-0	0-20-1	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	4-17-0	0-18-3	0-20-1	0-18-3	0-20-1	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0
C3R	**2-19-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	0-21-0	0-19-2	0-21-0	0-19-0	0-21-0
P1C	0-17-4	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	2-19-0	0-21-0	3-18-0	0-21-0	0-21-0	0-20-1	0-20-1	0-18-3	**5-16-0**	3-18-0	0-21-0	0-21-0	0-17-4	0-21-0	0-21-0
P1L	0-8-13	1-20-0	1-20-0	0-21-0	1-20-0	1-20-0	1-20-0	1-20-0	0-21-0	1-20-0	1-20-0	0-21-0	0-20-1	0-21-0	0-21-0	**2-19-0**	1-20-0	1-20-0	0-21-0	1-20-0	0-21-0	1-20-0
P1R	0-16-5	0-20-1	1-20-0	0-20-1	0-21-0	4-17-0	4-17-0	4-17-0	2-19-0	2-19-0	0-21-0	0-19-2	0-11-10	0-12-9	0-20-1	**10-11-0**	5-16-0	2-19-0	2-19-0	0-16-5	0-20-1	0-20-1
P2C	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-17-4	1-20-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0
P2L	0-17-4	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-20-1	0-19-2	2-19-0	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P2R	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	3-18-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	3-18-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2
P3C	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-20-1	**4-17-0**	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	0-21-0
P3L	0-18-3	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	2-19-0	2-19-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P3R	0-20-1	0-20-1	0-21-0	0-21-0	0-21-0	3-18-0	3-18-0	3-18-0	1-20-0	3-18-0	0-21-0	0-21-0	0-12-9	0-14-7	0-21-0	**7-14-0**	3-18-0	3-18-0	2-19-0	0-13-8	0-20-1	0-20-1
μ_{wins}	**2.15**	0.41	0.07	0	0.15	0.48	0.33	0.63	0.15	0.96	0.04	0.15	0.04	0.04	0	1.81	1.07	0.3	0.3	0.33	0.11	0.04
σ_{wins}	5.3	1.39	0.27	0	0.36	0.98	0.96	1.33	0.46	1.32	0.19	0.77	0.19	0.19	0	2.68	1.94	0.72	0.67	1	0.42	0.19
μ_{draws}	16.89	20.26	20.74	20.74	20.78	20.33	20.59	20	20.7	19.89	20.85	20.7	19.67	19.96	19.85	18.89	19.78	20.56	20.41	19.78	20.74	20.78
σ_{draws}	5.52	1.43	0.53	0.59	0.42	1	0.97	1.62	0.54	1.28	0.36	0.87	2.65	2.23	1.23	2.55	1.91	0.8	0.84	2.1	0.53	0.51
μ_{loss}	1.96	0.33	0.19	0.26	0.07	0.19	0.07	0.37	0.15	0.15	0.11	0.15	1.3	1	1.15	0.3	0.15	0.15	0.3	0.89	0.15	0.19
σ_{loss}	3.35	0.62	0.48	0.59	0.27	0.48	0.27	1.15	0.36	0.46	0.32	0.46	2.66	2.22	1.23	0.67	0.46	0.46	0.67	2.01	0.36	0.48

Table A.19: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *island* topology and the RT trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	15.18	12.46	13.63	14.63	8.58	9.14	7.55	8.30	8.11	8.92	12.17	11.46	14.77	13.87	13.11	13.61	**7.68**	11.30	11.52	14.04	11.01	11.94
A1L	14.61	12.07	13.54	13.48	9.45	8.76	8.30	8.75	9.08	9.17	10.96	11.31	14.41	14.07	12.83	11.76	**7.93**	12.30	12.97	13.86	11.56	11.85
A1R	14.23	11.38	12.94	13.97	9.63	9.80	**8.27**	9.37	9.51	9.55	10.30	10.96	14.76	14.94	11.48	11.92	9.34	10.24	12.86	15.77	10.83	10.96
A2C	11.41	12.45	13.21	13.99	12.31	11.82	11.89	10.15	10.79	9.79	12.30	11.24	13.07	13.48	10.14	10.56	**8.42**	10.15	10.83	12.58	11.35	11.07
A2L	13.56	12.28	13.03	14.25	9.59	10.63	8.80	**8.31**	8.96	9.38	11.00	11.75	14.46	14.83	11.80	11.90	8.68	11.17	10.70	14.24	12.08	11.58
A2R	11.87	13.07	13.66	12.85	11.25	9.97	10.83	10.00	11.28	9.58	11.48	12.87	14.15	12.99	11.13	10.28	**8.39**	8.26	10.63	13.04	13.07	12.34
A3C	10.18	12.96	12.29	12.05	11.18	11.37	10.45	9.96	10.66	**8.61**	12.68	12.41	13.00	12.92	12.27	11.38	9.54	11.34	9.87	12.72	12.23	12.96
A3L	12.65	11.22	12.46	12.76	11.85	11.80	10.89	9.56	10.07	9.79	11.90	11.60	13.49	13.46	11.82	12.06	**8.79**	10.61	10.28	13.11	11.87	10.96
A3R	11.44	10.93	13.28	13.23	10.90	9.80	10.28	10.10	10.39	10.13	12.30	11.18	13.45	14.44	11.79	10.83	**8.80**	10.70	11.82	13.14	12.08	11.99
C1C	15.58	12.85	12.75	12.41	10.54	8.42	10.83	8.77	10.25	8.87	10.82	10.76	12.77	14.06	14.41	13.21	**8.13**	10.38	12.58	10.51	12.03	12.08
C1L	16.15	12.58	11.10	12.90	11.00	**8.18**	9.62	8.31	9.46	8.82	13.24	12.77	10.89	11.80	12.07	12.11	8.37	11.58	14.61	11.08	13.17	13.18
C1R	15.15	12.23	13.41	12.92	8.97	**7.10**	10.25	8.86	9.83	10.17	10.77	12.35	11.52	12.17	12.08	13.01	10.52	12.46	13.30	11.97	12.44	11.51
C2C	8.94	14.04	11.72	14.37	12.59	12.34	11.08	9.27	12.15	**7.10**	13.66	12.55	12.58	14.66	12.17	9.92	8.23	8.83	8.37	13.00	12.61	12.83
C2L	16.38	12.08	12.31	12.85	10.83	**8.72**	9.80	9.46	10.04	9.85	11.56	11.90	12.72	11.46	14.66	12.68	8.75	11.56	13.13	9.75	11.41	11.10
C2R	10.97	11.97	12.82	13.58	12.56	11.89	13.37	9.27	10.80	8.89	11.97	11.73	12.38	13.79	13.04	9.04	**8.13**	8.63	10.23	12.21	12.69	13.04
C3C	10.39	12.94	12.31	12.15	12.13	11.31	12.34	**8.97**	11.21	9.06	12.21	12.35	11.56	12.85	12.94	10.06	9.73	10.62	9.87	11.83	12.80	13.35
C3L	12.83	12.15	12.80	11.89	11.77	11.13	11.80	9.69	11.11	**7.90**	11.48	13.20	11.44	12.63	12.24	9.99	9.28	12.07	10.87	13.04	12.87	10.80
C3R	13.68	12.61	13.82	13.25	11.27	9.89	12.17	10.13	10.35	**9.17**	11.27	10.94	10.96	11.35	14.24	10.28	9.93	10.69	11.62	12.28	11.62	11.49
P1C	14.97	12.21	11.66	12.08	10.89	8.77	10.46	8.73	10.61	9.14	11.06	11.18	12.38	13.75	15.00	12.86	**8.45**	10.37	12.87	10.45	12.77	12.32
P1L	16.10	12.37	13.08	13.16	10.99	8.86	11.28	8.32	10.39	9.24	10.30	11.66	12.62	11.54	14.75	13.11	**8.14**	10.51	12.56	9.96	11.56	12.51
P1R	15.85	12.58	13.08	13.00	10.21	**8.03**	11.08	9.20	10.10	8.63	11.18	11.20	13.73	11.86	13.75	12.54	8.72	10.42	12.23	11.03	11.89	12.70
P2C	13.44	11.75	12.37	12.45	11.80	10.69	10.75	9.48	11.83	8.56	12.45	12.93	13.34	12.63	13.08	10.48	**7.90**	9.70	9.82	11.93	13.23	12.39
P2L	15.61	12.77	12.07	13.17	11.01	8.63	10.49	9.58	10.99	9.41	10.49	12.13	11.52	12.37	13.56	12.13	**8.11**	11.42	13.01	10.55	12.01	11.96
P2R	15.39	11.89	11.83	12.32	11.52	9.30	10.56	8.73	11.41	10.00	11.32	11.37	12.25	12.01	14.41	12.10	**7.42**	10.44	13.27	11.10	11.66	12.69
P3C	15.61	12.55	12.08	12.15	11.00	8.92	10.75	9.08	11.14	8.93	11.39	11.68	12.07	13.70	13.86	12.13	**8.07**	10.24	12.66	10.34	12.44	12.21
P3L	15.96	12.10	12.15	12.63	11.18	9.15	10.48	9.25	9.96	**8.93**	11.34	10.85	11.52	11.89	14.49	12.68	9.25	9.86	12.73	11.18	12.66	12.75
P3R	16.92	12.41	12.17	12.94	11.44	8.18	10.44	9.66	9.82	**7.93**	11.82	10.46	12.87	12.80	14.15	12.55	8.06	10.23	11.61	11.42	12.68	12.45
Mean	13.89	12.33	12.65	13.02	10.98	9.73	10.55	9.23	10.38	9.09	11.61	11.73	12.77	13.05	13.01	11.67	**8.62**	10.60	11.73	12.08	12.17	12.11
p values	1.110223e-16	1.110223e-16	1.110223e-16	1.148477e-06	1.110223e-16	4.810263e-12	1.276231e-05	2.178043e-05	2.844347e-07	1.110223e-16	1.110223e-16	1.110223e-16	7.771561e-16	1.110223e-16	1.110223e-16	3.530509e-14	4.308362e-06	2.084742e-06	8.372642e-07	1.110223e-16	1.110223e-16	1.110223e-16	1.600225e-06

Table A.20: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *island* topology and the RT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	0-13-8	0-17-4	0-14-7	0-14-7	8-13-0	8-13-0	**12-9-0**	9-12-0	10-11-0	8-13-0	0-18-3	0-21-0	0-14-7	0-14-7	0-14-7	0-14-7	11-10-0	0-21-0	0-20-1	0-14-7	1-20-0	0-19-2
A1L	0-14-7	0-20-1	0-14-7	0-14-7	6-15-0	8-13-0	9-12-0	8-13-0	6-15-0	6-15-0	0-21-0	0-21-0	0-14-7	0-14-7	0-17-4	0-21-0	**10-11-0**	0-19-2	0-17-4	0-14-7	0-21-0	0-21-0
A1R	0-13-8	1-20-0	0-20-1	0-14-7	5-16-0	5-16-0	**7-14-0**	5-16-0	5-16-0	5-16-0	3-18-0	2-19-0	0-12-9	0-9-12	1-20-0	0-21-0	5-16-0	4-17-0	0-20-1	0-7-14	2-19-0	2-19-0
A2C	0-21-0	0-20-1	0-20-1	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	**6-15-0**	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
A2L	0-16-5	0-21-0	0-17-4	0-15-6	4-17-0	1-20-0	6-15-0	6-15-0	6-15-0	5-16-0	0-21-0	0-21-0	0-15-6	0-13-8	0-21-0	0-21-0	6-15-0	0-21-0	1-20-0	0-15-6	0-21-0	0-21-0
A2R	0-21-0	0-19-2	0-18-3	0-19-2	0-21-0	1-20-0	0-21-0	1-20-0	0-21-0	2-19-0	0-21-0	0-19-2	0-16-5	0-19-2	0-21-0	0-21-0	8-13-0	**9-12-0**	0-21-0	0-19-2	0-19-2	0-20-1
A3C	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	**6-15-0**	0-20-1	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-20-1
A3L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	**3-18-0**	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
A3R	0-21-0	0-21-0	0-20-1	0-20-1	0-21-0	1-20-0	1-20-0	1-20-0	1-20-0	1-20-0	0-21-0	0-21-0	0-20-1	0-15-6	0-21-0	0-21-0	**5-16-0**	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0
C1C	0-10-11	0-18-3	0-19-2	0-19-2	1-20-0	9-12-0	1-20-0	6-15-0	2-19-0	4-17-0	1-20-0	1-20-0	0-18-3	0-17-4	0-15-6	0-17-4	9-12-0	2-19-0	0-19-2	1-20-0	0-21-0	0-21-0
C1L	0-7-14	0-18-3	1-20-0	0-17-4	1-20-0	8-13-0	2-19-0	8-13-0	2-19-0	7-14-0	0-17-4	0-17-4	1-20-0	1-20-0	1-20-0	1-20-0	8-13-0	1-20-0	0-15-6	1-20-0	0-17-4	0-17-4
C1R	0-13-8	0-20-1	0-18-3	0-19-2	4-17-0	**14-7-0**	1-20-0	5-16-0	1-20-0	1-20-0	1-20-0	0-20-0	0-20-1	0-20-1	0-20-1	0-18-3	1-20-0	0-20-1	0-18-3	0-20-1	0-20-1	0-20-1
C2C	5-16-0	0-14-7	0-20-1	0-14-7	0-18-3	0-18-3	0-20-1	4-17-0	0-20-1	**15-6-0**	0-15-6	0-18-3	0-18-3	0-14-7	0-20-1	3-18-0	11-10-0	6-15-0	11-10-0	0-16-5	0-18-3	0-17-4
C2L	0-5-16	1-20-0	1-20-0	0-19-2	1-20-0	**5-16-0**	2-19-0	2-19-0	2-19-0	2-19-0	1-20-0	1-20-0	0-20-1	1-20-0	0-14-7	0-21-0	4-17-0	1-20-0	0-19-2	2-19-0	1-20-0	1-20-0
C2R	0-21-0	0-21-0	0-19-2	0-16-5	0-20-1	0-21-0	0-16-5	3-18-0	0-21-0	5-16-0	0-21-0	0-21-0	0-20-1	0-16-5	0-17-4	5-16-0	**10-11-0**	7-14-0	0-21-0	0-20-1	0-19-2	0-17-4
C3C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2
C3L	0-20-1	0-20-1	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	**9-12-0**	0-21-0	0-20-1	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	0-20-1	0-21-0	0-20-1	0-20-1	0-21-0
C3R	0-20-1	0-21-0	0-20-1	0-20-1	0-21-0	1-20-0	0-21-0	1-20-0	0-21-0	**4-17-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-17-4	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P1C	0-12-9	0-21-0	0-21-0	0-21-0	2-19-0	6-15-0	2-19-0	6-15-0	2-19-0	3-18-0	0-21-0	0-21-0	0-21-0	0-17-4	0-12-9	0-18-3	6-15-0	2-19-0	0-18-3	2-19-0	0-18-3	0-21-0
P1L	0-8-13	0-19-2	0-18-3	0-18-3	1-20-0	5-16-0	1-20-0	9-12-0	2-19-0	2-19-0	2-19-0	1-20-0	0-19-2	1-20-0	0-13-8	0-18-3	9-12-0	2-19-0	0-19-2	2-19-0	1-20-0	0-19-2
P1R	0-9-12	0-20-1	0-18-3	0-18-3	1-20-0	**9-12-0**	1-20-0	3-18-0	1-20-0	6-15-0	1-20-0	1-20-0	0-17-4	1-20-0	0-17-4	0-20-1	6-15-0	1-20-0	0-20-1	1-20-0	0-21-0	0-18-3
P2C	0-19-2	0-21-0	0-20-1	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	6-15-0	0-20-1	0-19-2	0-19-2	0-19-2	0-19-2	0-21-0	**11-10-0**	0-21-0	0-21-0	0-20-1	0-19-2	0-20-1
P2L	0-10-11	0-19-2	0-21-0	0-19-2	1-20-0	5-16-0	1-20-0	2-19-0	1-20-0	2-19-0	1-20-0	0-20-1	1-20-0	0-20-1	0-17-4	0-20-1	**8-13-0**	1-20-0	0-19-2	1-20-0	0-21-0	0-21-0
P2R	0-11-10	0-20-1	0-20-1	0-20-1	2-19-0	1-20-0	1-20-0	3-18-0	1-19-1	2-19-0	1-20-0	1-20-0	0-20-1	0-20-1	0-17-4	0-20-1	**13-8-0**	1-20-0	0-19-2	1-20-0	0-20-1	0-20-1
P3C	0-11-10	0-20-1	0-20-1	0-20-1	1-20-0	3-18-0	1-20-0	3-18-0	1-20-0	3-18-0	1-20-0	0-21-0	0-20-1	0-17-4	0-17-4	0-20-1	**11-10-0**	1-20-0	0-20-1	1-20-0	0-20-1	0-20-1
P3L	0-8-13	0-21-0	0-21-0	0-21-0	1-20-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	1-20-0	1-20-0	1-20-0	1-20-0	0-14-7	0-21-0	2-19-0	2-19-0	0-21-0	1-20-0	0-21-0	0-21-0
P3R	0-1-20	1-17-3	1-17-3	1-17-3	1-20-0	10-11-0	1-20-0	2-19-0	2-19-0	10-11-0	1-20-0	1-20-0	1-17-3	1-17-3	0-16-5	1-17-3	10-11-0	1-20-0	1-20-0	1-20-0	1-17-3	1-17-3
μ_{wins}	0.19	0.11	0.11	0.04	1.41	3.81	1.89	3.33	1.74	4.37	0.52	0.33	0.15	0.22	0.07	0.37	**6.44**	1.52	0.48	0.52	0.22	0.15
σ_{wins}	0.96	0.32	0.32	**0.19**	2.12	3.99	3.03	2.91	2.4	3.4	0.75	0.55	0.36	0.42	0.27	1.11	3.97	2.34	2.12	0.7	0.51	0.46
μ_{draws}	14.19	19.63	19.19	18.41	19.41	17.07	18.89	17.67	19.19	16.63	19.93	20.15	18.63	17.89	17.89	19.63	14.56	19.33	19.41	18.67	19.93	19.74
σ_{draws}	5.88	1.64	1.92	2.44	2.08	3.92	3.04	2.91	2.37	3.4	1.44	1.03	2.47	3.03	2.86	1.92	3.97	2.29	2.39	3.04	1.27	1.43
μ_{loss}	6.63	1.26	1.7	2.56	0.19	0.11	0.22	0	0.07	0	0.56	0.52	2.22	2.89	3.04	1	0	0.15	1.11	1.81	0.85	1.11
σ_{loss}	6.02	1.63	1.94	2.42	0.62	0.58	0.97	0	0.27	0	1.45	1.05	2.53	3.18	2.93	1.73	0	0.46	1.53	3.27	1.23	1.4

Table A.21: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *island* topology and the PT trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
p values																						1.110223e-16
A1C	14.20	11.68	12.71	14.96	9.35	9.23	9.90	8.85	10.11	9.99	12.04	11.83	13.92	14.17	12.52	10.69	**8.48**	11.75	10.61	12.92	11.21	11.90
A1L	14.87	12.90	11.66	13.55	10.61	8.73	10.48	**8.13**	10.63	10.32	12.15	11.59	14.14	13.31	12.80	10.96	8.58	11.97	10.21	12.99	10.79	11.62
A1R	14.38	11.23	13.60	12.68	9.77	**7.11**	8.35	8.24	9.97	10.79	10.73	10.76	14.19	14.51	12.15	12.45	10.49	14.03	12.61	12.25	10.96	11.73
A2C	12.14	12.13	11.83	12.17	12.69	**9.76**	11.68	10.59	12.38	10.59	12.69	12.31	11.96	11.62	12.77	9.94	10.18	9.80	11.38	12.59	10.37	11.42
A2L	13.01	12.62	12.83	13.61	11.08	10.62	11.01	10.65	10.49	9.70	11.27	12.34	12.83	14.41	10.70	**9.56**	8.55	10.79	11.30	11.56	12.08	11.97
A2R	11.73	12.92	12.58	12.54	12.62	11.18	11.85	11.04	10.80	10.58	10.65	13.07	12.97	12.75	11.27	**8.97**	9.31	10.52	11.20	11.10	11.68	11.69
A3C	10.61	11.44	12.58	12.81	11.11	12.08	11.11	**9.62**	11.32	10.01	12.70	11.75	13.21	12.70	11.56	11.24	9.85	10.56	10.17	11.83	13.62	11.10
A3L	13.08	10.10	11.81	12.35	11.87	11.34	11.04	11.75	12.07	**9.76**	11.27	11.04	12.56	13.44	10.96	11.86	11.20	11.27	10.00	11.61	10.44	12.18
A3R	11.83	11.46	13.44	12.98	11.14	10.44	11.17	9.48	11.46	**9.34**	10.51	10.31	13.56	13.26	11.99	11.58	10.61	12.80	11.65	12.80	9.80	11.39
C1C	15.44	10.20	12.21	13.73	11.07	9.42	11.01	8.89	11.39	9.77	11.68	10.99	11.89	12.62	15.58	13.23	**7.37**	10.80	12.31	10.79	11.18	11.44
C1L	15.79	12.13	12.66	12.93	10.23	8.83	11.14	**8.54**	10.08	9.01	12.04	11.87	12.45	13.35	12.66	12.01	9.62	11.52	12.80	9.85	11.61	11.87
C1R	14.58	11.45	11.69	12.83	10.51	**8.00**	9.46	9.80	9.97	9.73	11.55	11.63	10.52	12.23	13.66	13.01	9.80	12.41	14.66	10.86	11.51	13.13
C2C	8.75	12.69	13.73	13.55	11.46	12.80	12.73	10.44	12.06	9.76	13.61	12.87	13.10	13.27	11.14	8.54	7.92	8.04	**6.14**	12.79	14.01	13.61
C2L	16.32	12.27	11.72	11.20	11.18	9.82	10.46	10.21	10.93	10.79	11.27	11.35	10.85	11.96	15.34	12.34	**9.03**	12.30	11.34	8.69	11.17	12.48
C2R	11.04	12.03	13.80	12.85	11.76	11.11	12.24	11.70	11.01	9.44	12.17	14.11	13.63	12.70	10.82	9.69	8.85	8.66	**8.20**	11.65	13.54	12.00
C3C	9.96	11.46	13.62	13.68	12.28	12.37	11.97	12.01	12.35	9.37	12.23	11.11	12.59	13.30	12.54	9.99	**9.18**	10.04	10.03	10.27	10.69	11.97
C3L	12.61	11.11	12.83	13.04	11.86	11.61	12.69	10.20	10.97	9.39	11.70	11.01	12.90	12.80	13.07	11.38	**8.99**	10.99	10.69	10.72	10.61	11.83
C3R	13.86	12.04	12.04	13.31	10.44	9.75	10.20	**8.41**	9.13	9.48	11.07	11.80	13.18	12.39	13.76	13.10	10.73	10.82	12.73	12.24	10.01	12.51
P1C	15.07	10.45	12.61	13.90	11.15	10.24	11.55	8.66	11.10	9.82	11.96	10.39	11.77	12.96	15.24	13.03	**7.24**	10.77	12.92	9.90	10.66	11.61
P1L	15.49	11.45	11.68	12.13	11.51	10.83	12.15	8.51	11.35	10.72	10.27	12.00	11.65	12.54	15.08	12.56	**7.07**	11.27	12.66	9.96	11.17	10.96
P1R	15.61	11.41	11.54	13.44	10.32	10.37	10.85	8.65	10.82	9.39	11.07	10.77	11.65	13.12	14.25	12.07	**7.06**	12.66	13.96	11.30	11.42	11.28
P2C	13.48	11.83	13.23	13.79	12.04	9.90	11.42	9.15	12.14	8.45	12.61	12.15	12.99	13.79	13.20	10.90	**8.00**	9.77	10.35	10.20	11.61	12.00
P2L	15.62	10.87	13.08	13.90	10.75	11.58	11.30	9.56	10.66	10.25	11.58	11.65	11.86	11.82	14.42	12.73	**7.80**	10.68	11.52	9.01	10.93	11.42
P2R	15.10	10.46	12.58	13.03	10.38	10.11	11.31	8.63	10.51	9.56	12.10	12.48	12.00	13.10	15.17	13.03	**7.94**	10.01	11.68	9.86	12.11	11.85
P3C	15.35	10.04	12.34	13.03	11.27	9.86	11.59	8.72	11.30	10.14	11.24	10.75	12.33	13.74	14.51	13.25	**7.59**	9.55	12.42	10.10	11.76	12.13
P3L	15.66	10.83	10.80	12.02	11.30	11.32	12.31	9.86	11.46	10.42	10.94	11.03	11.75	10.75	15.20	13.20	**8.01**	11.00	12.45	10.15	12.01	10.52
P3R	16.42	10.61	10.52	12.00	11.14	9.17	11.92	9.70	11.34	8.90	12.21	11.03	11.24	12.23	15.17	12.13	**7.68**	11.52	12.86	10.90	12.06	12.27
Mean	13.78	11.47	12.43	13.04	11.14	10.28	11.22	9.63	11.03	9.83	11.68	11.63	12.51	12.92	13.24	11.61	**8.78**	10.97	11.44	11.07	11.44	11.85
p values																						3.263336e-03

Table A.22: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *island* topology and the PT trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	0-14-7	0-21-0	0-20-1	0-12-9	4-17-0	4-17-0	4-17-0	5-16-0	3-18-0	3-18-0	0-21-0	0-21-0	0-16-5	0-14-7	0-20-1	1-20-0	**7-14-0**	0-21-0	1-20-0	0-19-2	0-21-0	0-21-0
A1L	0-12-9	0-18-3	0-21-0	0-18-3	1-20-0	7-14-0	1-20-0	**8-13-0**	1-20-0	1-20-0	0-20-1	0-21-0	0-18-3	0-18-3	0-18-3	0-21-0	7-14-0	0-21-0	1-20-0	0-18-3	1-20-0	0-21-0
A1R	0-16-5	0-20-1	0-18-3	0-18-3	4-17-0	**12-9-0**	8-13-0	9-12-0	4-17-0	0-21-0	0-21-0	0-21-0	0-16-5	0-15-6	0-20-1	0-18-3	1-20-0	0-16-5	0-18-3	0-19-2	0-21-0	0-20-1
A2C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2L	0-20-1	0-20-1	0-20-1	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-18-3	0-21-0	2-19-0	**6-15-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	**1-20-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3R	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	**2-19-0**	0-21-0	0-21-0	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C1C	0-8-13	2-19-0	0-20-1	0-18-3	2-19-0	3-18-0	2-19-0	4-17-0	2-18-1	2-19-0	0-20-1	2-19-0	0-20-1	0-20-1	0-8-13	0-19-2	**11-10-0**	2-19-0	0-20-1	2-19-0	2-19-0	2-18-1
C1L	0-11-10	0-21-0	0-20-1	0-19-2	1-20-0	3-18-0	1-20-0	**6-15-0**	1-20-0	2-19-0	0-21-0	0-21-0	0-21-0	0-18-3	0-20-1	0-21-0	1-20-0	1-20-0	0-20-1	1-20-0	1-20-0	0-21-0
C1R	0-13-8	0-21-0	0-21-0	0-20-1	2-19-0	**8-13-0**	3-18-0	2-19-0	2-19-0	2-19-0	0-21-0	0-21-0	2-19-0	0-20-1	0-19-2	0-20-1	2-19-0	0-20-1	0-13-8	0-21-0	0-21-0	0-20-1
C2C	12-9-0	0-16-5	0-15-6	0-16-5	0-20-1	0-16-5	0-16-5	0-20-1	0-18-3	2-19-0	0-16-5	0-16-5	0-16-5	0-16-5	0-20-1	12-9-0	13-8-0	13-8-0	**16-5-0**	0-16-5	0-15-6	0-16-5
C2L	0-2-19	1-20-0	1-20-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	1-20-0	0-7-14	1-20-0	2-19-0	1-20-0	2-19-0	2-19-0	2-19-0	0-21-0
C2R	0-21-0	0-21-0	0-16-5	0-18-3	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	4-17-0	0-21-0	0-16-5	0-17-4	0-19-2	0-21-0	2-19-0	5-16-0	6-15-0	**7-14-0**	0-21-0	0-17-4	0-21-0
C3C	0-21-0	0-21-0	0-19-2	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	2-19-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	**3-18-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C3L	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-20-1	0-21-0	**2-19-0**	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C3R	0-17-4	0-21-0	0-21-0	0-19-2	0-21-0	2-19-0	0-21-0	**7-14-0**	4-17-0	2-19-0	0-21-0	0-21-0	0-19-2	0-21-0	0-17-4	0-20-1	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-20-1
P1C	0-12-9	2-19-0	0-20-1	0-17-4	1-20-0	2-19-0	0-20-1	6-15-0	1-20-0	3-18-0	0-20-1	2-19-0	0-20-1	0-19-2	0-10-11	0-19-2	**11-10-0**	2-19-0	0-19-2	3-18-0	2-19-0	0-20-1
P1L	0-9-12	1-19-1	0-20-1	0-20-1	1-19-1	2-19-0	0-20-1	5-16-0	1-19-1	2-19-0	2-19-0	0-20-1	0-20-1	0-19-2	0-14-7	0-19-2	**15-6-0**	1-19-1	0-19-2	2-19-0	1-19-1	2-19-0
P1R	0-7-14	1-19-1	1-19-1	0-18-3	1-20-0	1-20-0	1-20-0	6-15-0	1-20-0	4-17-0	1-19-1	1-20-0	0-20-1	0-19-2	0-18-3	0-20-1	**14-7-0**	0-19-2	0-18-3	1-19-1	1-19-1	1-19-1
P2C	0-18-3	0-21-0	0-18-3	0-17-4	0-20-1	0-21-0	0-21-0	5-16-0	0-20-1	7-14-0	0-19-2	0-20-1	0-19-2	0-17-4	0-18-3	0-21-0	**11-10-0**	2-19-0	0-21-0	0-21-0	0-21-0	0-20-1
P2L	0-7-14	1-20-0	0-19-2	0-18-3	1-20-0	1-20-0	1-20-0	3-18-0	1-20-0	2-19-0	1-20-0	0-21-0	0-20-1	0-20-1	0-17-4	0-20-1	**7-14-0**	1-20-0	1-20-0	4-17-0	1-20-0	1-20-0
P2R	0-12-9	2-19-0	0-20-1	0-19-2	2-19-0	2-19-0	0-21-0	5-16-0	2-19-0	2-19-0	0-20-1	0-20-1	0-20-1	0-19-2	0-12-9	0-19-2	**10-11-0**	2-19-0	0-21-0	2-19-0	0-20-1	0-21-0
P3C	0-10-11	2-19-0	0-20-1	0-19-2	1-20-0	2-19-0	0-20-1	5-16-0	1-20-0	2-19-0	1-20-0	1-20-0	0-20-1	0-18-3	0-14-7	0-19-2	**11-10-0**	3-18-0	0-20-1	2-19-0	0-20-1	0-20-1
P3L	0-7-14	2-19-0	2-19-0	0-20-1	1-20-0	1-20-0	0-20-1	2-19-0	1-20-0	2-19-0	2-19-0	2-19-0	0-21-0	2-19-0	0-10-11	0-20-1	**7-14-0**	2-19-0	0-20-1	2-19-0	0-20-1	2-19-0
P3R	0-2-19	2-19-0	2-19-0	1-19-1	2-19-0	2-19-0	1-19-1	2-19-0	1-20-0	2-19-0	1-19-1	2-19-0	1-20-0	1-19-1	0-12-9	1-19-1	**10-11-0**	1-20-0	0-20-1	2-19-0	1-19-1	1-19-1
μ_{wins}	0.44	0.59	0.22	0.11	0.96	2	0.89	3.07	1.04	1.89	0.37	0.44	0.19	0.15	0	0.74	**5.78**	1.37	1.04	0.85	0.44	0.33
σ_{wins}	2.31	0.84	0.58	**0.42**	1.16	2.87	1.76	2.87	1.19	1.53	0.69	0.8	0.56	0.46	0	2.33	5.01	2.69	3.3	1.17	0.7	0.68
μ_{draws}	13.85	19.96	19.63	18.78	19.93	18.81	19.7	17.89	19.74	19.11	20.15	20.04	19.48	19.04	17.11	19.56	15.22	19.3	19.07	19.67	19.96	20.15
σ_{draws}	6.25	1.26	1.5	1.91	1.14	2.9	1.86	2.83	1.26	1.53	1.17	1.4	1.6	1.87	4.5	2.29	5.01	2.73	3.44	1.41	1.43	1.2
μ_{loss}	6.7	0.44	1.15	2.11	0.11	0.19	0.41	0.04	0.22	0	0.48	0.52	1.33	1.81	3.89	0.7	0	0.33	0.89	0.48	0.59	0.52
σ_{loss}	6.32	1.12	1.54	1.97	0.32	0.96	1.01	0.19	0.64	0	1.05	1.34	1.66	1.94	4.5	0.91	0	1.04	1.69	1.19	1.37	1.01

Table A.23: Friedman ranks and p-values achieved by each investigated hyper-heuristic in each environment. Each hyper-heuristic used the *island* topology and the ST trigger (where $k = 20\%$). Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
p values																						4.859149e-04
A1C	14.65	10.93	10.58	11.90	**10.10**	10.20	12.15	10.85	10.55	10.60	11.22	10.39	10.84	11.89	14.15	12.30	11.53	11.25	12.70	11.40	11.80	11.01
A1L	14.89	10.73	11.32	**10.20**	12.08	11.21	11.37	11.23	11.15	11.65	11.63	10.92	10.45	10.30	13.18	11.56	10.70	13.42	11.65	10.40	11.89	11.03
A1R	14.32	11.01	11.58	13.58	10.38	**9.99**	10.21	11.18	10.70	11.40	11.48	10.13	11.94	11.98	14.01	11.92	10.44	12.46	12.15	10.90	10.60	10.65
A2C	12.93	11.08	11.20	11.68	12.68	12.50	11.61	12.52	10.43	10.96	**9.68**	10.00	10.73	10.65	12.49	11.82	12.49	12.17	12.39	12.07	10.68	10.24
A2L	13.73	10.97	10.85	12.56	11.06	**10.32**	11.84	11.10	11.26	11.18	11.60	12.01	11.46	11.44	11.70	11.51	11.13	11.50	12.35	11.36	10.69	11.37
A2R	12.65	11.34	**10.08**	11.08	10.80	11.35	10.76	11.53	10.61	11.16	11.56	11.38	12.18	12.70	12.35	12.44	11.26	11.22	13.00	11.38	11.36	10.82
A3C	12.61	10.51	11.26	11.95	10.39	11.32	12.42	11.11	12.13	**10.00**	10.53	11.23	11.14	11.33	12.50	12.82	11.37	12.26	12.55	11.10	10.92	11.55
A3L	14.87	10.32	11.37	11.36	11.04	11.08	10.08	12.22	10.30	10.46	**10.03**	10.30	11.90	12.55	12.35	12.31	12.36	12.98	11.63	10.35	10.56	12.58
A3R	12.46	10.01	12.56	12.27	10.83	11.82	10.20	12.92	10.93	11.18	10.19	10.11	12.04	11.89	11.96	12.38	**9.92**	11.52	12.62	12.25	11.39	11.54
C1C	15.63	11.63	11.68	11.08	11.45	10.15	11.08	**9.14**	10.63	9.92	10.61	11.51	12.11	11.92	15.90	13.51	9.46	10.39	12.63	10.25	10.76	11.54
C1L	16.37	11.13	10.54	11.13	11.48	10.68	11.66	9.83	10.80	**8.80**	11.75	11.85	11.32	12.89	14.58	11.56	9.83	12.25	12.04	10.01	10.13	12.38
C1R	14.62	11.63	11.15	11.58	11.24	10.58	10.90	10.30	10.11	**9.66**	10.68	11.59	11.34	11.27	13.86	12.77	9.77	12.61	12.68	11.00	11.59	12.07
C2C	10.24	12.70	13.52	12.61	12.37	12.61	12.21	10.77	12.03	**8.69**	13.08	12.69	12.58	13.01	11.56	9.73	9.44	9.03	9.62	9.90	11.77	12.83
C2L	15.76	11.48	11.18	12.27	10.62	10.80	10.34	10.54	10.21	9.87	12.08	11.04	12.48	12.28	15.03	12.13	8.93	11.80	12.20	**7.82**	11.93	12.21
C2R	11.44	13.03	12.56	12.18	12.41	11.24	12.32	10.41	12.39	8.68	11.87	13.59	11.55	11.66	12.94	10.68	9.34	**8.56**	9.63	12.10	13.04	11.37
C3C	10.66	11.58	11.77	13.34	12.89	11.55	12.65	11.61	11.56	**9.72**	11.41	11.63	11.52	11.63	12.69	11.15	10.45	10.58	11.30	9.87	10.87	12.56
C3L	13.07	11.96	12.66	12.79	11.75	11.27	12.90	9.82	10.90	10.83	11.69	12.23	12.20	12.46	11.97	9.93	**9.14**	11.28	11.04	11.75	9.31	12.06
C3R	13.77	11.69	11.59	12.08	11.59	10.07	11.21	**9.01**	10.48	10.03	12.04	12.24	11.66	13.80	14.11	11.54	9.58	12.70	10.61	11.34	11.34	10.51
P1C	14.97	11.59	11.80	10.62	10.89	9.75	11.45	9.77	11.28	10.23	10.80	11.75	12.25	12.04	15.85	12.93	**8.65**	10.63	12.83	9.90	11.58	11.44
P1L	15.86	12.73	11.07	11.30	11.54	9.61	11.42	9.73	10.89	10.04	11.56	12.14	11.87	11.79	14.56	13.23	**8.46**	10.15	12.35	9.93	11.31	11.45
P1R	16.27	12.04	10.24	11.30	10.55	9.30	11.86	**9.21**	10.44	9.70	10.25	12.32	12.21	11.25	15.69	13.79	9.85	11.45	12.00	10.23	11.42	11.63
P2C	13.61	12.15	11.58	11.63	10.15	10.55	11.92	10.94	11.25	9.87	11.77	11.83	12.42	12.93	12.63	11.61	**9.55**	9.65	11.80	12.51	11.55	11.08
P2L	15.31	11.58	11.99	11.39	11.92	10.24	11.90	10.08	12.14	10.46	10.08	11.17	12.55	13.54	15.10	13.14	**8.59**	10.35	11.80	9.58	10.07	10.03
P2R	15.08	11.73	11.84	12.27	10.90	9.66	11.10	9.93	11.44	10.41	10.23	11.11	11.55	12.06	14.87	13.34	**9.03**	11.37	11.63	10.92	11.61	10.93
P3C	15.52	11.11	11.76	11.31	10.52	10.62	10.82	**9.58**	11.13	9.89	10.52	11.86	11.86	11.97	15.72	12.87	9.59	10.46	13.25	9.63	11.24	11.76
P3L	15.82	12.23	11.37	11.76	11.42	10.25	9.96	11.68	11.35	10.13	11.56	11.41	10.85	11.21	14.86	12.99	**9.66**	10.37	12.27	10.45	10.68	10.75
P3R	16.96	11.30	**8.73**	11.13	12.03	10.45	11.18	9.46	11.61	9.42	10.52	12.45	11.27	11.28	14.17	13.52	10.69	12.24	12.94	10.79	10.10	10.76
Mean	14.22	11.49	11.40	11.79	11.30	10.71	11.39	10.61	11.06	10.18	11.13	11.51	11.71	11.99	13.73	12.20	**10.05**	11.28	11.99	10.71	11.12	11.41
p values																						6.324894e-02

Table A.24: Wins, draws, and losses achieved by each investigated hyper-heuristic in each environment using the Shaffer post hoc test (at $\alpha = 0.05$). Each hyper-heuristic used the *island* topology and the ST trigger (where $k = 20\%$). The notation *W-D-L* indicates the wins, draws, and losses for each hyper-heuristic. Bold values indicate the best configurations.

Env	IFix	Rand	RoulM	RoulX	Freq2	Freq5	HTour2M	HTour5M	HTour2X	HTour5X	ARank	NARank	AProp	NAProp	ETour2	ETour25	RLFreq	RLProp	DProp	Comp	Soft	Perm
A1C	0-15-6	0-21-0	1-20-0	0-21-0	**2-19-0**	1-20-0	0-21-0	0-21-0	1-20-0	1-20-0	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A1L	0-15-6	1-20-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	1-20-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0
A1R	0-18-3	0-21-0	0-21-0	0-21-0	0-21-0	**2-19-0**	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A2R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
A3L	0-13-8	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	1-20-0	1-20-0	1-20-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	1-20-0	0-21-0
A3R	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C1C	0-6-15	2-19-0	1-20-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	0-21-0	1-20-0	0-4-17	0-19-2	3-18-0	2-19-0	0-21-0	2-19-0	2-19-0	2-19-0
C1L	0-2-19	1-20-0	2-19-0	1-20-0	1-20-0	1-20-0	1-20-0	2-19-0	1-20-0	**3-18-0**	1-20-0	1-20-0	1-20-0	0-20-1	0-15-6	1-20-0	2-19-0	1-20-0	1-20-0	2-19-0	2-19-0	1-20-0
C1R	0-16-5	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	1-20-0	1-20-0	2-19-0	0-21-0	0-21-0	0-21-0	0-21-0	0-19-2	0-21-0	2-19-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C2C	0-21-0	0-20-1	0-18-3	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	**5-16-0**	0-19-2	0-21-0	0-21-0	0-20-1	0-21-0	0-21-0	1-20-0	2-19-0	0-21-0	0-21-0	0-21-0	0-20-1
C2L	0-10-11	1-20-0	1-20-0	0-20-1	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	0-20-1	2-19-0	0-20-1	0-20-1	0-12-9	0-20-1	2-19-0	0-20-1	0-20-1	**11-10-0**	0-20-1	0-20-1
C2R	0-21-0	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	4-17-0	0-21-0	0-18-3	0-21-0	0-21-0	0-19-2	0-21-0	1-20-0	4-17-0	0-21-0	0-21-0	0-19-2	0-21-0
C3C	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C3L	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
C3R	0-19-2	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	3-18-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-19-2	0-17-4	0-21-0	3-18-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P1C	0-12-9	1-20-0	1-20-0	2-19-0	2-19-0	2-19-0	1-20-0	2-19-0	1-20-0	2-19-0	2-19-0	1-20-0	0-21-0	0-21-0	0-5-16	0-20-1	**4-17-0**	2-19-0	0-20-1	2-19-0	1-20-0	1-20-0
P1L	0-5-16	0-20-1	1-20-0	1-20-0	1-20-0	2-19-0	1-20-0	2-19-0	1-20-0	2-19-0	1-20-0	0-21-0	1-20-0	1-20-0	0-15-6	0-20-1	**4-17-0**	2-19-0	0-21-0	2-19-0	1-20-0	1-20-0
P1R	0-3-18	1-20-0	2-19-0	2-19-0	2-19-0	3-18-0	1-20-0	3-18-0	2-19-0	3-18-0	2-19-0	0-21-0	1-20-0	2-19-0	0-7-14	0-18-3	2-19-0	2-19-0	1-20-0	2-19-0	2-19-0	2-19-0
P2C	0-20-1	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0	1-20-0	0-21-0	0-21-0	0-21-0	0-21-0	0-21-0
P2L	0-11-10	0-21-0	0-21-0	0-21-0	0-21-0	2-19-0	0-21-0	2-19-0	0-21-0	2-19-0	2-19-0	1-20-0	0-21-0	0-20-1	0-12-9	0-20-1	**4-17-0**	2-19-0	0-21-0	2-19-0	2-19-0	2-19-0
P2R	0-13-8	0-21-0	0-21-0	0-21-0	1-20-0	2-19-0	0-21-0	2-19-0	0-21-0	2-19-0	2-19-0	0-21-0	0-21-0	0-21-0	0-16-5	0-20-1	**3-18-0**	0-21-0	0-21-0	1-20-0	0-21-0	1-20-0
P3C	0-8-13	2-19-0	0-21-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	2-19-0	0-21-0	0-21-0	0-21-0	0-8-13	0-21-0	2-19-0	2-19-0	0-21-0	2-19-0	2-19-0	0-21-0
P3L	0-4-17	0-21-0	1-20-0	1-20-0	1-20-0	2-19-0	2-19-0	1-20-0	2-19-0	2-19-0	1-20-0	1-20-0	2-19-0	1-20-0	0-12-9	0-21-0	2-19-0	2-19-0	0-21-0	2-19-0	2-19-0	2-19-0
P3R	0-2-19	1-20-0	**4-17-0**	1-20-0	1-20-0	1-20-0	1-20-0	3-18-0	1-20-0	3-18-0	1-20-0	1-20-0	1-20-0	1-20-0	0-17-4	0-18-3	1-20-0	1-20-0	1-19-1	1-20-0	2-19-0	1-20-0
μ_{wins}	0	0.41	0.52	0.48	0.63	0.96	0.56	1.04	0.59	**1.44**	0.63	0.44	0.26	0.26	0	0.04	1.41	0.81	0.11	1.15	0.63	0.48
σ_{wins}	0	0.64	0.94	0.75	0.84	0.98	0.75	1.19	0.75	1.4	0.84	0.64	0.53	0.53	0	**0.19**	1.39	1.11	0.32	2.16	0.88	0.75
μ_{draws}	14.11	20.44	20.37	20.48	20.37	20.04	20.44	19.96	20.41	19.56	20.26	20.44	20.7	20.52	16.63	20.48	19.59	20.15	20.78	19.85	20.26	20.44
σ_{draws}	6.92	0.7	1.04	0.75	0.84	0.98	0.75	1.19	0.75	1.4	0.86	0.8	0.54	0.64	5.46	0.89	1.39	1.1	0.51	2.16	0.9	0.75
μ_{loss}	6.89	0.15	0.11	0.04	0	0	0	0	0	0	0.11	0.11	0.04	0.22	4.37	0.48	0	0.04	0.11	0	0.11	0.07
σ_{loss}	6.92	0.46	0.58	0.19	0	0	0	0	0	0	0.42	0.58	0.19	0.51	5.46	0.89	0	0.19	0.32	0	0.42	0.27

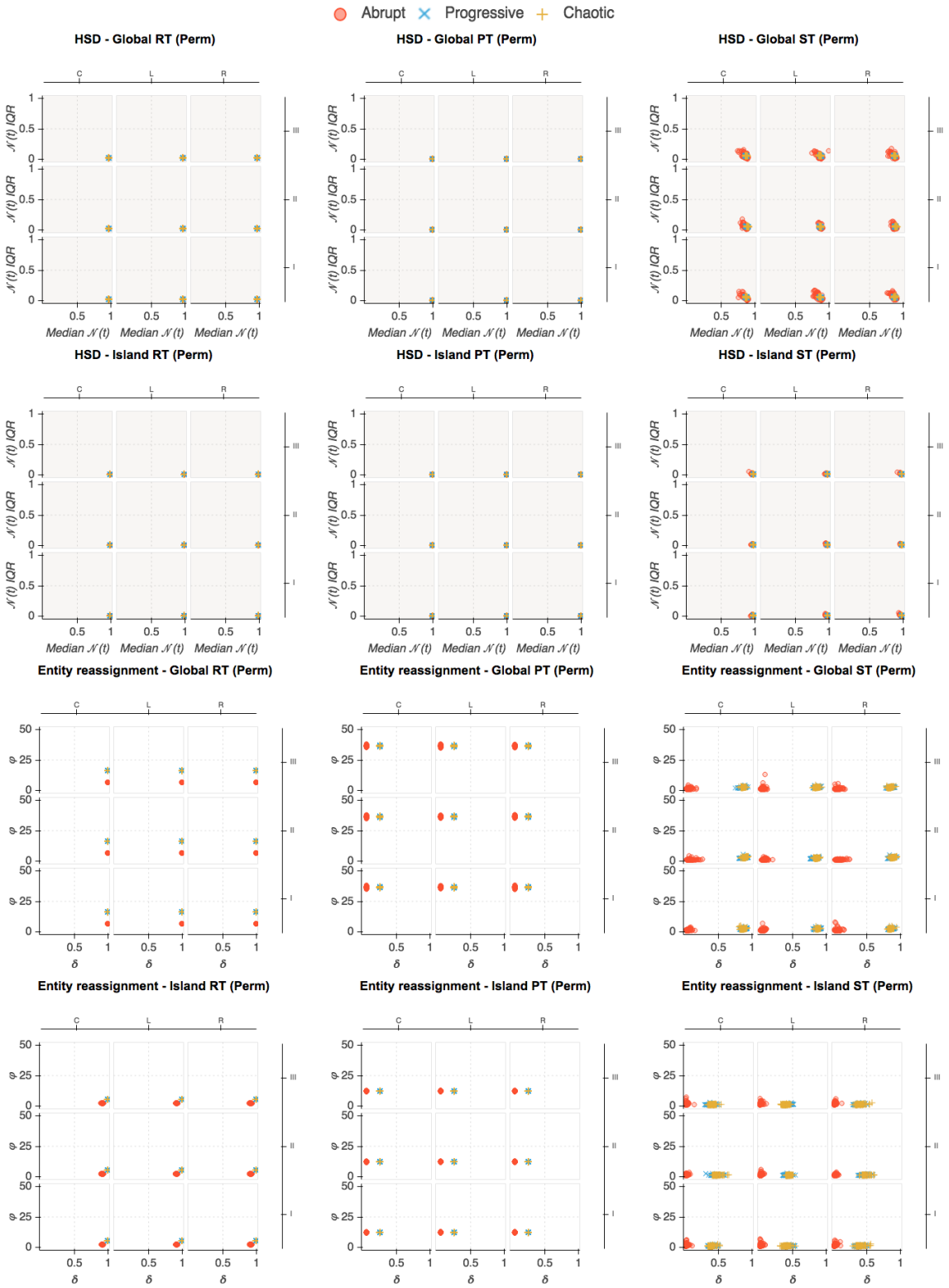


Figure A.1: HSD and entity reassignment rate analysis for **Perm**

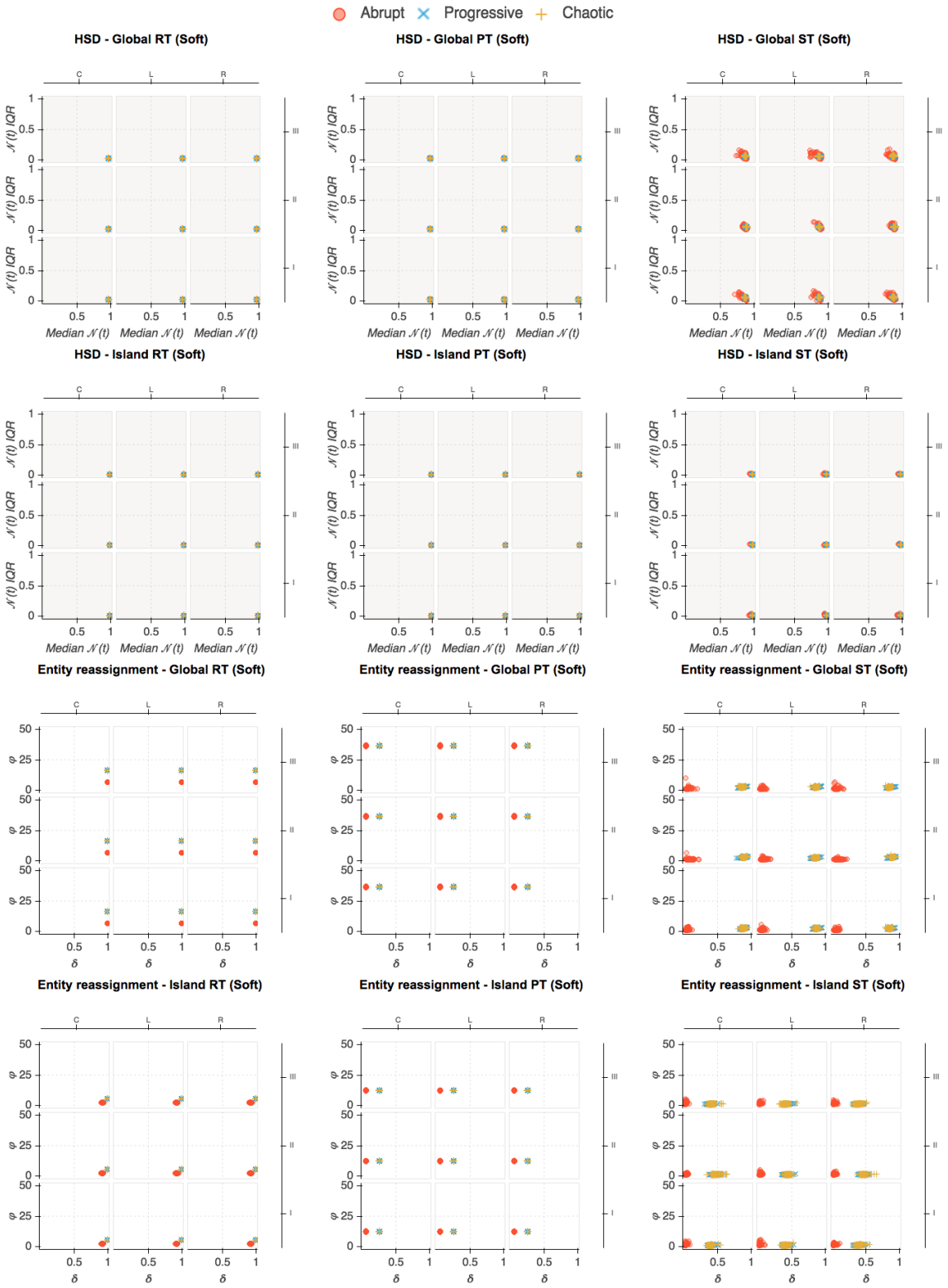


Figure A.2: HSD and entity reassignment rate analysis for **Soft**

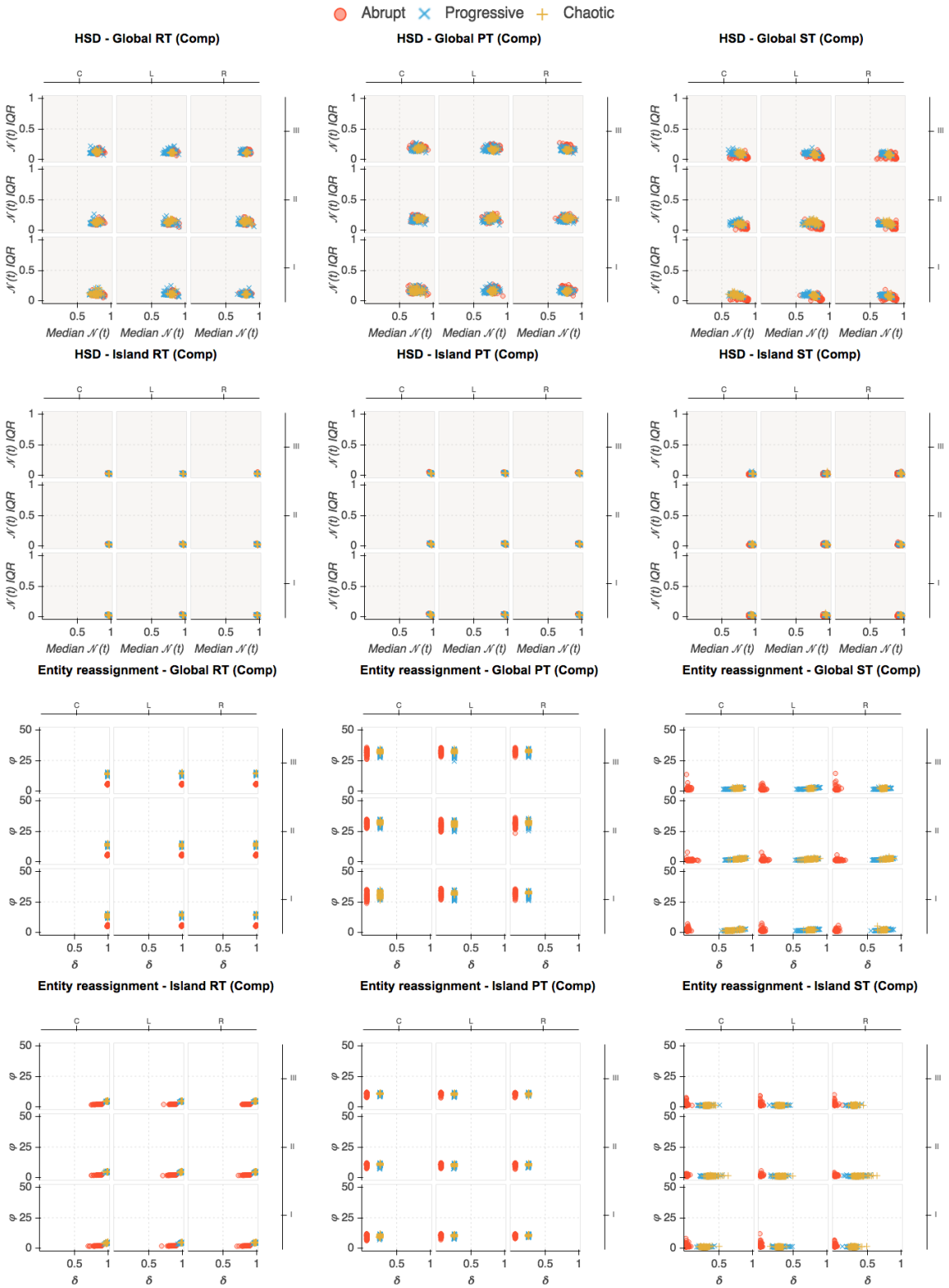


Figure A.3: HSD and entity reassignment rate analysis for **Comp**

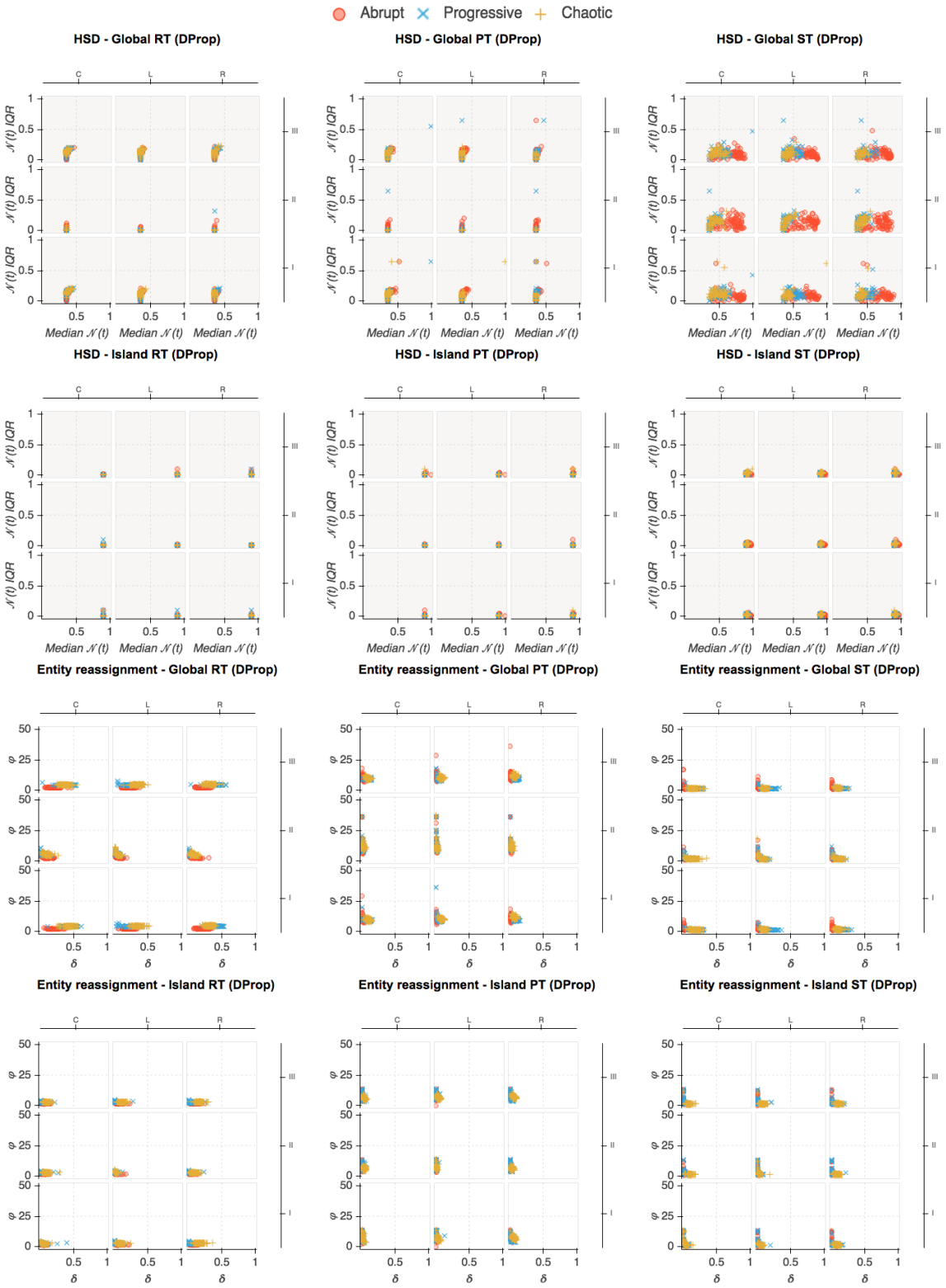


Figure A.4: HSD and entity reassignment rate analysis for **DProp**

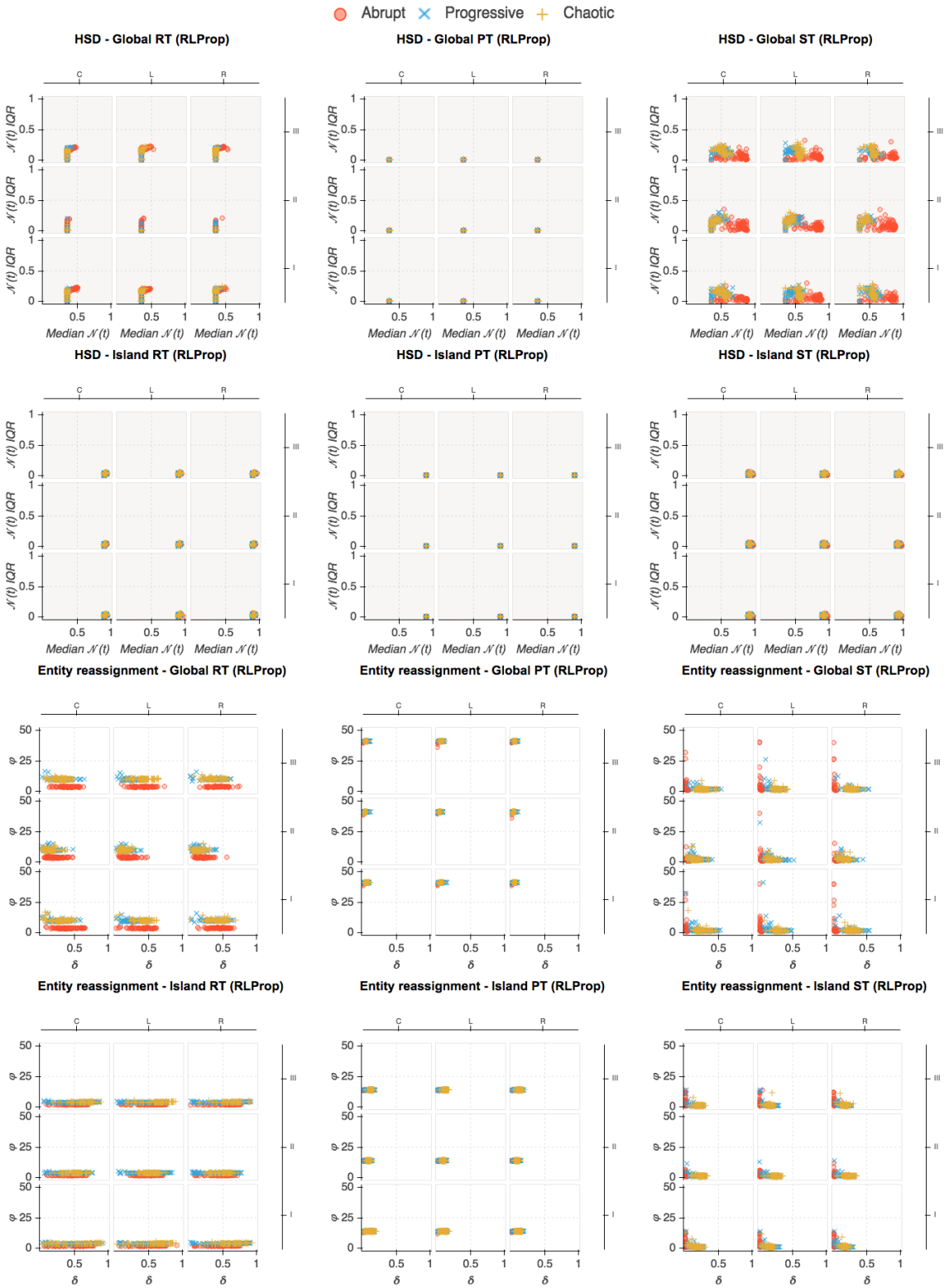


Figure A.5: HSD and entity reassignment rate analysis for **RLProp**

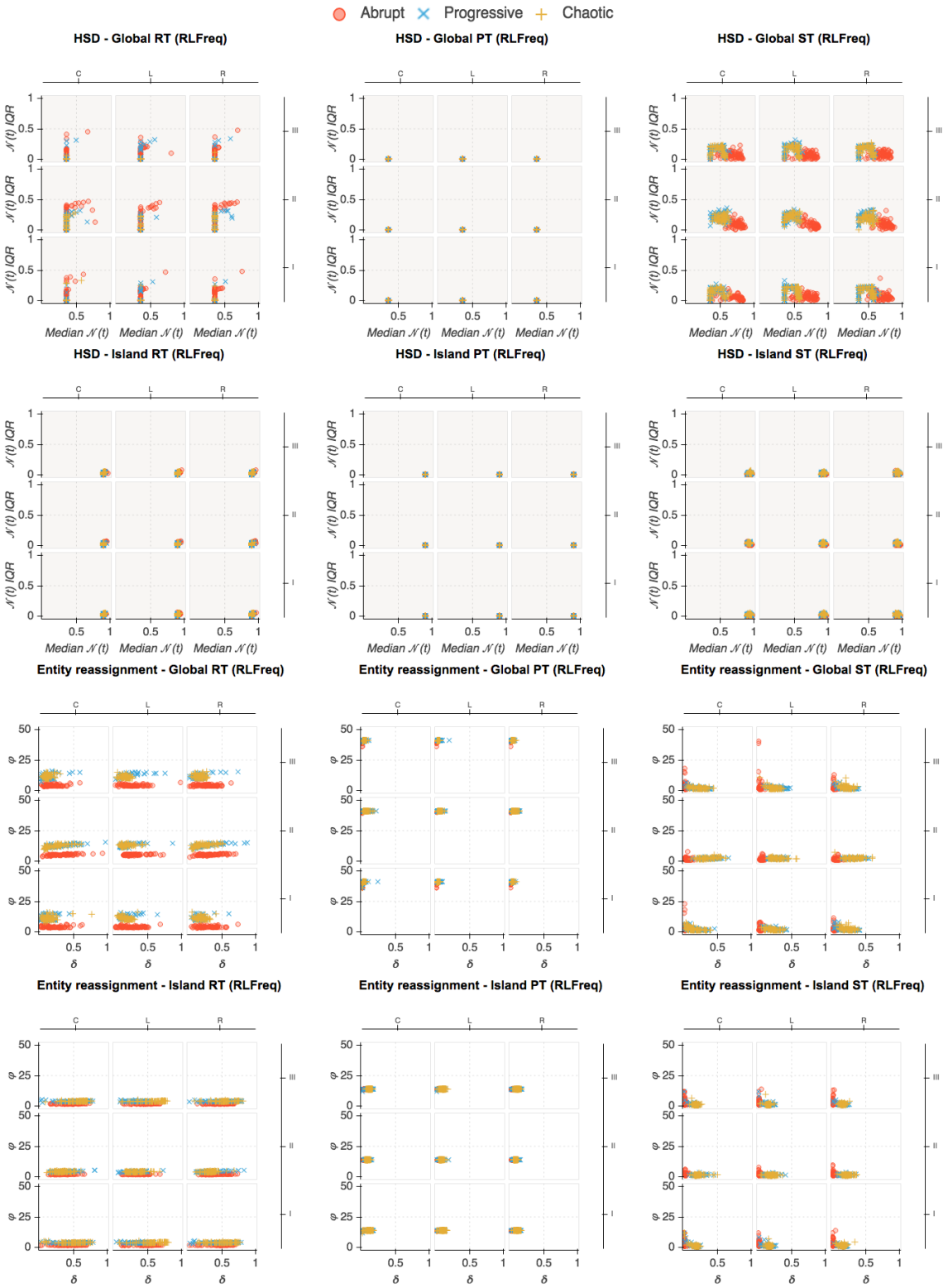


Figure A.6: HSD and entity reassignment rate analysis for **RLFreq**

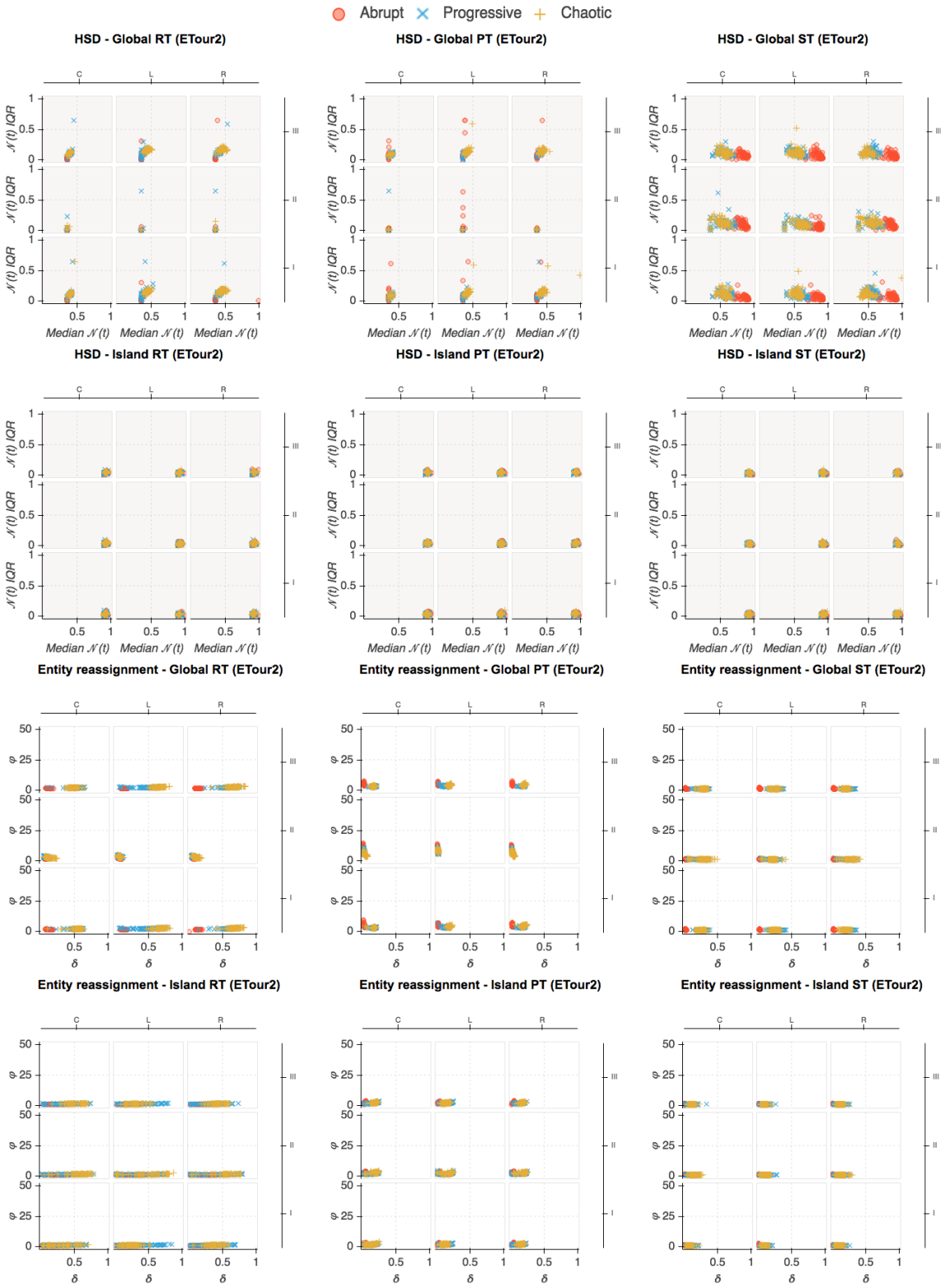


Figure A.7: HSD and entity reassignment rate analysis for **ETour2**

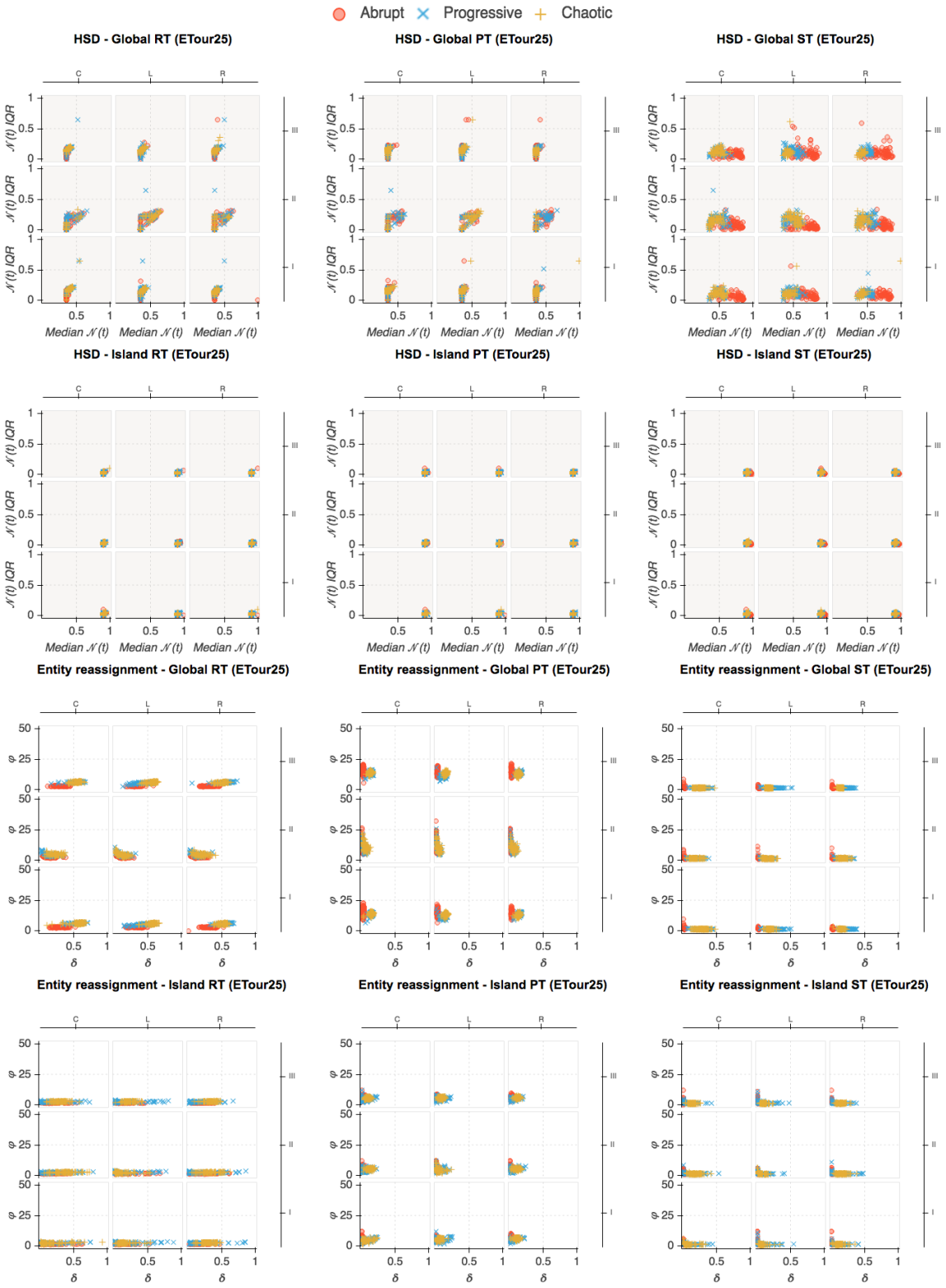


Figure A.8: HSD and entity reassignment rate analysis for **ETour25**

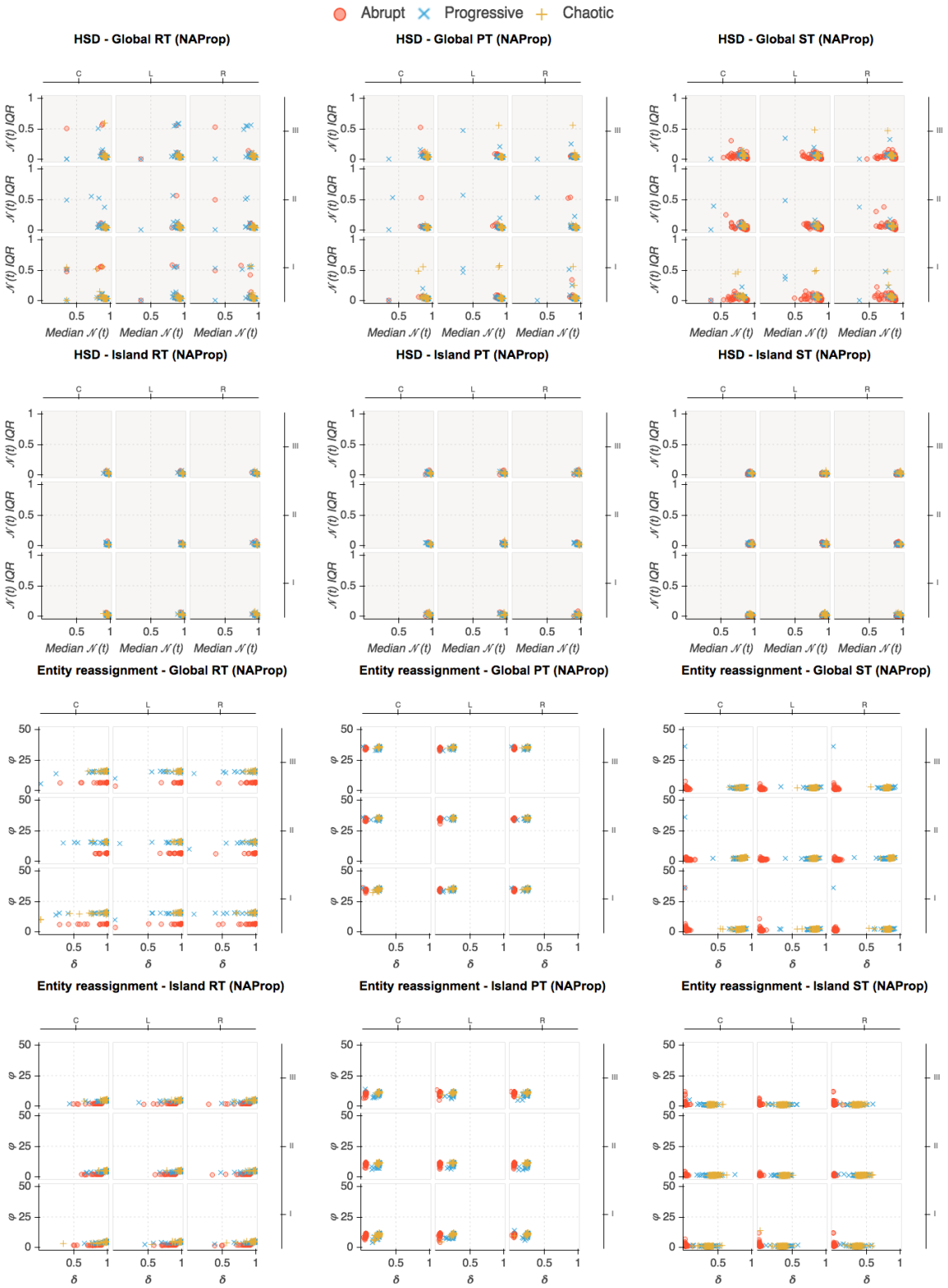


Figure A.9: HSD and entity reassignment rate analysis for **NAProp**

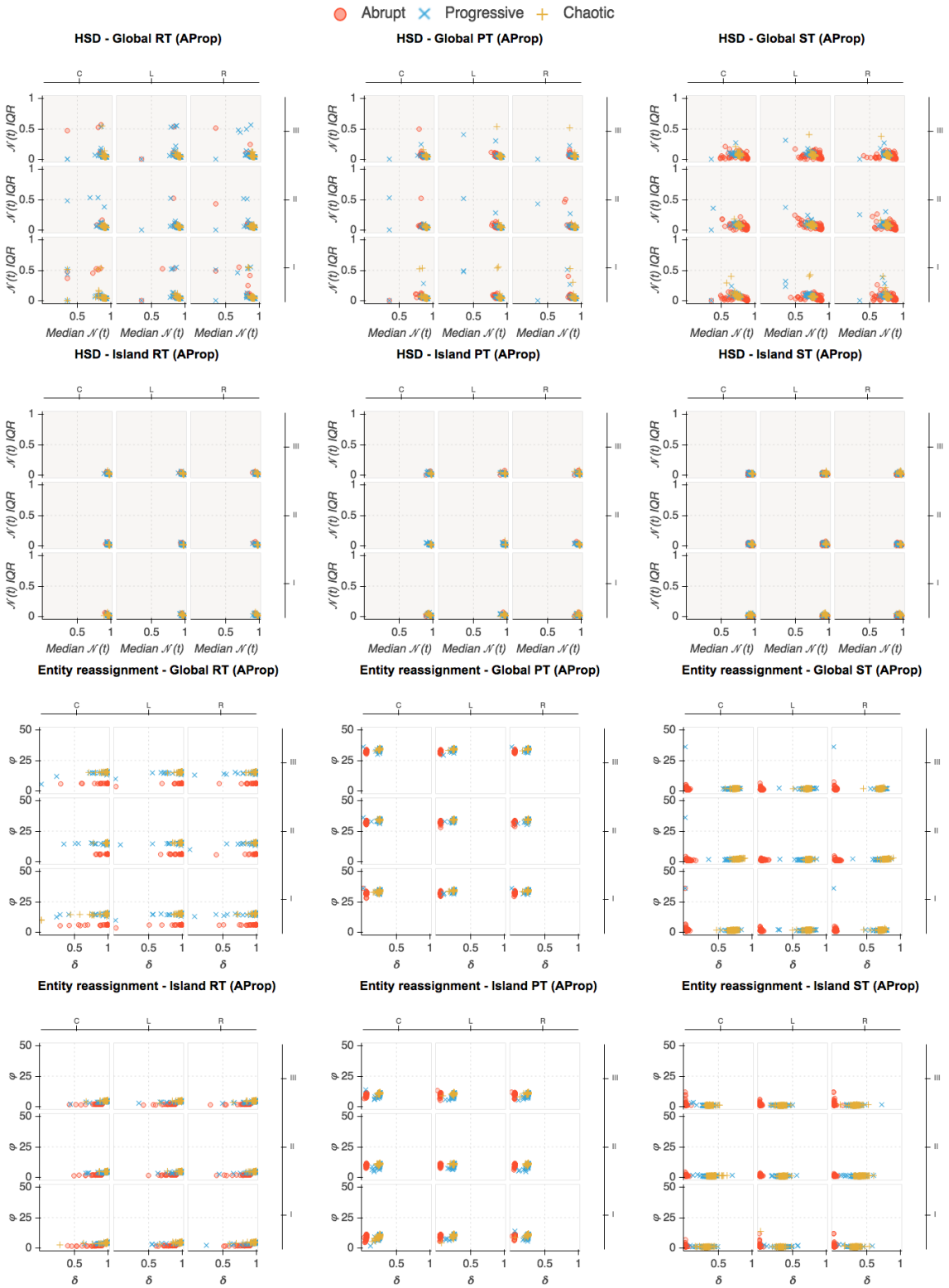


Figure A.10: HSD and entity reassignment rate analysis for **AProp**

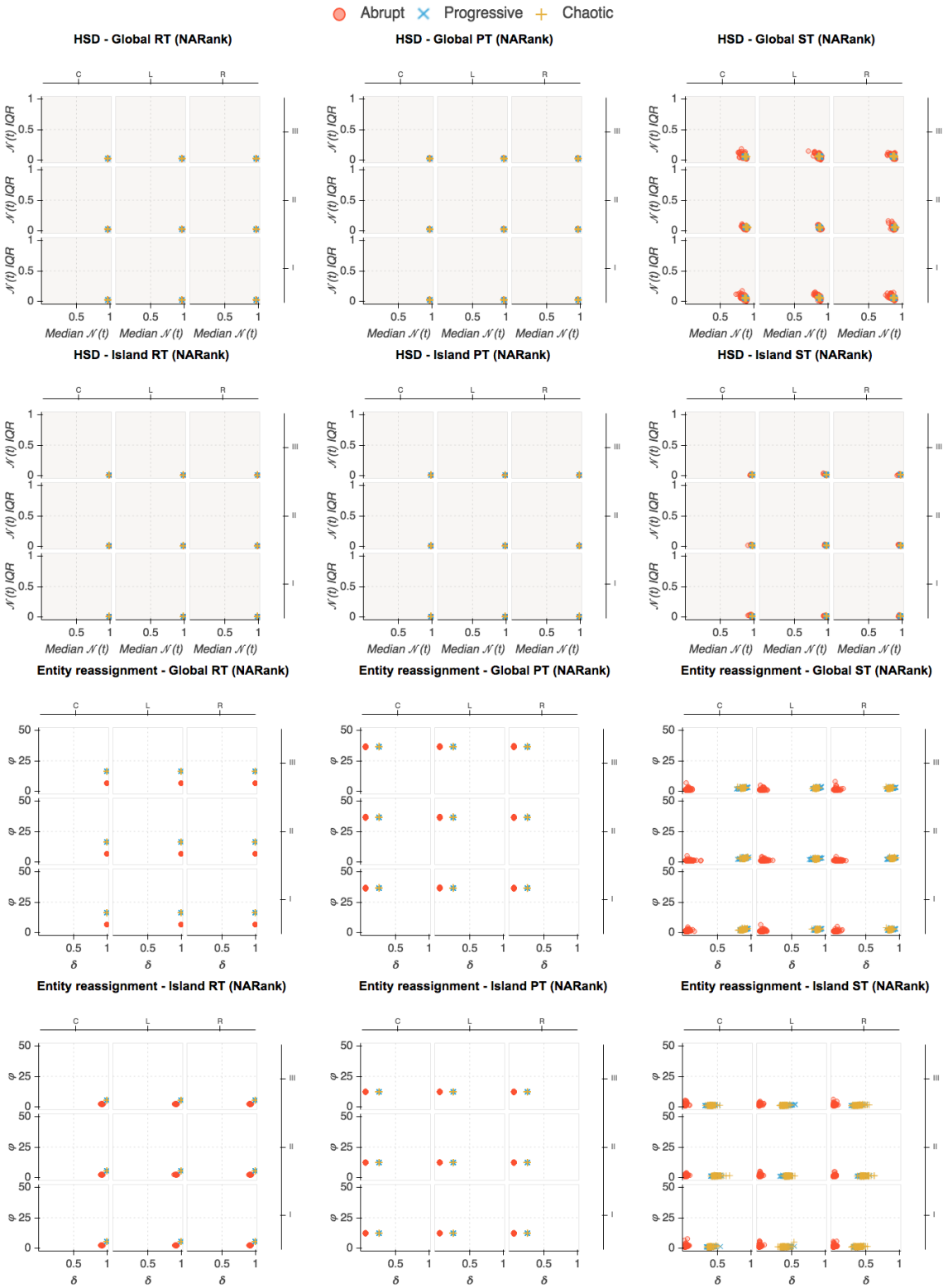


Figure A.11: HSD and entity reassignment rate analysis for **NARank**

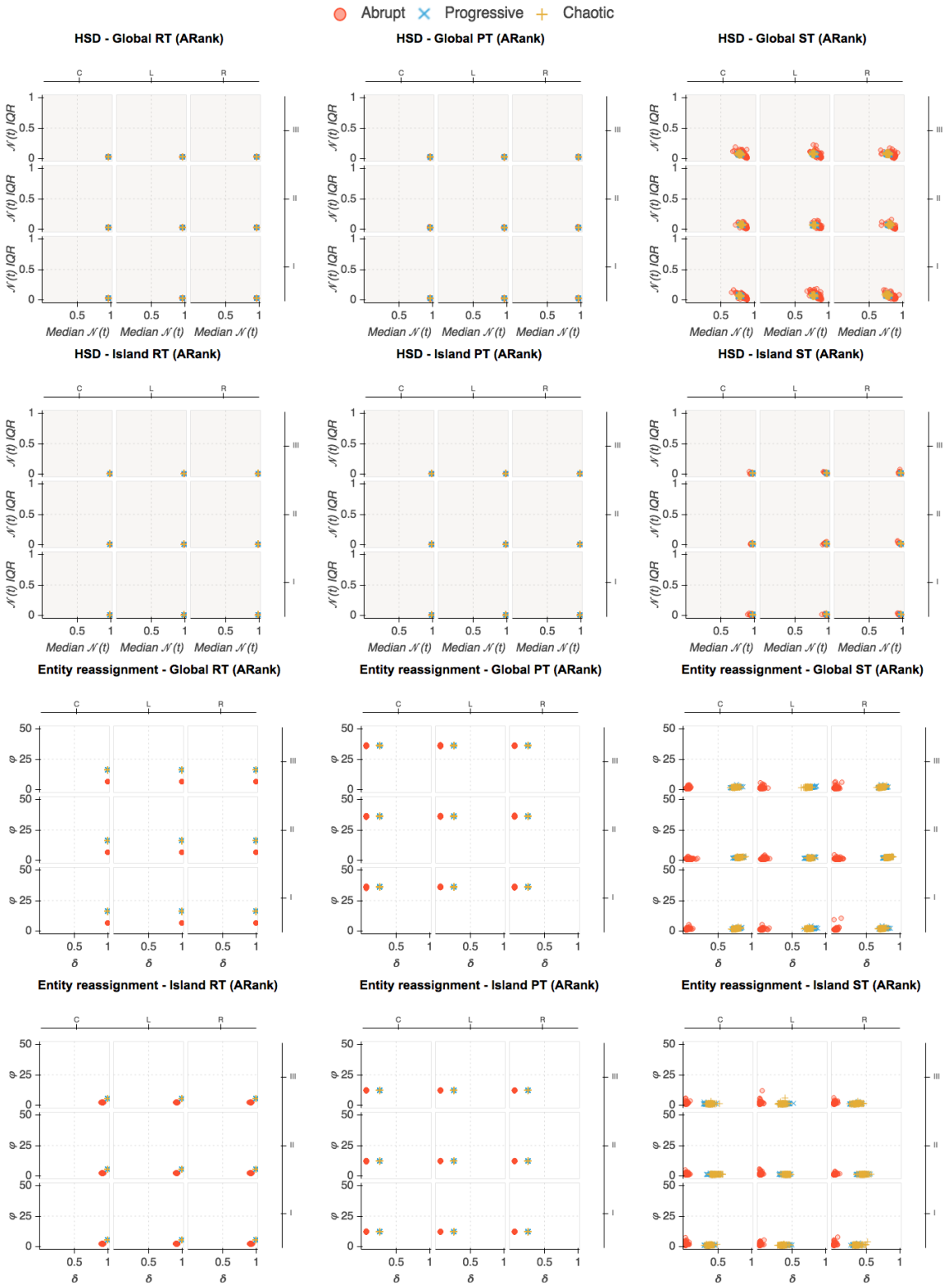


Figure A.12: HSD and entity reassignment rate analysis for **ARank**

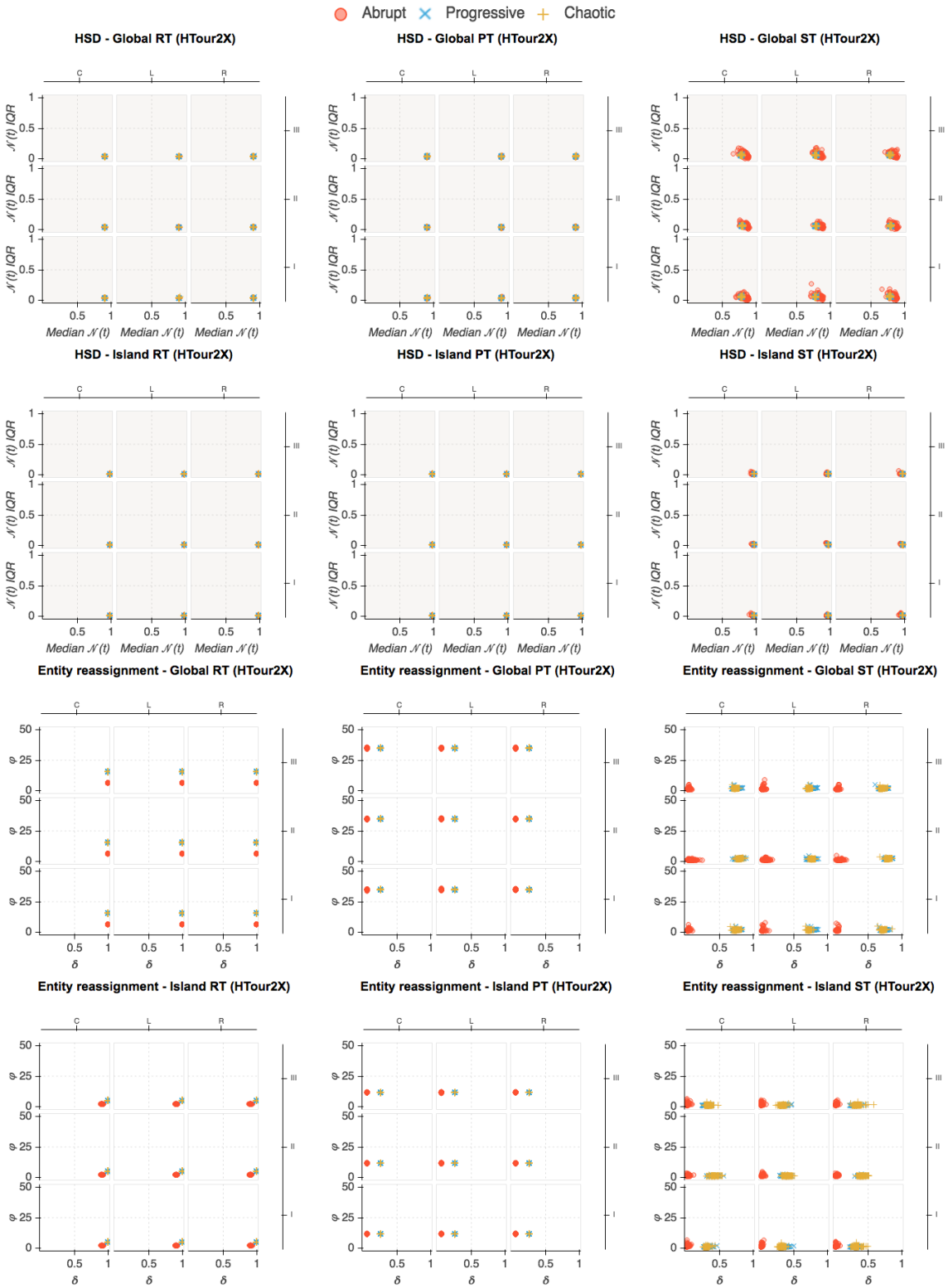


Figure A.13: HSD and entity reassignment rate analysis for **HTour2X**

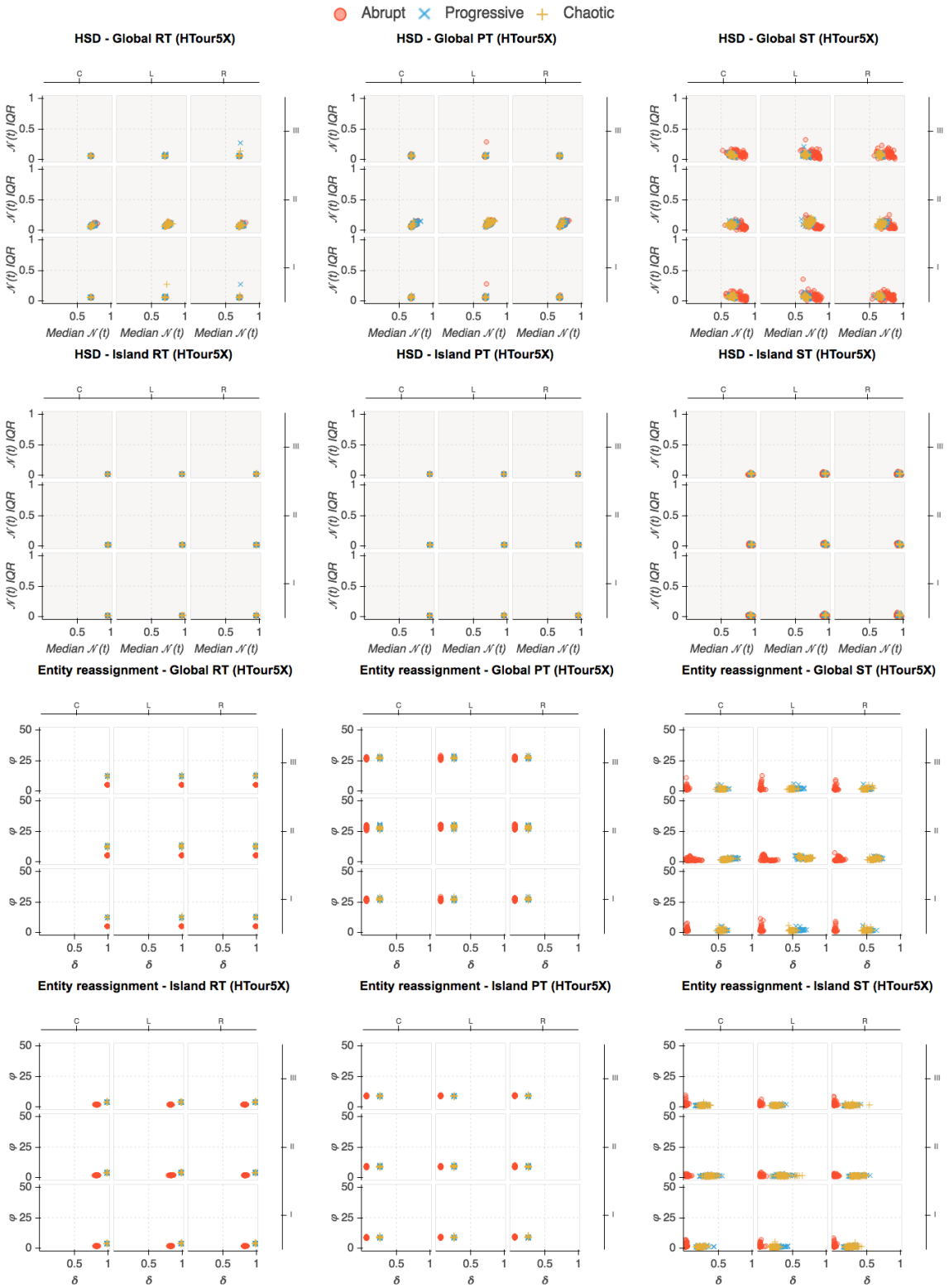


Figure A.14: HSD and entity reassignment rate analysis for **HTour5X**

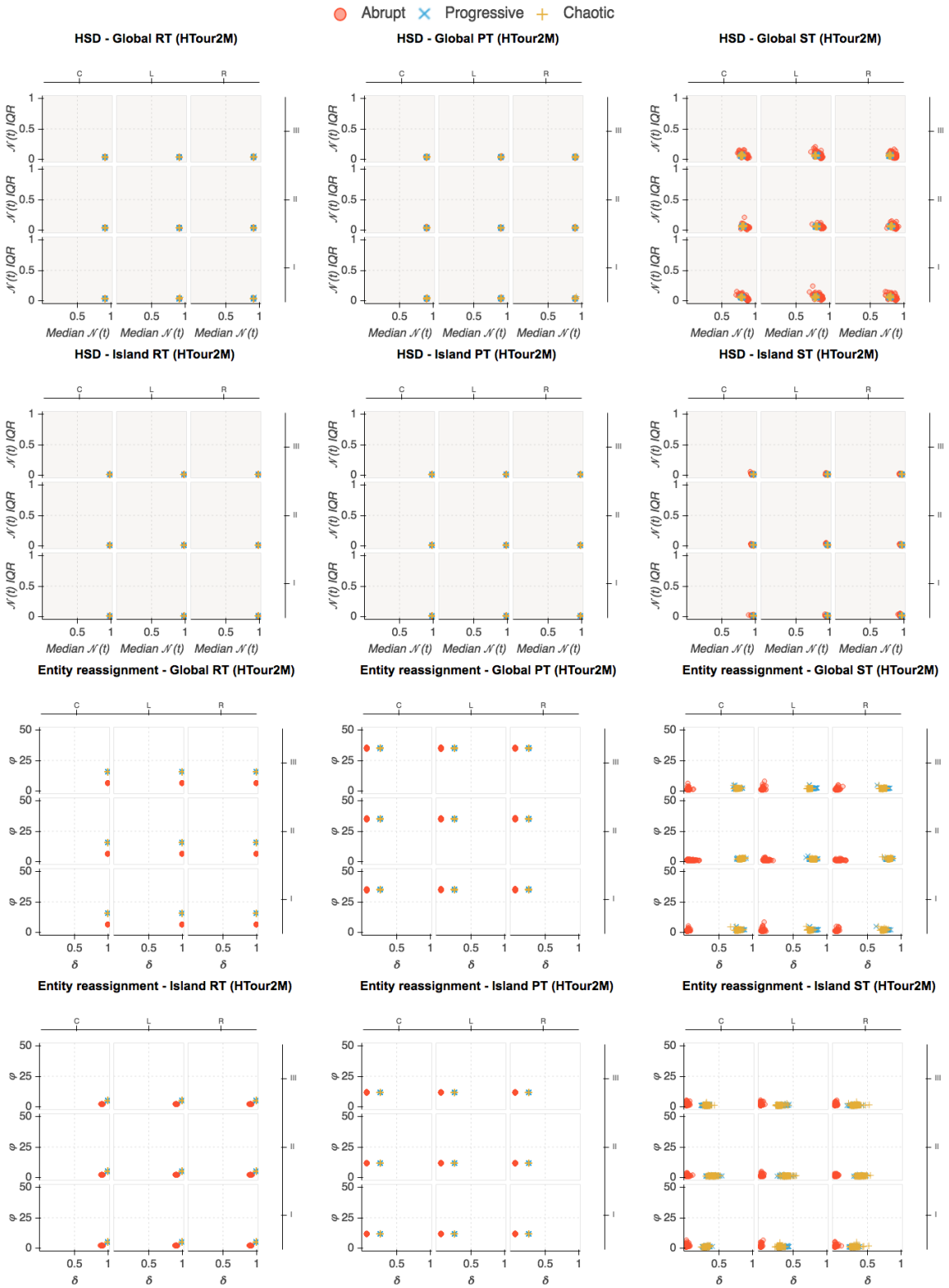


Figure A.15: HSD and entity reassignment rate analysis for **HTour2M**

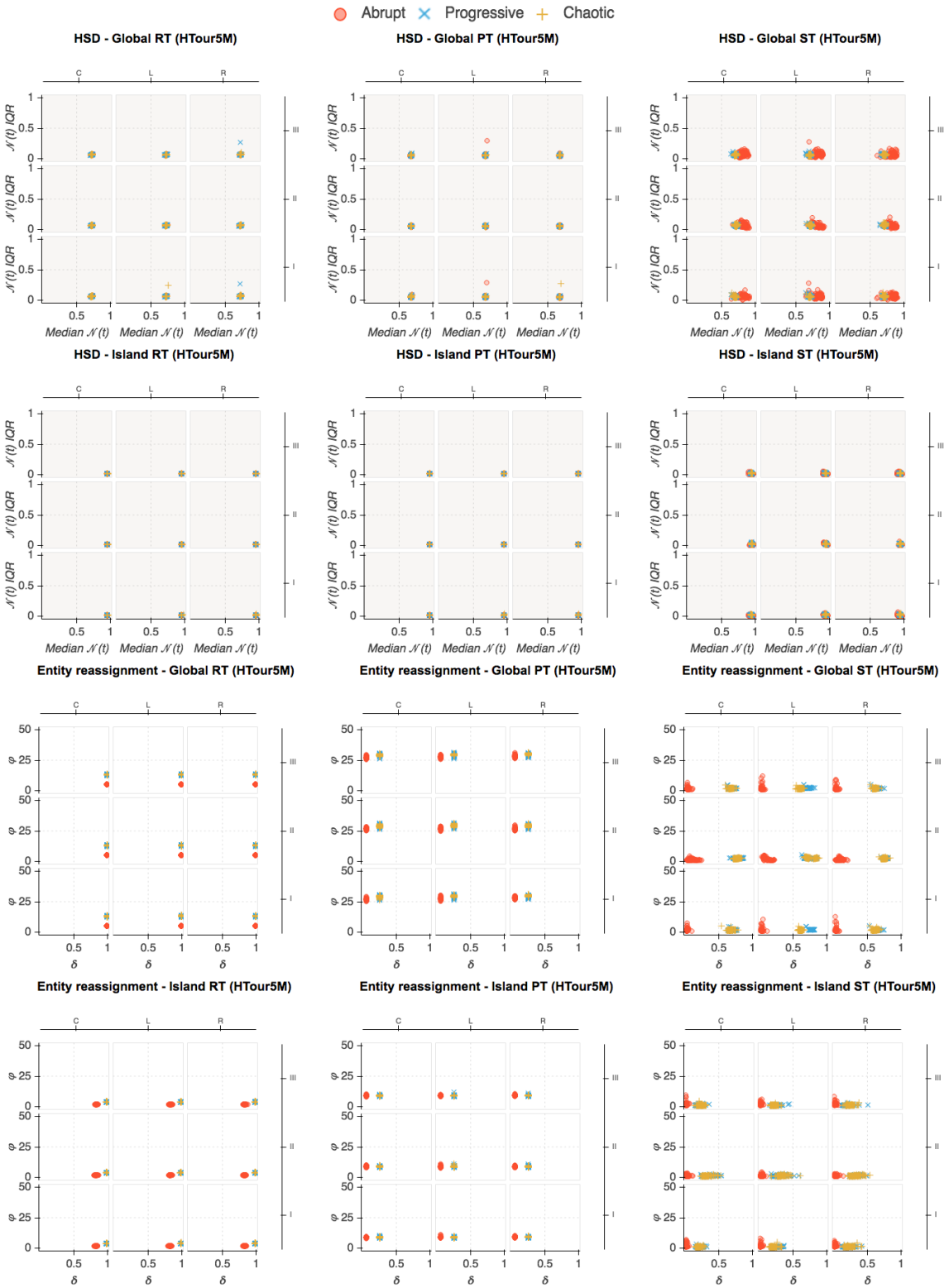


Figure A.16: HSD and entity reassignment rate analysis for **HTour5M**

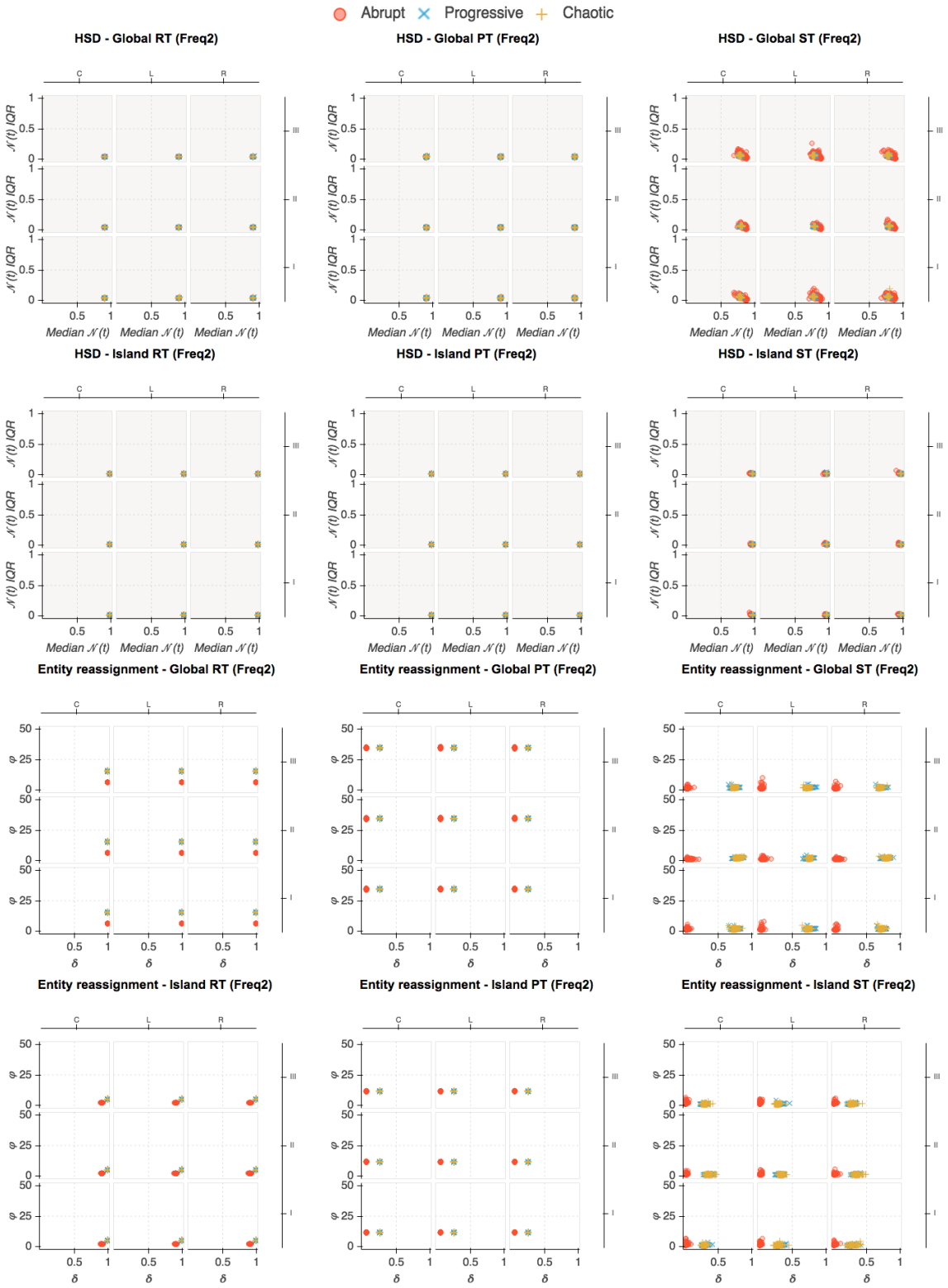


Figure A.17: HSD and entity reassignment rate analysis for **Freq2**

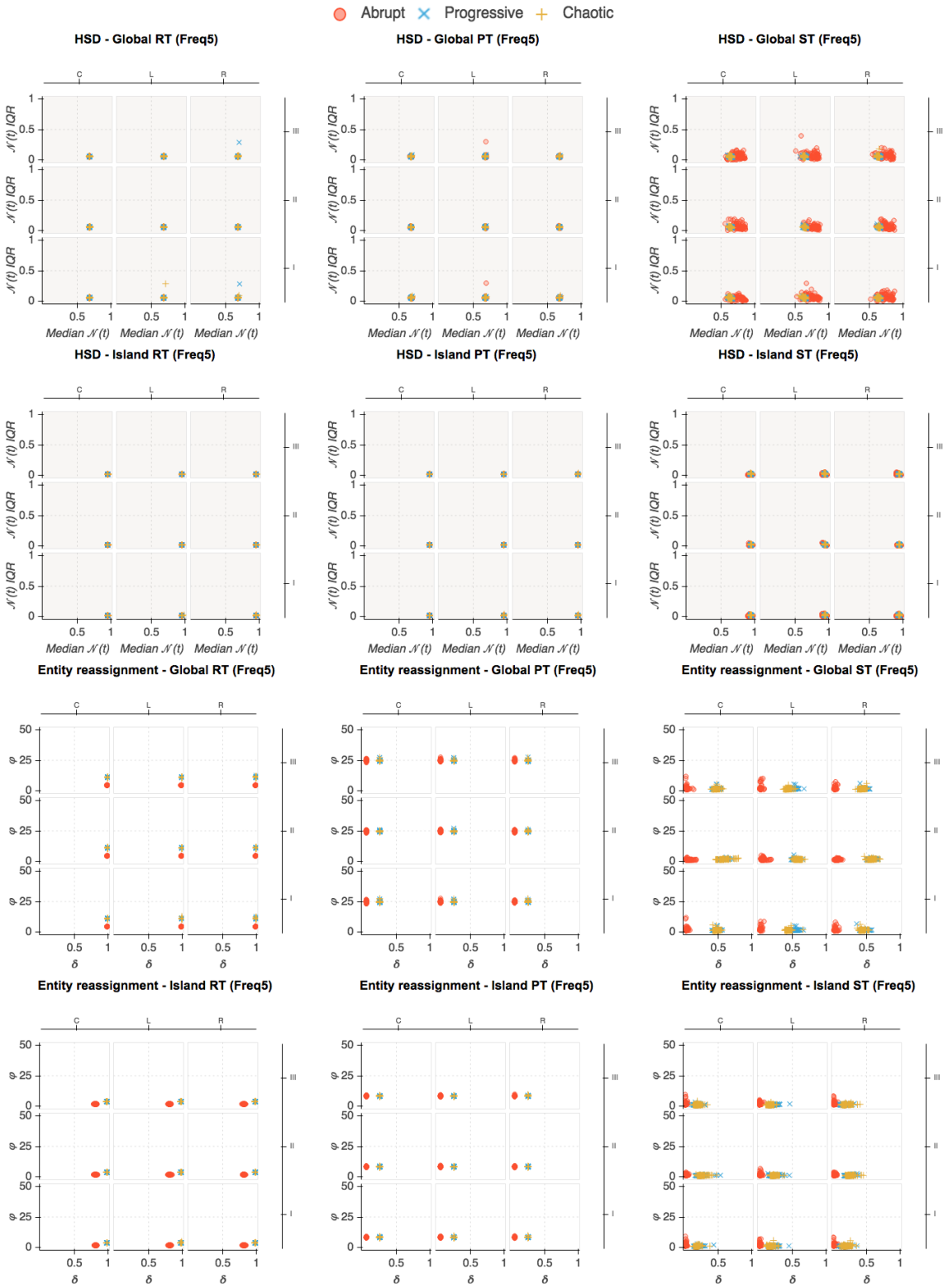


Figure A.18: HSD and entity reassignment rate analysis for **Freq5**

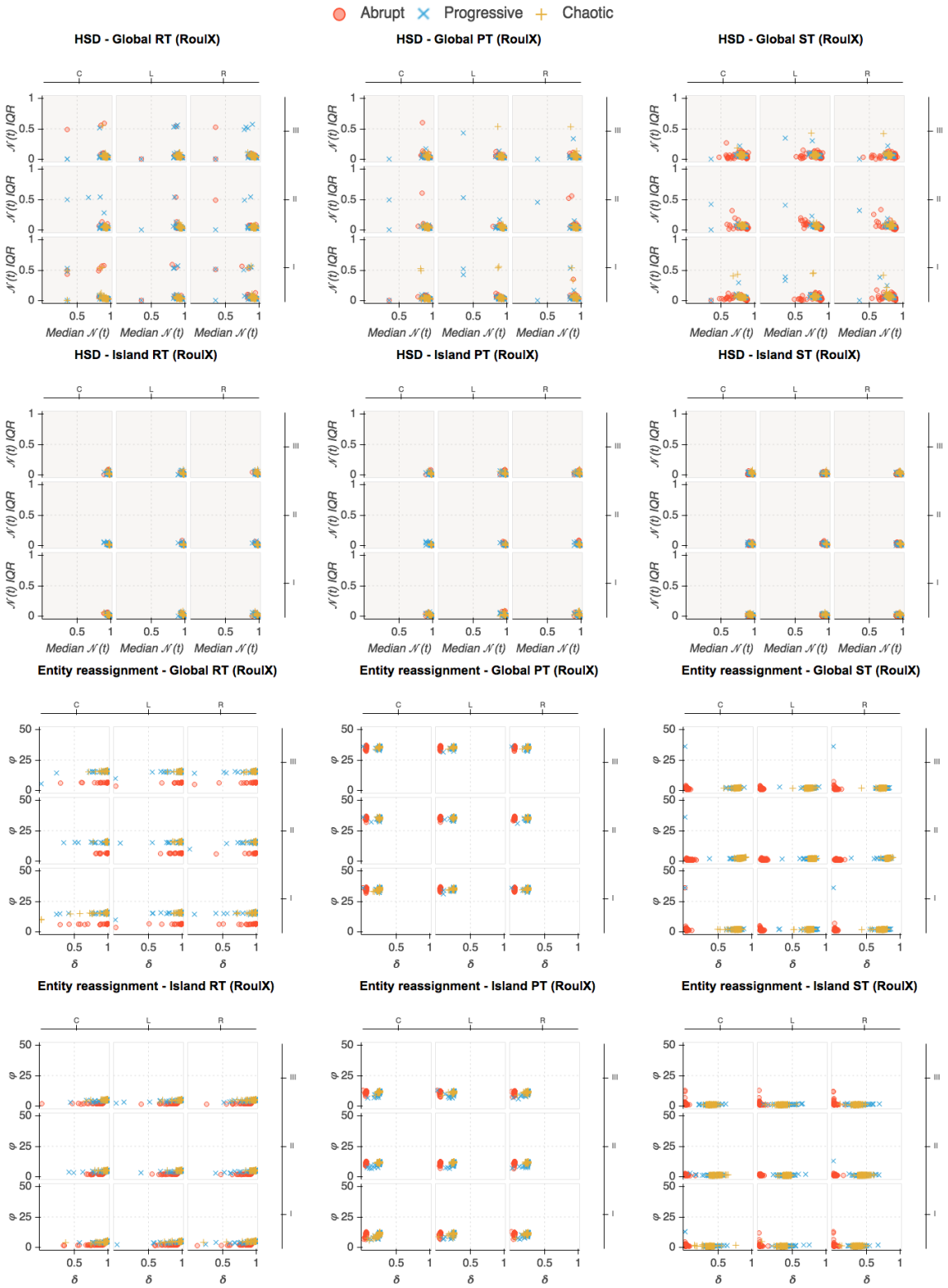


Figure A.19: HSD and entity reassignment rate analysis for **RoulX**

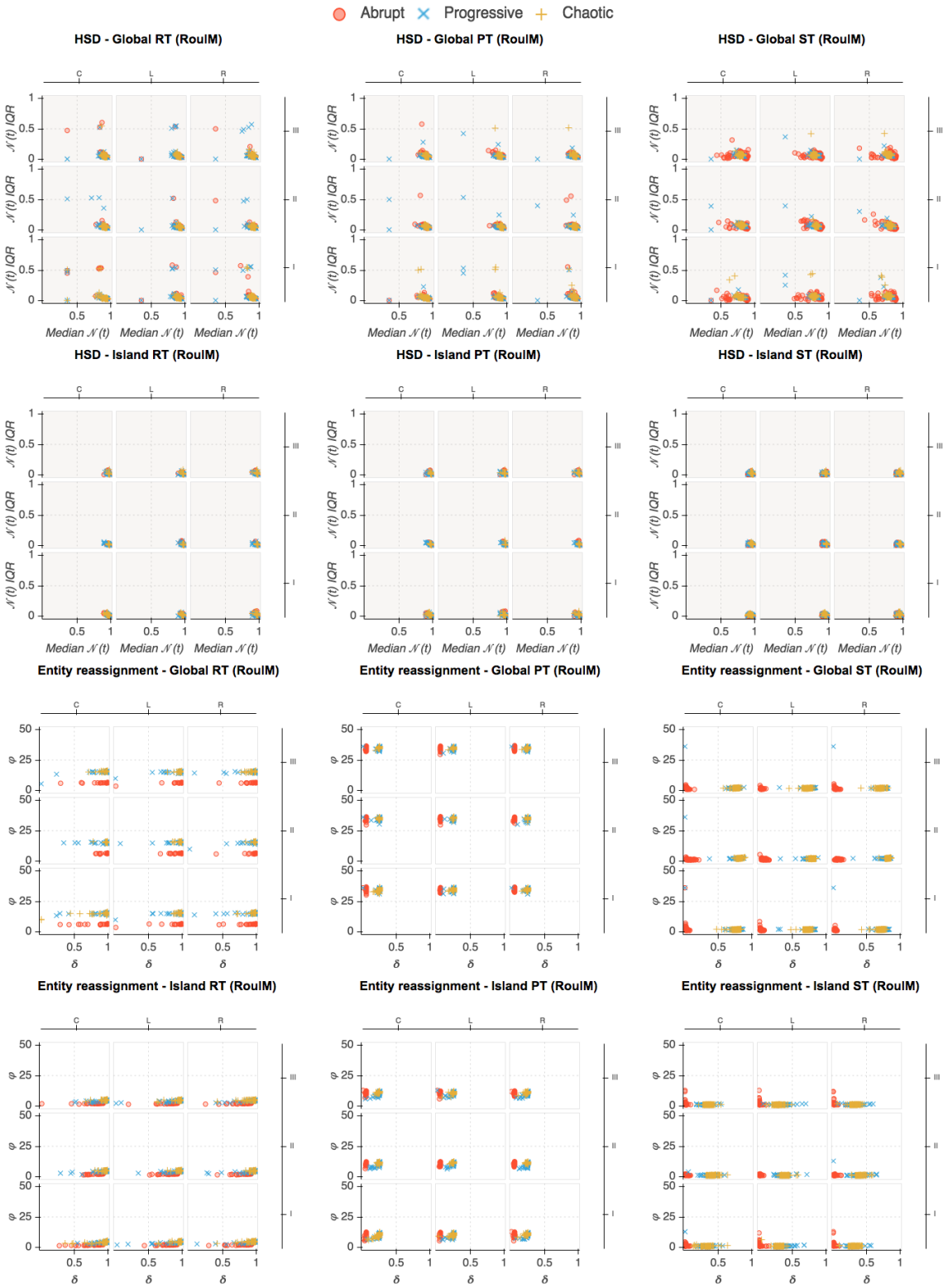


Figure A.20: HSD and entity reassignment rate analysis for **RoulM**

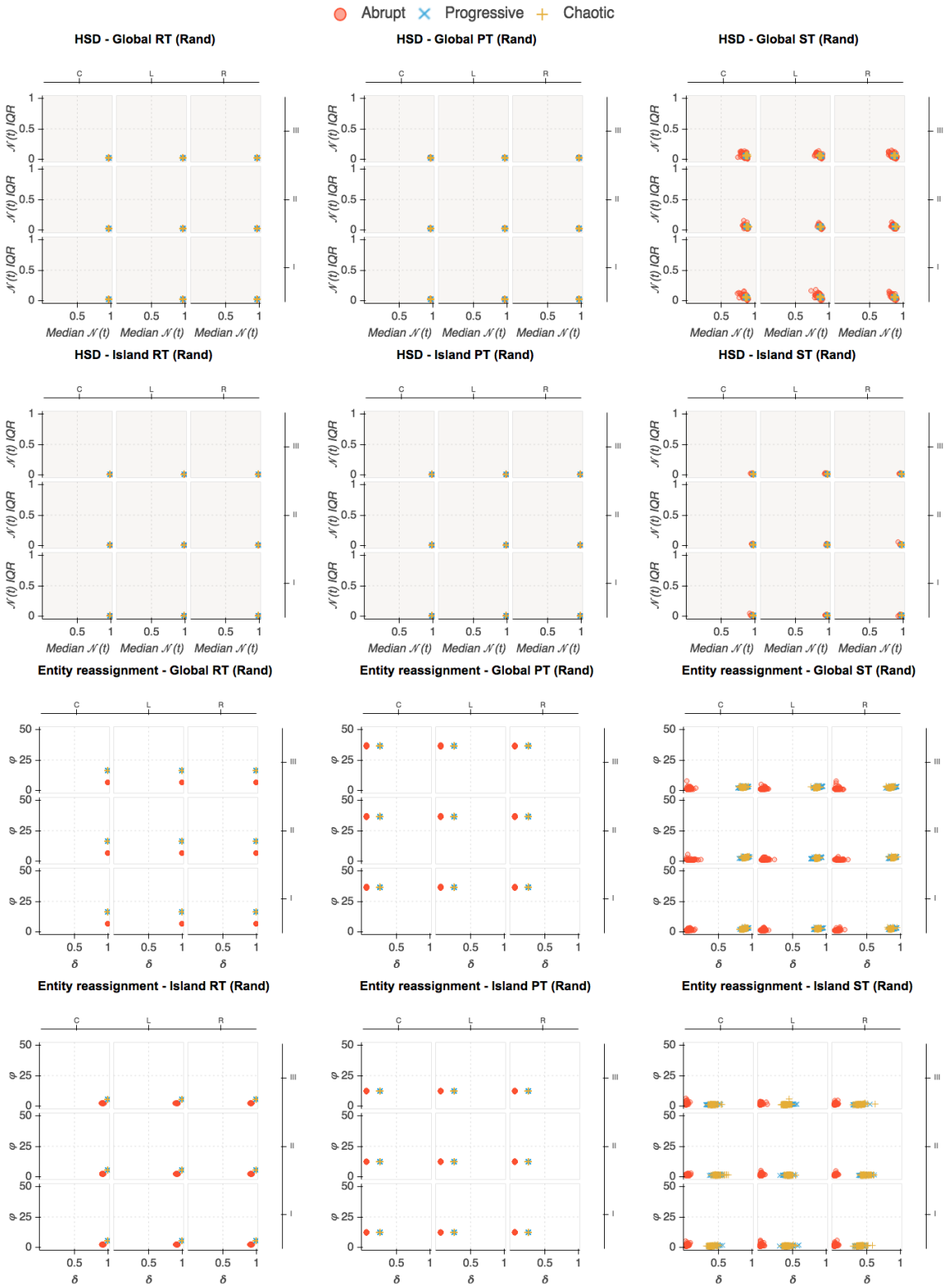


Figure A.21: HSD and entity reassignment rate analysis for **Rand**

Appendix B

Acronyms

ACO-MH	Ant colony optimization meta-heuristic
ADSC	Average distance to swarm center
AMP	Adaptive multi-population framework
AOS	Adaptive operator selection
APSO	Atomic particle swarm optimization
CI	Computational Intelligence
CDE	Competing differential evolution
CPE	Competitive population evaluation
CPSO	Charged particle swarm optimization
DE	Differential evolution
dEAs	Distributed evolutionary algorithms
DOP	Dynamic optimization problem
DynDE	Dynamic DE
EA	Evolutionary algorithm
EC	Evolutionary Computation
EDEV	Ensemble of DE variants
EDO	Evolutionary dynamic optimization
FP-HPSO	Frequency-based heterogeneous particle swarm optimization
FWER	Family-wise error rate

GA	Genetic algorithm
GMO	Gaussian mutation operator
HMHH	Heterogeneous meta-hyper-heuristic
HSD	Heuristic space diversity
IQR	Interquartile range
MA	Memetic algorithm
MMEO	Multi-phase multi-individual extremal optimization
MPEDE	Multi-population ensemble DE
MPB	Moving peaks benchmark
NFL	No free lunch
PAP	Population based algorithm portfolio
PBIL	Population based incremental learning
PC	Parameter control
PH-PSO	Partitioned hierarchical particle swarm optimization
PSO	Particle swarm optimization
PT	Periodic trigger
QPSO	Quantum particle swarm optimization
RIGA	Random immigrant genetic algorithm
RT	Random trigger
SADE	Self-adaptive differential evolution
SI	Swarm Intelligence
SIDO	Swarm intelligence dynamic optimization
ST	Stagnation trigger

Appendix C

Symbols

Chapter 2: Dynamic Optimization

$\mathbf{a}_{il}$	Acceleration coefficient between particles i and l
B_n	n_x-dimensional sphere of radius r_{cloud} centered the *gbest* particle
$B_{PD}(t)$	Population diameter
$B_{PR}(t)$	Population radius
$B_{ADSC}(t)$	Average distance around swarm center
$B_{ADP}(t)$	Mean of the average distance around all population members
$B_{MI}(t)$	Moment-of-inertia
$B_{stab}(t)$	Stability
$B_{rob}(t)$	Robustness
$B_{sat}(\Theta)$	Satisficability
$B_{react}(t, \varepsilon)$	Reactivity
B_{arr}	Absolute recovery rate
$\mathcal{C}(t)$	A collection of entities at time t
E	The objective function error
ε	Relative accuracy threshold for the *reactivity* measure
f, F	The objective function to be optimized
f_p	Cone function for the MPB function generator

h_p	Height of MPB peak p at time t
h_s	Height severity of the MPB
i	Entity counter
j	Dimension counter
k	Periods before a heuristic change
l	Entity counter
$l(t)$	Step length
$\mathbf{l}_p(t)$	Location of MPB peak p at time t
λ	Correlation coefficient of the MPB peak trajectories
M_i	Number of algorithm iterations before an MPB function change
M_C	The cycle length of a full function landscape rotation
M_ϕ	Peak growth type
M_C	MPB function rotation period
n_c	Number of landscape changes
n_{fe}	Number of function evaluations
n_p	Number of peaks
n_s	Total number of entities in the population
n_x	Number of dimensions of the objective function
$N(0,1)$	A normal distribution with a mean of 0 and standard deviation of 1
P	Population size
$P_{\mathbf{BOG}}(t)$	Best-of-generation fitness
P_{CMF}	Collective mean fitness
P_{CME}	Collective mean error
P_{OF}	On-line performance
P_{MOE}	Modified off-line error
P_{MOF}	Modified off-line performance
P_{ABEBC}	Average best error before change
P_{LBEBC}	Lowest best error before change
P_{HBEBC}	Highest best error before change

$P_{RE}(t)$	Optimization accuracy
$P_{WA}(t)$	Window accuracy
P_{MTE}	Mean tracking error
$P_{D_{min}}$	Average minimum distance of population to optimum
P_{norm}	Normalized scores
$\mathbf{q}(t)$	Search direction vector
Q_i	Charge of particle i
R_c	Core radius of CPSO and APSO
r_{cloud}	Radius of QPSO quantum particle
R_p	Perception limit of CPSO and APSO
$R_{ab}(\theta)$	Rotation matrix from axis a to axis b through angle θ
$\mathbf{p}_r$	Random vector normalized to length s
s	Spatial severity of the MPB
$\mathbf{s}_v$	The MPB shift vector
Θ	Minimum fitness level required for the *satificability* measure
t	time, time step counter
t_c	time of landscape change c
T	The total number of iterations
$\mathbf{v}_i(t)$	Velocity of particle i
$w_p(t)$	Width of MPB peak p at time t
w_s	Width severity of the MPB
$\boldsymbol{\omega}(t)$	Time dependent control parameters of f at time t
$\mathbf{x}$	The n_x dimensional position vector representing a solution
x_{ij}^t	The j-th dimension of the i-th candidate solution at time step t
$\mathbf{x}_{best}^t$	The best position vector at time t
$\mathbf{x}_{best}^{t_c}$	The best position vector at time t_c just before an environment change
$\mathbf{x}^e$	The position vector evaluated during function evaluation e
$\mathbf{x}^*$	The global optimum
$\mathbf{x}_{opt}^t$	The global optimum at time t

$\hat{\mathbf{y}}_i$	Neighborhood best position of particle i
$\mathbf{y}_i$	Personal best position of particle i

Chapter 3: Selection Hyper-heuristics

E	Parent population of all entities in HMHH
$E^*(t) \subseteq E$	Entities in E that are to be assigned new heuristics at time t
e_j	A single entity in E
$f_j(t)$	Fitness of any entity e_j at time t
f_k	Optimization problem in set of problems F
H	Pool of heuristics
$\mathcal{H}(t)$	Heuristic space diversity measure, by Grobler and Engelbrecht [76]
h_i	A single heuristic i
k	Algorithm iterations before a heuristic change occurs
$s_m(t) \subseteq E$	Entities that have been uniquely assigned to heuristic h_m at time t
$\mu_m(t)$	The mean fitness of all entities in the subset s_m
$n_m(t) = \lvert s_m(t) \rvert$	Number of entities assigned to heuristic h_m at time t
$n_h = \lvert H \rvert$	Total number of heuristics in H
$n_s = \lvert E \rvert$	Total number of entities in E
$\mathcal{N}(t)$	Heuristic space diversity based on Shannon entropy
$\phi_m(t)$	Mean change in fitness of entities $e_j \in s_m$
$\mathcal{P}_m(t)$	The performance of h_m at time t for CPE
$Q_{\delta_m}(t)$	Problem space measure that serves as feedback for the selection operator
$\rho_m(t)$	Pheromone concentration og heuristic h_m at time t
$\Upsilon_m(t)$	The maximum fitness of all entities in the subset s_m
$\varsigma(e_j, Q_{\delta_m}(t))$	The selection operator that assigns entity e_j to a heuristic
$\Theta(t)$	Probabilities of assigning an entity to each heuristic at time t
$\chi_m(t)$	Frequency of improvement for heuristic h_m

Chapter 4: Estimation of Control Group Performance Baselines

β	Scaling factor
$\mathbf{b}$	Vector of n_c performance measures values
$\mathbf{d}$	Vector of perfect performance values
C_r	Crossover rate
M_r	Mutation rate
$\mathcal{M}$	Mutation strategy
n_c	Number of search landscapes
n_v	Number of difference vectors for DE
$\mathbf{o}_i$	Offspring i
P_r	Relative error distance
$P_{r,L1}$	Relative error distance using the L1 norm
Q	Charge value for charged PSO particles
R_r	Replacement rate
$\mathbf{r}_a$	Rejection vector of a from the vector $\mathbf{d}$
R_c	Core limit of CPSO
R_p	Perception limit of CPSO
$r_{\mathbf{cloud}}$	Quantum radius

Chapter 6: Performance and Behavior Analysis of Selection Hyper-heuristics

δ	Entity reassignment frequency
φ	Entity reassignment volume
$\upsilon(t)$	Number of entities that are reassigned to new heuristics at time t
E^{**}	Subset of entities in E^* that are reassigned to new heuristics

Appendix D

Derived Publications

The following publications were derived from this study, and have been accepted or are under review.

- Stéfan A.G. van der Stockt and Andries P. Engelbrecht, Analysis of hyper-heuristic performance in different dynamic environments, *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, Orlando, FL, 2014, 1–8

- Stéfan A.G. van der Stockt and Andries P. Engelbrecht, Analysis of global information sharing in hyper-heuristics for different dynamic environments, *Proceedings of the 2015 IEEE Congress on Evolutionary Computation*, Sendai, Japan, 2015, 822–829

- Stéfan A.G. van der Stockt and Andries P. Engelbrecht, Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization. *Swarm and Evolutionary Computation*, 43, 127–146, 2018.

- Stéfan A.G. van der Stockt, Andries P. Engelbrecht and Christopher W. Cleghorn, A Novel Performance Metric and Analysis Method for Continuous Dynamic Optimization, *IEEE Transactions on Evolutionary Computation*, 1–10, 2020, (Under review)

- Stéfan A.G. van der Stockt, Andries P. Engelbrecht and Christopher W. Cleghorn, An Improved Heuristic Space Diversity Measure for Population-based Hyper-heuris-

tics, *Proceedings of the 2020 IEEE Congress on Evolutionary Computation*, 1–10, IEEE Press, Piscataway, NJ, 2020 (Accepted)

- Stéfan A.G. van der Stockt, Andries P. Engelbrecht and Christopher W. Cleghorn, Extending the Hyper-meta-heuristic Framework with Neighborhood Topologies and Heuristic Change Triggering Mechanisms, *Swarm and Evolutionary Computation*, 1–13, 2020 (Under review)

- Stéfan A.G. van der Stockt, Andries P. Engelbrecht and Christopher W. Cleghorn, Characterizing Popoluation-based Hyper-heuristic Entity Reassignment Behavior for Dynamic Environments, *Swarm and Evolutionary Computation*, 1–13 2020 (Under review)