Characterising Continuous Optimisation Problems for Particle Swarm Optimisation Performance Prediction

by

Katherine Mary Malan

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

January 2014

Publication data:

Katherine Mary Malan. Characterising Continuous Optimisation Problems for Particle Swarm Optimisation Performance Prediction. Doctoral thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, January 2014.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

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

http://upetd.up.ac.za/UPeTD.htm

Characterising Continuous Optimisation Problems for Particle Swarm Optimisation Performance Prediction

by

Katherine Mary Malan
E-mail: kmalan@cs.up.ac.za

Abstract

Real-world optimisation problems are often very complex. Population-based metaheuristics, such as evolutionary algorithms and particle swarm optimisation (PSO) algorithms, have been successful in solving many of these problems, but it is well known that they sometimes fail. Over the last few decades the focus of research in the field has been largely on the algorithmic side with relatively little attention being paid to the study of the problems. Questions such as 'Which algorithm will most accurately solve my problem?' or 'Which algorithm will most quickly produce a reasonable answer to my problem?' remain unanswered.

This thesis contributes to the understanding of optimisation problems and what makes them hard for algorithms, in particular PSO algorithms. Fitness landscape analysis techniques are developed to characterise continuous optimisation problems and it is shown that this characterisation can be used to predict PSO failure. An essential feature of this approach is that multiple problem characteristics are analysed together, moving away from the idea of a single measure of problem hardness. The resulting prediction models not only lead to a better understanding of the algorithms themselves, but also takes the field a step closer towards the goal of informed decision-making where the most appropriate algorithm is chosen to solve any new complex problem.

Keywords: Fitness landscape analysis, problem hardness measures, particle swarm optimisation.

Supervisor	**:** Prof. Andries P. Engelbrecht
Department	**:** Department of Computer Science
Degree	**:** Philosophiae Doctor

“The real voyage of discovery consists not in seeking new landscapes but in having new eyes.”

– Marcel Proust

Acknowledgements

I would like to acknowledge the following people who helped me through the various stages of this PhD:

- My supervisor, Professor Andries Engelbrecht, for his sharp insights, clear guidance, for the countless benefits that come from being part of a research group with strong leadership, and his support and encouragement of my career as an academic;
- All the members of CIRG, current and past, for their feedback, help and for making it fun;
- My husband, Greyling, for sharing the highs and lows (of my PhD and his), for his sense of humour and for helping to maintain a home environment where our beautiful daughters could thrive, blissfully unaware of our perpetual lack of sleep.

Contents

List of Figures vii

List of Algorithms ix

List of Tables x

1 Introduction 1

1.1 The algorithm selection problem 3

1.2 What makes an optimisation problem hard? 6

1.3 Existing approaches to fitness landscape analysis 8

1.4 Research objectives 9

1.4.1 Objective 1: Survey of existing techniques 9

1.4.2 Objective 2: Develop a problem characteriser 10

1.4.3 Objective 3: Develop prediction models for PSO performance . . 12

1.5 Contributions 13

1.6 Thesis outline 15

2 Background: Analysis of Fitness Landscapes 17

2.1 Introduction 17

2.2 Fitness landscapes 18

2.2.1 Wright's fitness landscape 18

2.2.2 Fitness landscape formalisations 19

2.2.3 One function, many landscapes 21

2.3 Features of fitness functions and landscapes 22

2.3.1 Degree of variable interdependency (including epistasis) 22
2.3.2 Noise 23
2.3.3 Fitness distribution 24
2.3.4 Fitness distribution in search space 24
2.3.5 Modality and the landscape structure of optima 24
2.3.6 Information to guide search and deception 25
2.3.7 Global landscape structure (funnels) 26
2.3.8 Ruggedness and smoothness 26
2.3.9 Neutrality 27
2.3.10 Symmetry 28
2.3.11 Evolvability/Searchability 29
2.3.12 Discussion 29
2.4 Measures and techniques for analysing fitness landscapes 30
2.4.1 Types of measures not included in this survey 30
2.4.2 Introduction to the survey 32
2.5 Discussion of survey of fitness landscape analysis techniques 47
2.5.1 The focus of techniques 47
2.5.2 Search independence 50
2.5.3 Further proposed work 51
2.6 The link between problems and algorithms 52
2.7 Summary 53

3 Benchmarks, Algorithms and Performance 55

3.1 Benchmarks 56
3.2 PSO algorithms 57
3.2.1 Traditional global best PSO 58
3.2.2 Cognitive PSO 59
3.2.3 Social PSO 59
3.2.4 Local best PSO 59
3.2.5 Asynchronous global best PSO 60
3.2.6 Bare bones PSO 60
3.2.7 Modified bare bones PSO 61

3.3 Measuring algorithm performance 61
3.3.1 Existing approaches to measuring performance 62
3.3.2 Proposed performance metrics 63
3.3.3 Performance metrics applied to PSO 68
3.3.4 Performance classes 70
3.4 Summary 71

4 Ruggedness, Funnels and Gradients 72
4.1 Introduction 72
4.2 Random walk algorithms 74
4.2.1 Simple random walk 75
4.2.2 Progressive random walk 77
4.2.3 Testing coverage of walks 82
4.3 Simple benchmark problems 86
4.4 Ruggedness measures 86
4.4.1 Measuring ruggedness using entropy 86
4.4.2 Modifications for continuous landscapes 90
4.4.3 Testing entropic measure on one-dimensional functions 91
4.4.4 Proposed single measure of ruggedness 94
4.4.5 The influence of the step bound parameter on FEM 95
4.5 Estimating the presence of funnels 98
4.5.1 Dispersion metric 98
4.5.2 Proposed dispersion metric 99
4.5.3 Testing dispersion metric 101
4.6 Gradient measures 102
4.6.1 Proposed gradient measures 102
4.6.2 Manhattan progressive random walk 103
4.6.3 Normalising gradient measures 105
4.6.4 Testing gradient measures on one-dimensional problems 106
4.7 Linking to PSO performance on higher dimensional problems 111
4.7.1 Experimental setup 112
4.7.2 Results 114

4.7.3 The link to PSO performance 120
4.8 Summary 122

5 Searchability 125
5.1 Existing techniques for measuring evolvability 126
5.1.1 Fitness distance correlation 127
5.1.2 Information landscape measures 128
5.1.3 Fitness cloud 130
5.1.4 Negative slope coefficient 131
5.2 Proposed general searchability measures 134
5.2.1 Fitness distance correlation as a searchability measure 135
5.2.2 Information landscape negative searchability measure 135
5.2.3 Summary of proposed general searchability measures 137
5.3 Proposed PSO searchability measures 137
5.3.1 Determining neighbours for fitness clouds using PSO 138
5.3.2 Proposed measures based on fitness clouds 142
5.4 Experimentation on one-dimensional problems 143
5.4.1 Benchmark functions 144
5.4.2 Experimental setup 147
5.4.3 Results and discussion 147
5.4.4 Investigation into NSC measure for PSO updates 149
5.5 Linking to PSO performance on higher dimensional problems 156
5.5.1 Benchmark problems and expected results 157
5.5.2 Experimental setup 157
5.5.3 Searchability measures results 158
5.5.4 PSO performance and searchability measures 163
5.5.5 Discussion 168
5.6 Summary 169

6 Predicting PSO Performance 170
6.1 Introduction 170
6.2 Dataset generation 172

6.3 Predicting algorithm failure 175
6.3.1 Decision tree induction 175
6.3.2 Fuzzy rule extraction 179
6.3.3 Discussion 180
6.3.4 Summary 183
6.4 Predicting algorithm performance 183
6.5 Summary 188

7 Conclusions 191
7.1 Summary of conclusions 191
7.2 Future work 193
7.2.1 Validating failure prediction models with other functions 194
7.2.2 Predicting PSO performance with a larger dataset 194
7.2.3 Further analysis of proposed fitness landscape measures 194
7.2.4 Measuring other landscape features 195
7.2.5 Further adaptations of existing techniques 195
7.2.6 Automating algorithm selection 195
7.2.7 Applying techniques to real-world problems 196

Bibliography 197

A Benchmark Functions 218

B Proposed Fitness Landscape Metrics 225

C Training and Testing Datasets 229

D Acronyms 236

E Symbols 237
E.1 Chapter 1 237
E.2 Chapter 3 237
E.3 Chapter 4 238
E.4 Chapter 5 239

F Derived Publications 240

Index 242

List of Figures

1.1 Framework for describing algorithm selection and performance prediction 5
1.2 Simple one-dimensional fitness landscapes 7
1.3 Problem characteriser black box 11

2.1 Adaptation of Wright's fitness landscape 20
2.2 Step benchmark function viewed at different resolutions 22
2.3 Two sample problem landscapes with different funnel characteristics . . . 27

3.1 Example one-dimensional minimisation benchmark functions. 64
3.2 Function used to scale fitness quality measure q to QMetric. 67

4.1 Landscape structure resulting in deception for PSO 73
4.2 Plots of simple random walk sample runs 76
4.3 Four starting zones for a two-dimensional search space. 78
4.4 One of the eight starting zones for a three-dimensional search space. . . . 79
4.5 Plots of sample progressive random walk runs 82
4.6 Sample random walks in three-dimensional space 83
4.7 Histograms of the distribution of samples in two-dimensional space . . . 84
4.8 One-dimensional benchmark functions for ruggedness and gradients . . . 88
4.9 $H(\varepsilon)$ over different values of ε for seven different 1-dimensional functions 93
4.10 How the value of $H(\varepsilon)$ can increase with an increase in ε 94
4.11 Plot of Mean FEM ruggedness for different step bounds 97
4.12 Decreasing fitness threshold and the effect on dispersion 99
4.13 Mean DM values for one-dimensional benchmarks 101
4.14 Plots of four sample Manhattan progressive random walks 104

4.15 Two different functions with the same landscape shapes 106
4.16 The effect of step size on the G$_{avg}$ measure 109
4.17 The effect of step size on the G$_{dev}$ measure 110
4.18 The effect of step size on the G_{max} measure 111
4.19 Scatter diagrams of the correlation between fitness landscape measures . 119
4.20 PSO performance plotted against ruggedness measures 121
4.21 PSO performance against ruggedness. 122
4.22 PSO performance plotted against dispersion metric and gradient measures 123
4.23 PSO performance against dispersion metric. 124
4.24 PSO performance against gradient measures. 124

5.1 Rastrigin function in 1 dimension with associated information landscape 129
5.2 Fitness cloud examples 131
5.3 Strategies for binning of fitness clouds. 133
5.4 Sample fitness clouds based on cognitive PSO updates 142
5.5 Fitness clouds of Hole-in-mountain function. 149
5.6 Fitness clouds of the Straight function 150
5.7 Rastrigin function with initial random sample of points. 151
5.8 Fitness clouds of 10 and 9 points 154
5.9 Fitness clouds of the Straight function where particles are kept in bounds 156
5.10 Scatter diagrams of searchability measures 163
5.11 Correlation between PSO performance and searchability measures 164
5.12 PSO performance plotted against searchability. 165
5.13 PSO performance and the FCI$_{\overline{\sigma}}$ measure 168

A.1 Graphical plots of benchmark functions from Table A.1. 222
A.2 Graphical plots of benchmark functions from Table A.1 continued. 223
A.3 Graphical plots of benchmark functions from Table A.1 continued. 224

List of Algorithms

4.1 Simple Random Walk Algorithm 77
4.2 Progressive random walk algorithm 81
4.3 Algorithm for computing FEM based on the search space of a problem . 95
4.4 Algorithm for computing dispersion metric (DM) 100
4.5 Manhattan progressive random walk algorithm 105
5.1 Algorithm for calculating the NSC measure based on a fitness cloud. . . . 134
5.2 Algorithm for computing the IL$_{ns}$ measure 137
5.3 Algorithm for constructing a fitness cloud using PSO updates. 140
5.4 Modified algorithm for constructing a fitness cloud using PSO updates. . 154

List of Tables

2.1 Techniques for characterising fitness functions and landscapes 35

2.2 Summary of some possible ways forward 49

3.1 Example benchmark problems with fixed accuracy levels. 66

3.2 Traditional gbest PSO results on sample benchmark problems. 69

4.1 Proposed starting zones of progressive random walks. 79

4.2 Quantifying coverage of a sample 85

4.3 One-dimensional benchmarks for ruggedness, gradients and funnels . . . 87

4.4 Classification and encoding of objects as neutral, rugged or smooth . . . 89

4.5 Mean FEM ruggedness using different step bounds 96

4.6 Proposed measure of ruggedness based on random walks 97

4.7 Proposed dispersion metric (DM) 100

4.8 Values of G_{avg} for one-dimensional benchmark functions 107

4.9 Values of G_{dev} for one-dimensional benchmark functions 108

4.10 Values of G_{max} for one-dimensional benchmark functions 108

4.11 Proposed gradient estimation measures 112

4.12 Landscape measures alongside PSO performance metrics. 115

5.1 Proposed general measures of searchability 138

5.2 Proposed measures of searchability based on fitness clouds 144

5.3 One-dimensional benchmarks with different searachability characteristics. 145

5.4 Values of searchability measures for one-dimensional problems 148

5.5 Fitness pairs with resulting bins and midpoints. 152

5.6 One fewer fitness pair and the effect on bin partitioning. 153

5.7 Values of the FCI and NSC measures based on revised fitness clouds. . . 155
5.8 Searchability measures alongside PSO performance 159
5.9 Additional proposed measure of searchability based on fitness clouds . . . 167

6.1 PSO failure prediction models for four PSO variations. 176
6.2 PSO failure predictions models for two PSO variations. 177
6.3 Summary statistics of the fitness landscape measures. 180
6.4 Fuzzy rules for predicting algorithm failure 181
6.5 Global best PSO performance prediction model 185
6.6 Cognitive PSO performance prediction model 185
6.7 Social PSO performance prediction model 186
6.8 Local best PSO performance prediction model 186
6.9 Asynchronous global best PSO performance prediction model 187
6.10 Barebones PSO performance prediction model 188
6.11 Modified barebones PSO performance prediction model 189

A.1 Benchmark Functions 219

B.1 Summary of proposed fitness landscape metrics 226

C.1 Training data set 230
C.2 Testing data set 234

Chapter 1

Introduction

In many ways we strive for optimality in our lives: we want what is best at the minimum cost. Optimisation is the name given to the mathematical discipline concerned with this striving for optimality. Given a set of variables representing a solution to a problem, optimisation involves finding the best solution: the values of the problem variables that result in the minimum or maximum of the objective of the problem. Designing a concert hall that maximises acoustics or an aircraft wing that minimises drag, finding a set of trading rules that maximises profit or a route through a telecommunications network that minimises delay are all examples of real-world optimisation problems.

Traditional mathematical optimisation techniques, such as gradient-based techniques, use the derivative of the objective function to determine the exact maximum or minimum point of a problem. Many optimisation problems are, however, not able to be solved using such traditional techniques. For example, multimodal problems where gradient information is not sufficient for finding the global optimum, problems with discontinuous search spaces that are non-differentiable, or black-box optimisation problems where there is no objective function in mathematical form to be differentiated. For these kinds of problems, often the only feasible alternative is to use approaches that find approximate solutions. Metaheuristics is the name given to such approximate optimisation techniques that are general, in that they can be applied to solve any optimisation problem.

There are many different approaches within the field of metaheuristics. Some metaheuristics perform local search, usually with extensions to enhance exploration of the search space, such as simulated annealing [22, 77], iterated local search [82] and tabu search [43, 44]. Other metaheuristics are population-based in that they work on a collection of solutions in parallel, such as evolutionary algorithms [5], ant colony optimisation algorithms [30] and PSO algorithms [75]. Many of these population-based metaheuristics are inspired by nature; the way that nature constructs elegant solutions to problems within extremely complex systems has inspired scientists to capture some of this 'magic' in simple forms within algorithms and the results are often surprisingly good. Talbi [152] provides a genealogy of metaheuristics as applied to optimisation from the 1940s to the 1990s containing no less than 24 different kinds of algorithms. Since then many more algorithms have been proposed from bio-inspired algorithms like firefly [185] and krill herd [40] algorithms to algorithms inspired by the interaction between magnetic particles [154] and the way musicians improvise (harmony search) [42].

Real-world optimisation problems frequently involve real-valued variables and there are many algorithms that were designed to work in these continuous spaces, such as PSO, differential evolution [120], evolution strategies [128, 138] and cuckoo search [186]. Many metaheuristics originally designed for discrete spaces have also been adapted to work in continuous environments such as real-coded genetic algorithm (GA)s [60], evolutionary programming [78], ant colony optimisation algorithms [11], estimation of distribution algorithms [1] and scatter search [59]. Given this plethora of algorithms, the challenge of choosing the most appropriate algorithm for solving a given problem can be a daunting task. A metaheuristic algorithm can easily be described in terms of its properties, such as the way in which search history is used, but due to stochastic elements, the result of these algorithms is not easily predicted. The properties of an algorithm therefore only have limited value in guiding the choice of an appropriate algorithm to solve a problem. Questions such as 'Which algorithm will most accurately solve my problem?' or 'Which algorithm will most quickly produce a reasonable answer to my problem?' remain unanswered. When the choice is between vastly different algorithms (such as a genetic algorithm or a particle swarm optimisation algorithm) there is even less guidance and the most common technique for choosing an appropriate algorithm is trial and error. If there existed one algorithm that out-performed all others in solving optimisation problems, then this 'super-algorithm' could be used in all cases. It is well known, however, that no such algorithm can exist as was proved by Wolpert and Macready with their famous 'No-Free-Lunch' theorems for search/optimisation [180, 181]. The emphasis is therefore not on finding the best optimisation algorithm in general, but on finding the most appropriate optimisation algorithm for solving a particular problem. This challenge is expressed aptly by Culberson [25]: *"The researcher trying to solve a problem is then placed in the unfortunate position of having to find a representation, operators and parameter settings to make a poorly understood system solve a poorly understood problem. In many cases he might be better served concentrating on the problem itself"*. This research focuses on ways of better understanding optimisation problems in the hope that this will guide practitioners and researchers in the use of appropriate algorithms or avoiding the use of inappropriate algorithms.

This introductory chapter starts in the following section by describing the algorithm selection problem: what is involved in finding the most appropriate algorithm to solve a problem in general. Given this framework, Section 1.2 introduces some of the factors that can influence the difficulty of optimisation problems, while Section 1.3 summarises some of the existing approaches to analysing problem difficulty. The overall objectives of the research and the main contributions are described in Sections 1.4 and 1.5, while the outline of the thesis is provided in Section 1.6.

1.1 The algorithm selection problem

The general problem of selecting an effective or good or best algorithm to solve a given problem was formulated by Rice in the 1970s [132]. One of the models described by Rice is the model where algorithm selection is based on features of the problem. This model has four main characteristics:

- A set of problem instances (problem space P),
- a set of algorithms for solving the problems (algorithm space A),
- measures for comparing the performance of algorithms on a particular problem (performance measure space Y), and
- measurable characteristics of the problem instances (feature space G).

The relationship between these components is illustrated in Figure 1.1 (this diagram is based on Rice's Figure 3 [132] but includes an additional mapping for performance prediction). A given algorithm $a \in A$ can be applied to a problem instance p to produce performance measure(s) $y(a(p))$. In a trial-and-error approach to finding the best algorithm to solve a problem, this process of applying algorithms to a problem is simply repeated until the best algorithm from a set of algorithms is found, based on the given performance measures. The algorithm selection problem, however, involves avoiding this trial-and-error approach by achieving the following:

- Feature extraction: devising a mapping from problem space to feature space, so that any problem instance p can be characterised by features $g(p)$; and
- Algorithm selection: devising a mapping from problem feature space to algorithm space, so that a given problem p, with extracted features $g(p)$, can be matched to the most appropriate algorithm a, such that performance $y(a(p))$ is maximised.

Figure 1.1 also illustrates the related problem of performance prediction, which involves predicting the performance $y(a(p))$ of a given algorithm a applied to problem p based on extracted features $g(p)$. If a solution to the algorithm selection / performance prediction problem is found, it becomes possible to take an unseen optimisation problem, extract its features, and from these features select the most appropriate metaheuristic algorithm from a subset of metaheuristics for solving the problem (the algorithm selection problem) or predict the performance of a given metaheuristic algorithm on the problem (the performance prediction problem).

Rice's model [132] aims to find the single most appropriate algorithm for solving a given problem. Peng *et al.* [116] emphasise the inherent risk associated with algorithm selection and propose a framework, called population-based algorithm portfolios, where the problem is distributed among multiple different algorithms. This introduces a different perspective, but does not remove the essential problem of algorithm selection. Instead, the problem changes to one of selecting an appropriate set of algorithms to make up the portfolio.

Smith-Miles [143] used Rice's model [132] to address the algorithm selection problem for a subset of combinatorial optimisation problems, namely quadratic assignment prob-

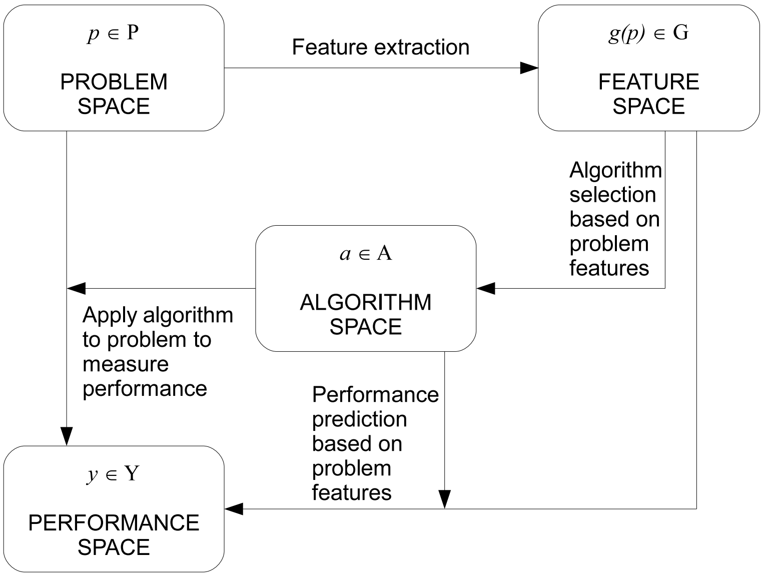

$p \in$ P
PROBLEM SPACE
Feature extraction
$g(p) \in$ G
FEATURE SPACE
Algorithm selection based on problem features
$a \in$ A
ALGORITHM SPACE
Apply algorithm to problem to measure performance
Performance prediction based on problem features
$y \in$ Y
PERFORMANCE SPACE

Figure 1.1: A framework for describing the general problems of algorithm selection and performance prediction based on problem features (based Rice's model [132]).

lems. In that study, 28 instances of the problem were used with three metaheuristics (tabu search, iterated local search and min-max ant system). The features of the problem were a combination of measures quantifying the size of the quadratic assignment problem with fitness-distance metrics based on local search runs (requiring knowledge of the global optima). A neural network was used to solve the problem of mapping problem features to performance measures. Although restricted to a specific class of optimisation problems, the study by Smith-Miles demonstrated the potential relevance of using such an approach. This thesis proposes how Rice's model [132] can be applied to continuous optimisation problems and PSO algorithms, where the features of the problem are based on the analysis of fitness landscapes.

The following are prerequisites to solving the algorithm selection problem [144] in general:

1. A large number of problem instances with different levels of difficulty;

2. A large number of different algorithms for solving these problem instances;

3. Metrics for evaluating the performance of algorithms; and

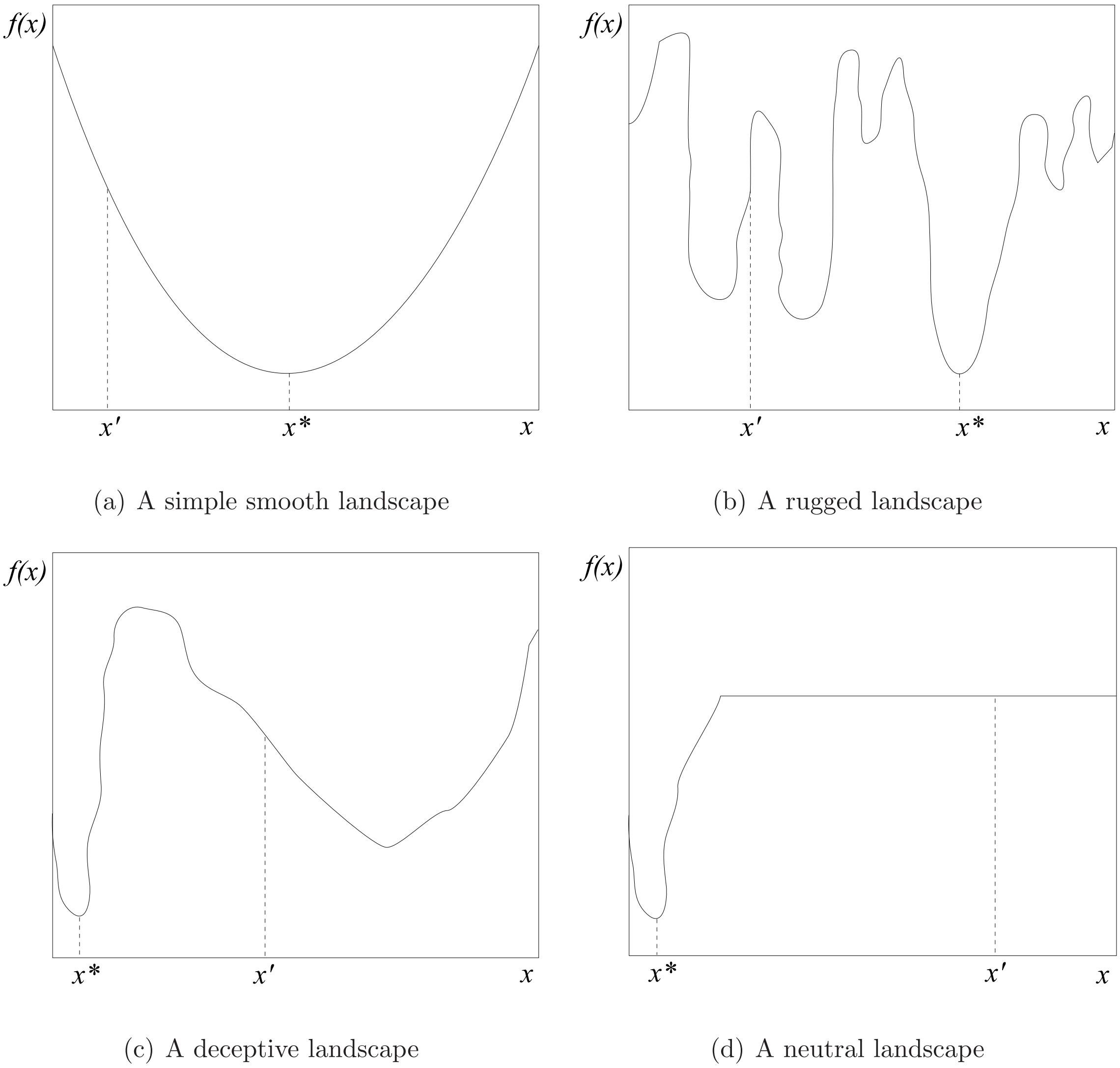
4. The existence of features that can be used to suitably characterise the properties of problems.

Considering the domain of real-encoded optimisation problems, item 1 above is addressed with the extensive range of benchmark problems in the literature. Some of these benchmarks are described in Section 3.1 of the thesis. Focussing on PSO algorithms, item 2 is met by the many different variations of the basic PSO algorithm, displaying different search behaviours. The variations that are used in this study are described in Section 3.2. Considering item 3 above, although there are metrics that are commonly used for evaluating the performance of optimisation algorithms, there are problems with using these metrics across the domain of different problems. This issue is addressed in Section 3.3 of the thesis. The bulk of this research is focussed on item 4 above with Chapter 2 providing the background to the features of optimisation problems and a survey of existing techniques for measuring features and Chapters 4 and 5 proposing techniques for approximating some of these features for continuous problems. Chapter 6 then attempts to find the link between problem features and performance using data mining on a dataset of features extracted from a selection of benchmark problems and solved using different PSO algorithms.

1.2 What makes an optimisation problem hard?

Although it is now known that a generic answer to the above question is unlikely to exist [55], many have tried to predict problem difficulty based on the features of a problem. This section discusses some of the properties or features of optimisation problems that could influence the degree of difficulty in solving them. Consider the fitness[1] landscapes of simple one-dimensional continuous problems as illustrated in Figure 1.2, where x' is a candidate solution found by a search process and x^* is the global optimum solution. A search algorithm would use information from the search space to decide how to proceed – information such as the gradient of the fitness function, or fitness values of solutions in the neighbourhood of x', or a whole population of alternative solutions with associated

[1] The term fitness is used to describe the objective of the function and is not restricted to the evolutionary sense of 'survival of the fittest'.



(a) A simple smooth landscape

(b) A rugged landscape

(c) A deceptive landscape

(d) A neutral landscape

Figure 1.2: Simple one-dimensional fitness landscapes to be minimised where x' is a candidate solution to the problem, $f(x')$ the fitness of solution x' and x^* the optimal solution.

fitness values. Whichever approach is used, a simple function such as the one illustrated in Figure 1.2(a), clearly provides good information to guide search towards the global optimum.

In contrast, the rugged landscape (landscapes are described in Section 2.2) in Figure 1.2(b) provides very little useful information to guide search towards the global optimum.

In addition, the vertical gradients in places (e.g. at point x') make it difficult for some search algorithms, either because the gradient is not defined or because the same solution can have multiple fitness values. Ruggedness clearly affects problem difficulty and many studies on problem hardness have focussed on ruggedness as the main determining factor [80, 92, 149]. The ruggedness of a fitness landscape is, however, not the only factor affecting problem hardness. Consider for example the problem in Figure 1.2(c). This landscape would not be regarded as rugged compared with the landscape in Figure 1.2(b), but presents misleading information for a local search algorithm. Starting at position x', a local search algorithm would typically be guided away from the global optimum at x^*. Problems such as these that present an algorithm with misleading information are known as deceptive problems and many studies on problem hardness have focussed on deception as the main determining factor [15, 45, 46, 66]. Neutrality is yet another factor that can have an influence on problem difficulty [114, 165, 167, 172, 174]. This phenomenon is illustrated in Figure 1.2(d), where there is a lack of information around the candidate solution x' for guiding search towards the global optimum. These and other characteristics that affect problem difficulty are explored further in Section 2.3.

1.3 Existing approaches to fitness landscape analysis

Much of the research in fitness landscape analysis over the last few decades has been focussed on finding measures for predicting general problem hardness. These attempts have not been very successful and the general agreement in the literature seems to be that no satisfactory problem difficulty measure for search heuristics has been found [50, 55, 65]. Although many proposed techniques have therefore been 'shot down' as difficulty measures, it is a premise of this study that such techniques can still be useful if viewed as measures of particular problem characteristics, rather than of problem difficulty. This view is supported by Müller and Sbalzarini when they used one proposed difficulty measure (fitness distance correlation) to characterise the CEC 2005 benchmark suite [150]. They concluded that *'fitness-distance analysis can only provide one out of several useful landscape descriptors that need to be combined in order to form discriminative landscape fingerprints.'* With a shift in focus away from predicting hardness, each technique has its own place within a toolbox of techniques for characterising problems. For example, the measure called negative slope coefficient [162, 164], originally proposed as a difficulty measure, can be used as a measure of evolvability in combination with other measures of ruggedness, neutrality and gradients.

There are other perceived problems with existing techniques. For example, some techniques assume knowledge of the global optimum and some assume a discrete representation and are not defined for continuous problems. Part of this research aims to investigate existing techniques and to assess their suitability as measures of characteristics of continuous optimisation problems. Although some measures are unsuitable for continuous problems, others may require adaptations to be of practical use in characterising problems. In addition, new measures are proposed where appropriate. In this way, a whole host of characteristics can be analysed together to broadly characterise a problem.

1.4 Research objectives

There are three main objectives of this research as described in the subsections below.

1.4.1 Objective 1: Survey of existing techniques for characterising optimisation problems

Despite numerous studies on fitness landscape analysis and a large number of developed techniques, very few techniques are used in practice. This could be because fitness landscape analysis in itself can be complex. With the aim of using a wide range of fitness landscape techniques, the objective of this first part of the study was to describe existing techniques and highlight the attributes important for practical implementation. This objective is addressed in Chapter 2.

1.4.2 Objective 2: Develop a characteriser for continuous optimisation problems

Given an understanding of existing techniques for characterising problems, the second objective of this research was to develop a 'problem characteriser' that can take as input a fitness function of a real-valued optimisation problem and produce as output a number of characteristics or features of the problem. This objective is illustrated in Figure 1.3. The aim was not to develop exact or complete techniques, but rather approximate measures for partly characterising a problem, with the focus on the practical use of the techniques.

For the purposes of this study, the scope of the problems under investigation is limited to optimisation problems that are static, bound-constrained, multivariate and continuous (real-encoded) and it is assumed that problems are to be minimized. In general such a problem can be defined as:

$$\min f(\mathbf{x}),\ \ f : \mathbb{R}^n \to \mathbb{R},\ \ \mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^n$$

where $\mathbf{x}$ is an n-dimensional candidate solution vector and $\mathcal{S}$ defines the feasible subregion of $\mathbb{R}^n$ as defined by the domains of the variables within $\mathbf{x}$. For the purposes of this study, it is assumed that $\mathcal{S}$ is defined by simple boundary constraints, which are the same for all components of the solution vector; that is,

$$x^{min} \leq x_i \leq x^{max} \quad \forall \mathbf{x} \in \mathcal{S},\ 1 \leq i \leq n$$

Each technique within the problem characteriser should have the properties as outlined below.

Properties of measures for characterising problems

1. Each technique should measure characteristics that are likely to correlate with performance of algorithms. Without some correlation, a mapping from feature space to performance space (as illustrated in Figure 1.1) will not be achievable.

2. For the technique to be useful on unseen problems, it is assumed that there is no information on the nature of the problem beforehand other than the fitness function and the domains of the variables of the problem. For example, a technique that requires knowledge of the global optima would not be appropriate.

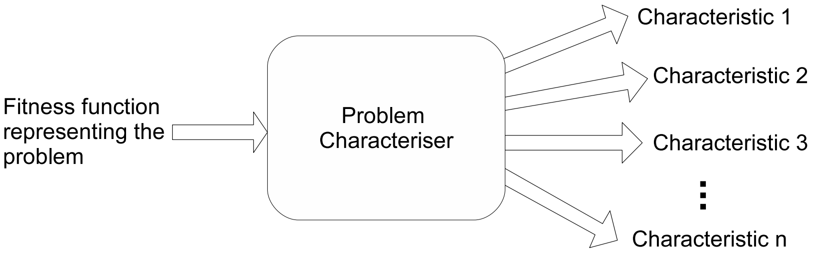


Figure 1.3: One of the aims of the project is to develop a “Problem Characteriser”: software that can take as input a real-encoded fitness function and produce as output a number of approximate characteristics of fitness landscapes.

3. Each technique should be analytical and result in numerical output values to facilitate data mining of generated data.

4. The computational work required in executing the technique should be significantly less than the computational work required to solve the problem using a typical search algorithm. In other words, characterising a problem should be less computationally intensive than solving the problem with multiple algorithms using a trial and error approach.

Although requirement 4 above states that the numerical effort of probing and characterising a problem in multiple ways should be significantly less than the numerical effort in using a trial and error approach with multiple algorithms, one could argue that this is not an essential feature. A trial and error approach to solving an unknown problem has no guarantee of producing a good solution to the problem. On the other hand, characterising a problem should lead to a deeper understanding of the problem and better choices of algorithms and therefore have an increased chance of producing a solution of higher quality than the uninformed application of multiple search algorithms.

Sub-objective 2.1: Adapting existing techniques to develop new measures

To meet the objective of developing a characteriser for continuous optimisation problems, a sub-objective was to investigate whether existing techniques could be adapted to either the context of continuous problems, or to meet the required properties of measures described above. Chapter 4 and 5 propose a number of adaptations to existing techniques and conduct preliminary investigations into the suitability of the proposed measures.

Sub-objective 2.2: Proposing new techniques

A further sub-objective was to propose new measures for problem features not previously considered in literature. Chapter 4 proposes new techniques for characterising the gradients of problems and Chapter 5 proposes a new technique for quantifying the unpredictability of fitness-improving solution updates.

1.4.3 Objective 3: Develop prediction models for PSO performance

Given a set of benchmark problems and a number of characteristics of those benchmarks, the benchmarks could then be solved using different PSO algorithms to produce performance metrics. The objective was to use data mining to develop prediction models based on this data. The resulting models could then be used to predict the performance of a given PSO algorithm on a new problem to decide whether the PSO algorithm would be a suitable approach to solving the problem or not. In addition, the models could be used to highlight what makes a problem difficult or easy for particular PSO algorithms.

Sub-objective 3.1: Failure prediction models for PSO

As a first step to solving main objective 3, the aim was to see if models could be developed to predict whether a particular PSO algorithm would fail or succeed to solve a particular problem.

Sub-objective 3.2: Performance prediction models for PSO

There are different levels of performance of algorithms for solving problems. An algorithm may fail outright to find a reasonable solution in a reasonable time even when the algorithm is tried many times, it may find a reasonable solution some of the time, or it may find a reasonable solution every time it is tried. How quickly an algorithm finds a solution is also a consideration. The second sub-objective was to see whether models could be developed to predict these different levels of success or outright failure of a particular PSO algorithm in solving a particular problem.

1.5 Contributions

The main contributions of this research are as follows:

1. A comprehensive survey of fitness landscape analysis techniques has been published [90] and appears in this thesis in Chapter 2. The survey highlights features of proposed techniques that are important for practical implementation and includes a new categorisation called the level of search independence. From the survey, adaptations of existing techniques are described as possible ways forward.

2. Two new normalised performance metrics are proposed called QMetric, for quantifying how close an algorithm came to finding an optimal solution to within a fixed accuracy level, and SSpeed, for quantifying how fast an algorithm was able to find the optimal solution. These measures are normalised to the range [0,1] and can be used as an absolute measure of performance for comparing any problem/algorithm combination with any other problem/algorithm combination. These metrics have been published as part of a chapter of a book on fitness landscapes [91].

3. A new algorithm for random walks in multidimensional continuous spaces has been developed, called a progressive random walk algorithm, described in Section 4.2.2. It is shown that this walk provides better coverage of a search space than a simple random walk algorithm. This approach is used to sample spaces when information on the neighbourhood structure is required and is used as the basis of measures of ruggedness and gradients described in Chapter 4.

4. An existing information theoretic technique for analysing the ruggedness of a fitness landscape with respect to neutrality [170, 171, 172] was adapted to work in continuous landscapes and to output a single measure of ruggedness. This proposed single measure of ruggedness was published as a conference paper [87] and is described in this thesis in Section 4.4.4. It is shown in Section 4.7 that the proposed

ruggedness measure, when used with a fairly large step bound (10% of the problem domain) has a strong correlation with the performance of a traditional global best (gbest) PSO algorithm for a number of benchmarks in higher dimensions (15 and 30 dimensions).

5. A computationally inexpensive technique for approximating gradients in multidimensional space has been developed, based on a variation of the progressive random walk, called a Manhattan progressive random walk. The technique is described in a conference paper [88] and it is shown in another paper that the measure can be used as a part-predictor of PSO failure for four different variations of the PSO algorithm [89]. The gradient estimation technique is described in Section 4.6 of this thesis.

6. Fitness clouds [173], an existing technique for visualising evolvability of genetic operators for discrete problems, has been adapted to visualise searchability of PSO updates for real-values problems (described in Section 5.3). Three new measures based on fitness clouds have been proposed for characterising problems with respect to cognitive and social PSO updates and the unpredictability of both updates. It is shown in Section 5.5 that the cognitive update and the unpredictability measures have a strong correlation to the performance of a traditional gbest PSO algorithm on a range of benchmark problems.

7. Negative slope coefficient (NSC) [162, 164], an existing measure of problem hardness based on fitness clouds, has been applied to fitness clouds for visualising the searchability of PSO updates (as described in point 6 above). It is shown in Section 5.4 that the NSC measure produces unpredictable results and does not show value as a measure of the ability of PSO updates to improve fitness. Possible reasons for this are investigated in Section 5.4.4.

8. Decision tree models have been produced for predicting failure for different variations on the PSO algorithm based on fitness landscape metrics. Based on a dataset of 24 benchmark functions generating 116 problem instances with different dimensions, decision trees were induced on 2/3 of the data and were tested

with the remaining 1/3 of the data. Training and testing accuracies of over 90% were achieved for the traditional gbest PSO, cognitive PSO, local best PSO, asynchronous gbest PSO and modified barebones PSO algorithms. These models are not only useful as predictors of algorithm failure, but also provide insight into the algorithms themselves, especially when expressed as fuzzy rules in terms of fitness landscape features. This contribution is described in Chapter 6 of the thesis.

1.6 Thesis outline

The remaining chapters of this thesis are as follows:

- **Chapter 2** provides a background to the analysis of fitness landscapes, including a summary of the kinds of features of fitness functions and landscapes that distinguish problems from each other, a survey of techniques for analysing fitness landscape, some possible ways forward, and a summary of related work on understanding the link between problems and algorithms.

- **Chapter 3** describes a number of benchmark problems for real-valued optimisation, describes the variations on the PSO algorithms that are used in this study and proposes a number of metrics for quantifying the performance of algorithms on problems.

- **Chapter 4** investigates a number of possible approaches to quantifying the fitness landscape features of ruggedness, presence of funnels and gradients.

- **Chapter 5** investigates a number of possible approaches to measuring searchability (or evolvability as it is better known), both in general terms and in terms of PSO updates.

- **Chapter 6** uses the proposed fitness landscape measures from Chapters 4 and 5 with the performance measures proposed in Chapter 3 to investigate the possibility of predicting algorithm performance based on approximate fitness landscape features.

- **Chapter 7** concludes the thesis by summarising the main points and providing suggestions for future work.

The appendices include the following:

- **Appendix A** defines 24 benchmark functions used in the study and provides graphical plots of the functions.
- **Appendix B** describes the ten proposed fitness landscape measures used in the study.
- **Appendix C** provides the training and testing datasets used as the basis for the final decision tree induction phase of the study.
- **Appendix D** provides a list of the important acronyms used in the thesis with their definitions.
- **Appendix E** lists and defines the mathematical symbols used in this work, categorised according to the relevant chapter in which they appear.
- **Appendix F** lists the publications derived from this work.

An index including some of the important terms is provided starting on page 242 of the text.

Chapter 2

Background: Analysis of Fitness Landscapes

2.1 Introduction

This chapter provides a background to the analysis of fitness landscapes and has been published as an article in *Information Sciences* [90].

The aims of the chapter are to, firstly, discuss characteristics of problems that could potentially make them hard to solve and, secondly, to provide an overview of existing techniques for analysing these problem characteristics. The chapter then discusses ways in which existing techniques can be adapted to be more practically useful. In addition, a number of recent advances to understanding the link between problems and algorithms are discussed.

The outline of the chapter is as follows: Section 2.2 starts with an overview of different views of fitness landscapes. Although the term 'fitness landscape' is used frequently in literature, it can have different meanings in different contexts and some of these interpretations are described. Section 2.3 provides a summary of different features of optimisation problems that could potentially affect the difficulty in solving the problem. In Section 2.4 a survey is provided of existing techniques to characterise optimisation problems from the 1980s to the present. Important features are highlighted such as the focus, the level of search independence, assumptions on which the technique is based, and the result produced by the technique. Section 2.5 provides suggestions on possible ways in which research in this area can move forward. Finally, Section 2.6 describes related work in understanding the link between problem understanding and algorithm performance.

2.2 Fitness landscapes

For most optimisation problems there is a fitness function[1] that reflects the objectives of the problem to be solved. (Problems that do not have a readily available fitness function are excluded from this study.) Potential solutions to a problem are compared based on their fitness values, which are determined using the fitness function. In some problem cases, the function is expressed so that the aim is to find the solution that maximises the fitness value and in other cases, the aim is to find the solution that minimises the fitness value.

There are many ways of analysing fitness functions, such as epistasis variance [27] and the density of states [136] and these are discussed further in Section 2.4. However, more interesting analyses can be performed, when a fitness function is extended into a fitness landscape, by introducing some form of topology onto the search space. Although the term ‘fitness landscape’ with its associated notions of ‘peaks’ and ‘valleys’ is widely used in many contexts and in academic writing, there is often a lack of understanding of what precisely is meant by the term. This section summarises some of the contributions towards understanding and formalising fitness landscapes.

2.2.1 Wright’s fitness landscape

Wright [182] introduced the notion of a fitness landscape (which he called a surface of selective values) for genetic evolution back in 1932, with further invited commentary on his seminal paper published 56 years later [183]. He proposed an abstract space where genotypes are packed, side by side, in a two-dimensional space in such a way

[1] Note that this study does not restrict the notion of fitness and fitness function to the meaning of fitness in the evolutionary sense, but rather to the broader notion of an objective and objective function to be optimised by an algorithm.

that each is surrounded by genotypes that differ by only one gene replacement. He used contour lines to indicate fitness values and in this way illustrated the peaks and valleys in a two dimensional diagram, as shown in Figure 2.1. Notice that the 'axes' of the diagram are not real axes, as there are no defined units or labels. In his own words, such a representation *"is useless for mathematical purposes"* [183]. Wright's aim with this representation was to provide an intuitive picture of evolutionary processes taking place in high dimensional space and not to provide any kind of formal model for analysis. Despite a lack of formal definition, this same basic fitness landscape metaphor with its associated 'valleys', 'peaks', 'ridges' and 'plateaus' has been used extensively within multiple disciplines to understand and explain complex systems.

2.2.2 Fitness landscape formalisations

An alternative, more formal view of a fitness landscape particular to search algorithms, is to define a landscape as a directed graph, where nodes correspond to solutions. Two nodes in the fitness landscape graph are neighbours if one solution can be reached from the other through a single step of a search operator (such as mutation or crossover in the case of a genetic algorithm). Jones [67] introduced such a model and argued for the view of *"one operator, one landscape"*, where each search operator defines its own fitness landscape. In his model, the fitness value (or 'height' in the fitness landscape metaphor) is indicated as a label attached to each node in the graph and probabilities of the step occurring are attached to edges of the graph.

In Jones' [67] study of fitness landscapes it is assumed that the landscape is discrete (or combinatorial). Stadler [148] provides a more general view of landscapes as consisting of three elements:

1. A set X of configurations (solutions to the problem),
2. a notion $\mathcal{X}$ of neighbourhood, nearness, distance, or accessibility on X, and
3. a fitness function $f : X \to \mathbb{R}$.

This description can be used in the case of both discrete and continuous landscapes. For example, in a discrete landscape, $\mathcal{X}$ could be described as a notion of neighbourhood

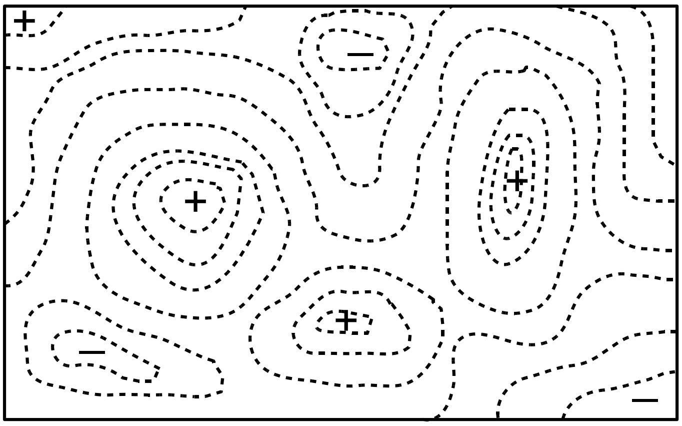


Figure 2.1: Adaptation of Wright's (1932) fitness landscape, which he called a two-dimensional 'surface of selective values'.

specified using a crossover operator, or using a more generic notion of neighbourhood, such as Hamming distance. For a continuous landscape, $\mathcal{X}$ can be described in terms of a distance metric, such as Euclidean distance or using some gradient-based search strategy for determining accessibility to a continuous subset of configurations.

In many studies, the term fitness landscape is used to refer to a problem encoding in combination with a fitness function (elements 1 and 3 in Stadler's description above). In these cases it is usually assumed that the notion of neighbourhood/distance is based on some 'natural' notion of order/distance. For example, in binary-encoded problems, Hamming distance is often assumed as the neighbourhood relationship: any two points are neighbours if their Hamming distance is 1. In real-encoded problems, Euclidean distance is usually assumed as the metric on which the fitness landscape is defined. In some representations, it is not as obvious to define neighbourhood. For example, when solutions are in the form of trees, there is no obvious way of deciding when two tree solutions are neighbours. In these cases, neighbourhood is often defined in terms of a particular search operator/strategy: two solutions are regarded as neighbours if it is possible to move from one to the other via a single application of the search operator.

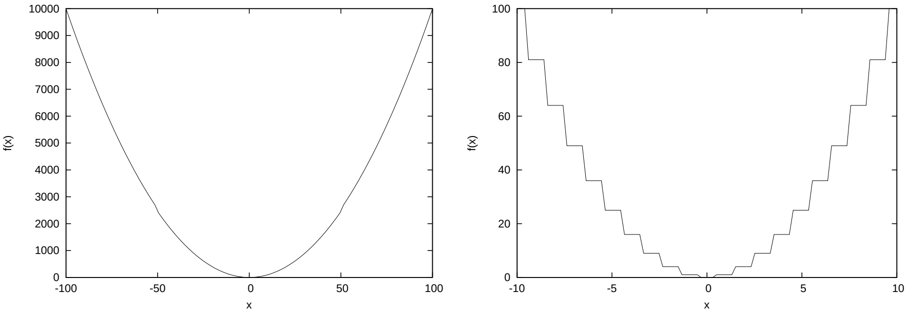
2.2.3 One function, many landscapes

Because a fitness landscape is defined using a particular notion of neighbourhood/distance, the same fitness function can generate many different fitness landscapes. For example, in Wright's [182] fitness landscape, a genotype is a neighbour of another genotype if they differ by a single gene. If instead, neighbourhood was defined based on a k-bit-flip mutation operator, a very different fitness landscape may result. The number of different possible fitness landscapes is dependent on the fitness distribution [13]. A constant fitness function, for example, has only one possible fitness landscape. The fitness landscape is therefore not a feature of the problem *per se*, but rather a feature of the encoding of the problem, the fitness function, and of the notion of neighbourhood/distance used to define the landscape.

In the same way, features such as ruggedness, deception or neutrality are not features of a fitness function, but rather features of a fitness landscape. Consider for example the Step benchmark function in D dimensions:

$$f(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2 \ . \tag{2.1}$$

The same function in one dimension is plotted in Figure 2.2 at different resolutions, rendering a seemingly smooth landscape in the case of Figure 2.2(a) and a landscape with high neutrality (many flat sections) in Figure 2.2(b). Similarly, a search process that samples the Step function by taking bigger step sizes may result in a smooth landscape, whereas a search process that samples the Step function by taking smaller steps may result in a landscape with high neutrality. Therefore, viewing a continuous problem at different levels of granularity, or exploring a multi-dimensional space in different ways, can lead to very different fitness landscapes and hence potentially different views on features such as ruggedness, deception or neutrality. Landscape theory serves as a reminder that other possibilities may exist, beyond the obvious ones, for defining landscapes and exploring and analysing optimisation problems.



(a) Larger domain

(b) Smaller domain

Figure 2.2: One-dimensional Step benchmark function viewed in at different resolutions rendering two different fitness landscapes.

2.3 Features of fitness functions and landscapes

This section summarises a number of features of optimisation problems that could influence the ability of algorithms to solve the problems. The features listed are not in any way exhaustive. There may be features not known or not mentioned here, which could influence the behaviour of optimisation algorithms. The purpose is to summarise those features that are commonly discussed in literature. Measuring or quantifying these features is not always straight-forward and this is discussed further in Section 2.4. The first three features (degree of variable interdependency, noise and fitness distribution) are features of the fitness function alone (without any defined fitness landscape). The remaining features are based on fitness landscapes. Although divided into separate subsections, many of these features are related to one another.

2.3.1 Degree of variable interdependency (including epistasis)

In genetics, epistasis refers to the degree of dependency between genes in a chromosome for expression [27]. If genes contribute independently to the overall fitness of the chromosome then the system has low epistasis. On the other hand, if the fitness contributions of genes depends on the values of the other genes, the system has high epistasis. In general, for optimisation problems, this characteristic can be referred to as the degree of interdependency between variables (also known as non-linear separability). When variables in an optimisation problem are dependent on each other, this means that it is impossible to tune one variable to find the optimal value independently of the others. For example, if different variables in a mathematical expression of a fitness function are separated by addition, then the variables contribute independently to the fitness. However, if different variables are combined in a term through multiplication, then these variables must cooperate in order to contribute to fitness; if either variable has a low value, then the product may be low even if the other variable has a high value. It is seldom as simple as in this example. For complex problems, the interactions between variables can take many different forms. Studies have shown that linearly separable functions are easier for genetic algorithms to solve than non-linearly separable functions [26, 137]. Naudts and Naudts [108] argue that it is the type of interaction (functions with first and second order dependencies) rather than the amount of higher order interaction that influences the difficulty of the problem for search algorithms. Caamaño *et al.* [20] propose dividing problems into three separability classes: linearly separable, non-linearly separable and non-separable functions. Measures for quantifying epistasis include epistasis variance [27] (Technique 5 in the survey), the site-wise optimisation measure [108], and bit-wise epistasis [36] (Technique 12 in the survey).

2.3.2 Noise

Noisy objective functions are common in many real-world optimisation problems. Levitan and Kauffman [79] studied the effect of noise on hill-climbing algorithms and found that although certain types and levels of noise had a negative effect on the ability of the algorithm to search well, small amounts of noise could help the algorithm to perform better than in the absence of noise. It is a common belief that evolutionary algorithms work well in noisy environments, but Beyer [10] has shown that this is not necessarily the case. A common way of reducing the effects of noise during search is to resample data points and average over a number of fitness evaluations [126]. Similarly, to detect noise in a fitness function, data points can be resampled. Some measure of difference (such as variance or standard deviation) could then be used to quantify the level of noise in multiple sampled data points.

2.3.3 Fitness distribution

A statistical analysis of fitness function values can provide some information on the problem at hand. For example, the distribution of fitness values (the frequency with which each fitness value occurs) can be used to provide a profile of the problem, as is done with the density of states technique [136] (Technique 9 in the survey) and Borenstein and Poli's analysis of the properties of a problem's fitness distribution [13]. In most cases the fitness distribution of a problem cannot be exactly determined and has to be estimated, based on some sampling and grouping strategy.

2.3.4 Fitness distribution in search space

Given a fitness landscape of a problem, a simple way of characterising the problem is to measure, in some way, how the fitness values are distributed across the search space. This differs from simple fitness value statistics, because the position of fitness values within the search space is taken into account. Techniques for quantifying fitness distribution layout in binary landscapes include the HDIL (Hamming Distance In a Level) and HDBL (Hamming Distance Between a Level) measures [8] (Technique 13 in the survey).

2.3.5 Modality and the landscape structure of optima

Unimodal functions have only one local optimum, which is also the global optimum. Multimodal functions have more than one local optimum. Horn and Goldberg [63] define a local optimum as a point or region (a set of interconnected points with equal fitness) with fitness function value greater than those of all its nearest neighbours. This definition would consider flat plateaus and ridges as single optima. Local optima are obstacles for local search algorithms in finding the global optimum because there is a lack of information in the neighbourhood to direct search out of the local optima.

Other than the number of optima, the distribution of basin sizes and the depth (or height) of the basins is a factor that may be more important in determining landscape difficulty [69]. Local optima with relatively small basins of attraction are called isolated. An extreme example of an isolated landscape could be a needle-in-a-haystack binary encoded maximisation problem, where the fitness value is 1 for one arbitrary bit string and

0 elsewhere. A less extreme example would be a landscape where the local optima have large basins of attraction and the global optima has a smaller basin of attraction. Rana [125] studied the effect of multimodality on GA performance and found that although the number of local optima did not always affect GA behaviour, highly fit local optima, particularly with large basins of attraction, did present a problem for GA search. Kinnear [76] found that in the case of genetic programming, landscape basin depths showed a good correlation with problem difficulty over a range of problems.

Techniques related to modality and landscape structure of optima include: Garnier and Kallel's [41] technique for estimating the number and distribution of local optima (extended for real-valued problems by Caamaño *et al.* [19]) and Merz's [94] escape rate measure for estimating the sizes of basins of local optima in the fitness landscapes of combinatorial problems. In addition, Ochoa *et al.* [112] have proposed a technique for compressing the essential landscape features for combinatorial optimisation problems into a graph called a local optima network. This graph-based model serves as a characterisation of the structure of a landscape and the distribution of local optima.

2.3.6 Information to guide search and deception

Some problems result in fitness landscapes that are structured in such a way that they guide search algorithms more easily towards the global optima. Both the quantity and quality of information available is important [14]. In other words, for an algorithm to perform well, the fitness landscape should not only provide sufficient information to guide the search, but the information should also guide the search in the right direction. The presence of misleading information is sometimes known as deception. Deception is clearly related to the landscape structure of optima. The positions of sub-optima in relation to the global optimum and the presence of isolation will have an effect on the level of deception. Deception only has meaning with reference to a particular search algorithm. A problem that is deceptive to a GA would not necessarily be deceptive to a PSO algorithm. Measures of deception include GA-deception [29, 45, 46] (Technique 1 in the survey) and the deceptiveness coefficient [66]. Xin *et al.* [184] studied the notion of deception for PSOs and concluded that the relative size of basins of attraction (local versus global) was the most important factor related to deception for PSOs.

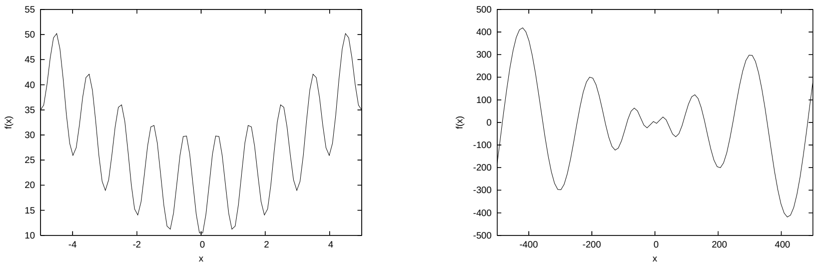
2.3.7 Global landscape structure (funnels)

A funnel in a landscape is a global basin shape that consists of clustered local optima [151]. Figure 2.3 shows two one-dimensional minimisation benchmark problems. Figure 2.3(a) shows the Rastrigin function as an example of a single-funnel landscape. Although Rastrigin is clearly multimodal, there is a distinct underlying unimodal structure, indicating the presence of a single funnel. Figure 2.3(b) illustrates the Schwefel 2.26 function, which is an example of a multi-funnel landscape. The exact number of funnels in Schwefel 2.26 would depend on the precise definition of a funnel. Multi-funnel landscapes can present problems for search, particularly in the case of algorithms that rely on local information, as they may become trapped in sub-optimal funnels [151, 184]. A technique for estimating the presence of funnels in a fitness landscape is Lunacek and Whitley's dispersion metric [84] (Technique 19 in the survey).

2.3.8 Ruggedness and smoothness

Ruggedness refers to the number and distribution of local optima. It therefore has to do with the level of variation in fitness values in a fitness landscape. If neighbouring points have very different fitness values, then the result is a rugged landscape. The opposite of a very rugged landscape would be a landscape with a single large basin/peak of attraction or a flat landscape with no features. In general, search algorithms struggle to optimise very rugged landscapes, because the algorithms can get trapped in local optima. Kauffman [70, 72] introduced a model of binary fitness landscapes with tuneable ruggedness, called NK landscapes, where the value of N specifies the number of variables and K can be set to determine the level of ruggedness. The NK landscapes have been used extensively in studies of ruggedness and the link to problem difficulty. Techniques for measuring ruggedness include adaptive walks [71], autocorrelation measures [92, 177] (Techniques 2 and 3 in the survey), correlation length [80] (Technique 4 in the survey), entropic measures [171, 172] (Technique 10 in the survey) and amplitude spectra [62] (Technique 11 in the survey).

A smooth landscape is one where neighbouring points have nearly the same fitness value [72]. Smoothness also relates to the size of the basins of attraction. A landscape



(a) One dimensional Rastrigin function: an example of a single-funnel landscape.

(b) One dimensional Schwefel 2.26 function: an example of a multi-funnel landscape.

Figure 2.3: Two sample minimisation benchmark problem landscapes with different funnel characteristics.

is smooth if the number of optima is low and the optima have large basins of attraction [172]. A technique for quantifying landscape smoothness is the second entropic measure by Vassilev *et al.* [171, 172] (Technique 10 in the survey).

2.3.9 Neutrality

Neutrality is present in a landscape when neighbouring points have equal fitness values. A discrete landscape is regarded as neutral if a substantial fraction of adjacent pairs of solutions are neutral [131]. A neutral landscape therefore does not imply a flat landscape (where the function is constant), but rather the presence of successive neutrality, which can manifest in features such as plateaus and ridges in a landscape. Neutrality can also feature in continuous fitness landscapes as regions of equal or nearly equal fitness (for an investigation into this topic of neutrality in continuous domains, see [64]). Neutrality is a feature which is often ignored, but can have a profound effect on the number and distribution of local optima [37] and on the success of search algorithms [6, 114, 142]. During search when a population moves through a neutral portion of a fitness landscape, this could be misinterpreted as convergence on a local optimum. Since the fitness values are not changing, it may seem as if the population is stagnating, when in fact the population is moving across a neutral area. In a study of neutral landscapes Beaudoin *et al.* [7] found that neutrality had a smoothing effect on problem difficulty in that adding neutrality to a deceptive landscape made the problem easier, whereas adding neutrality to an easy landscape made it harder (as measured by the fitness distance correlation difficulty metric [68]). Artificial test-beds for studying the effect of neutrality on search algorithms include the NKp family of binary landscapes [6] and the quantised classic functions [114] for continuous landscapes. Techniques that measure landscape neutrality include neutral walks [131] (Technique 14 in the survey) and neutral network analysis [165, 167] (Technique 20 in the survey). Verel *et al.* [175] show how local optima networks [112] can be extended to analyse the structure of neutral combinatorial fitness landscapes, but this approach currently requires a full enumeration of the search space.

2.3.10 Symmetry

Symmetry in a fitness function or landscape leads to multiple points with the same fitness values, so in some way partitions the search space into large equivalence classes [178]. There are many different forms of symmetry. For example, if a fitness landscape is symmetrical with respect to one of the axes this is known as axial bias. A fitness landscape is symmetrical with respect to an optimum if the fitness value of all points a set distance away from the optimum is the same regardless of the direction of the point. Some forms of symmetry are a feature of the fitness function alone. Van Hoyweghen and Naudts [161] define simple types of symmetry for discrete representations, such as symmetry on string positions (where a permutation on string positions results in no change to the fitness) and symmetry on the alphabet (e.g. spin-flip symmetry, where a binary string and the binary complement have the same fitness value). There are conflicting studies on the effect of symmetry on search. Whitley *et al.* [178] note several research findings where the presence of symmetry in functions results in failure for certain genetic algorithms. Naudts and Naudts [108] also show that the presence of symmetry can have a negative effect on the ability of a simple GA to converge. This could be due to the phenomenon where two dissimilar good (symmetrical) solutions, crossed over, result in inferior children. Other studies have shown that genetic algorithms show improved performance on landscapes with axial biases [26] and that a rotation of the coordinate system for such problems (resulting in the loss of symmetry) causes severe algorithmic performance loss [137].

2.3.11 Evolvability/Searchability

Evolvability can be loosely defined as the capacity to evolve [158]. Altenberg [2] describes evolvability with particular reference to genetic algorithms as the ability of a population to produce offspring that are fitter than their parents. Although the notion of evolvability is related to the algorithm's ability to evolve the population and is therefore primarily a performance measure of an algorithm, it can also be viewed as a characteristic of a fitness landscape in terms of a particular search operator/strategy. The evolvability of a fitness landscape is defined in this study as the ability of a given search process to move to a place in the landscape of better fitness and is henceforth referred to as *searchability*. Note that this definition broadens the scope of evolvability beyond evolutionary based algorithms to encompass any search process. Searchability is a characteristic of problems that only has meaning with reference to a particular search strategy. A problem that has high searchability in terms of one algorithm, may exhibit low searchability with reference to another algorithm. Fitness landscape analysis techniques that focus on evolvability include fitness evolvability portraits [142] (Technique 15 in the survey), fitness clouds [173] (Technique 16 in the survey), negative slope coefficient [162, 164] (Technique 17 in the survey), fitness-probability clouds [83] (Technique 21 in the survey) and accumulated escape probability [83] (Technique 22 in the survey).

2.3.12 Discussion

In the subsections above, a number of characteristics of fitness functions and landscapes were discussed. Many of these characteristics are related to each other. For example, modality and the structure of optima are clearly related to ruggedness, smoothness and neutrality of the landscape. Also, if a function has a high degree of variable interdependency, then this will probably affect the ruggedness of an associated landscape. Vassilev *et al.* [172] claim that the ruggedness, smoothness and neutrality alone can fully characterise a fitness landscape. Although this may be true, there could still be value in viewing a problem through a deception 'lens', or through a funnel 'lens' or through any other viewpoint that could shed light on the nature of the problem to be solved. The aim of this study is to work with a number of these characteristics together to form a more comprehensive view of the problem rather than limiting the focus to one viewpoint.

2.4 Measures and techniques for analysing fitness landscapes

For low dimensional problems, the associated fitness landscape could be visualised. A graphical representation could then give some indication of the features of the problem to be solved. Two problem landscapes could be compared in terms of ruggedness, deception, neutrality, etc. simply through visual inspection. In reality, however, problems are too complex to be visualised, so some other way of analysing problem characteristics is needed. The ideal would be to have a single numerical measure of difficulty for every problem. Given the issue with problem 'hardness' as described in Section 2.1, it is unrealistic to find such a single measure. Instead, it is proposed that problems should be characterised through multiple viewpoints and that this hopefully will provide sufficient insight into the problem so as to facilitate informed decisions regarding the approach used to solve the problem.

This section provides an overview of techniques used to characterise optimisation problems from the 1980s to the present. The techniques are summarised in Table 2.1, sorted in chronological order. Before presenting the table, the kinds of measures not included in this survey are described.

2.4.1 Types of measures not included in this survey

There are many reasons for characterising optimisation problems. Some measures are performed during execution of algorithms with the aim of adapting the algorithms on the fly. In other cases problems are studied and characterised to try to explain unexpected algorithmic behaviour in retrospect. In yet other cases the motivation is to divide problems into theoretical complexity classes. Jansen [65] distinguishes between two types of classifications of fitness functions: descriptive and analytical. A descriptive classification is one where classes of fitness functions are defined with some common property, whereas an analytical classification is a technique that takes a fitness function and produces a classifying attribute as output. For example, Naudts and Kallel [107] provide a precise definition for the class of site-wise optimisable fitness functions (a descriptive classification) and also define the site-wise optimisation measure as a measure of epistasis (an analytical classification). All techniques for classifying problems considered in this study are analytical. The aim is to obtain *a priori* information on the problem to help guide the choice of appropriate (or possibly not inappropriate) algorithms to solve the problem, in a less computationally-intensive way than actually solving the problem. In this section, three types of measures not included in this study are discussed. These are termed *theoretical measures*, *dynamic measures* and *retrospective measures*.

Theoretical measures

There are some estimators of problem complexity or difficulty that are theoretical in nature. Such measures, which cannot be practically implemented, are not discussed in this overview. An example of this is Kolmogorov complexity (KC). KC, also known as algorithmic information theory [49], is a measure of an object that relates to the complexity of the computer program required to produce that object and then halt. A discrete fitness function defined over a finite space can be described by a single binary string consisting of all possible output values of the function. The KC of this string is expected to capture the difficulty of the function [16]. Although this approach to using KC to quantify function complexity has been used extensively in theoretical studies and proofs, particularly in relation to the no-free-lunch theorems for search/optimisation, the KC of a problem cannot be computed [16], and is therefore not studied further.

Dynamic measures

Dynamic measures are those that are measured during execution of an optimisation algorithm and are typically used as a basis for adapting the search algorithm on the fly. Examples of such measures include the following:

- The correlation coefficient by Manderick *et al.* [92], which measures the correlation between two populations during execution,

- Riopka's average bit certainty measure [134], which is used to modify the behaviour of a GA relative to the landscape being searched,
- Generation Rate of Better Solutions (GRBS) measure by Waeselynck *et al.* [176] that monitors convergence, and
- Merz's escape rate measure [94] performed during the run of a memetic algorithm.

Retrospective measures

There are some measures of problems that involve the actual execution of an optimisation algorithm. By attempting to solve the problem (possibly using a number of approaches or iterations of an algorithm), characteristics of the problem can be deduced in retrospect. Examples of retrospective measures include the following:

- Kauffman and Levin's adaptive walks [71]: This is a measure for estimating the ruggedness of a landscape and involves determining the lengths of hill-climbing walks.
- Ochoa's consensus sequence plots [110], which involves running a GA on the problem multiple times with a decreasing mutation rate.
- Garnier and Kallel's [41] method for estimating the number and distribution of local optima, which involves performing a steepest ascent search from a random sample of starting positions.

The measures considered in more detail in this study are all computed *a priori* and although some measures are based on theories applicable to specific algorithms or on particular search operators, the purpose is to obtain information on the problem without actually executing a particular search algorithm.

2.4.2 Introduction to the survey

The aim of this survey was to obtain a better understanding of existing techniques for characterising optimisation problems. For a survey of techniques to be useful, distinguishing characteristics needed to be highlighted, but it was not clear what these distinguishing characteristics should be. Naudts and Kallel [107] distinguish between exact and approximate measures. A measure is *exact* if it is computed using all solutions in the search space, whereas an approximate measure is computed using a sample of the search space. He *et al.* [55] further distinguish between predictive and non-predictive measures. They define a predictive difficulty measure as one where the algorithm's worst-case running time is bounded by a polynomial in n (the problem size). Exact computation of many of the difficulty measures is in general exponential with respect to the problem size [65], so although many of the techniques were originally defined as exact measures, they are in practice used as approximate measures. Since the aim was not to divide techniques into classes, but rather to understand techniques to be of practical use, more descriptive distinguishing features are highlighted. These are described and motivated below and correspond to the attributes used in Table 2.1.

1. *Technique (with unique number)*: The first attribute gives the name of the technique and the reference to the authors that proposed the technique. The techniques in the table appear in chronological order by the year of the first reference to the technique. The reason for organising the survey in this way was to facilitate an understanding of how the techniques have evolved over the last two decades. Where a technique was adapted in different ways by subsequent studies by the same or different authors, citations to significant complementary research on the original technique are listed as "extensions". The year that the technique was first introduced in published form is also given.

2. *Focus*: The overall focus of the technique is given as the second attribute. This refers to what is measured or predicted by the technique. To explicitly tie each technique to the features discussed in Section 2.3, the relevant subsection discussing the feature is stated in parentheses.

3. *Search independence* (abbreviated to 'Search Indep.' in the table to save space): This descriptor refers to the level with which a technique is bound to a particular search algorithm. Four categories are used:

   (a) Complete: A technique for characterising a problem is regarded as having complete search independence when the technique is based on a fitness function alone and not on any notion of neighbourhood/nearness between solutions. In other words, there is no fitness landscape involved in the technique.

(b) High: A technique is regarded as having high search independence when it is based on some generic or neutral notion of neighbourhood/distance between solutions, such as Hamming distance or Euclidean distance, which defines the fitness landscape. An example of a technique with high search independence is one which is based on a random walk through the landscape, without any significant biased direction.

(c) Medium: A technique is regarded as having medium search independence when the sampling or analysis is based on, and therefore biased by, some theory or notion particular to a given search algorithm.

(d) Low: A technique is regarded as having low search independence if it is based on a sample generated by the actual execution of an optimisation algorithm. The survey does not include any techniques with low search independence, as these would be classified as retrospective measures (Section 2.4.1).

4. *Assumptions*: Where there are significant assumptions on which the technique is based, these are mentioned.

5. *Brief summary*: A brief summary of how the technique works is provided.

6. *Result*: There are many different forms of output produced by the techniques outlined in this survey. For example, some result in a single numerical output value, while others produce visual output in the form of scatterplots, graphs or charts. The *Result* attribute describes the output produced by the technique.

7. *Application*: Where applicable, some examples of applications of the technique in literature are provided.

8. *Critique*: Where significant, references are provided to literature that points out shortcomings or problems with the technique.

Table 2.1: Techniques for characterising fitness functions and landscapes

Technique 1:	**GA-deception** by Goldberg [45] with extensions [29, 46], 1987.
Focus:	Deception with respect to a GA (Section 2.3.6).
Search Indep.:	Medium: based on genetic operators and schemata, applicable to recombinative algorithms.
Assumptions:	Assumes knowledge of global optima. Assumes a binary representation.
Description:	A binary fitness function is expressed as a Walsh polynomial. The Walsh coefficients are then used to calculate schema average fitness values. A set of schema with relatively high fitness are determined and the effect of genetic operators on the fitness of the schema are analysed.
Result:	Decision on level of GA-deception (strictly deceptive, deceptive, simple, strictly simple).
Critique:	Grefenstette [48] presents counterexamples of functions that are highly deceptive and easy for GAs to optimize and functions that have no deception and are nearly impossible for GAs to optimize.
Technique 2:	**Autocorrelation function** by Weinberger [177] with extensions [61, 92], 1990.
Focus:	Ruggedness (Section 2.3.8).
Search Indep.:	High: based on random walks through a binary fitness landscape.
Assumptions:	Assumes a discrete landscape and that the landscape is statistically isotropic, meaning that the statistics of a random walk on a landscape will be the same, regardless of the starting position.
Description:	From a sequence of fitness values, obtained from a random walk through the fitness landscape, calculate the correlation with the same sequence of values a small distance away. Do this for all possible landscapes.
Result:	Plot of autocorrelation $\rho(s)$ against step size s (distance between sequences being correlated). The value of $\rho(s)$ is in the range $(-1, 1)$ where $\|\rho(s)\| = 1$ indicates maximal correlation and a value close to 0 indicates almost no correlation.

Continued on Next Page...

Table 2.1 – Continued

Application:	Autocorrelation is perhaps the most widely used fitness landscape analysis technique. Example applications include: travelling salesman problem [149], graph-partitioning problem [145], RNA folding [37], multidimensional knapsack problem [153], heuristic search for hyper-heuristics [111] and the problem of learning robot soccer goal-scoring behaviour [133].
Critique:	Kinnear [76] found that in the case of genetic programming, autocorrelation was a weak indicator of difficulty. Another criticism is that it does not consider neutrality in the landscape [37, 127, 172].
Technique 3:	**Correlation length** by Weinberger [177] with extensions [61, 92, 146], 1990.
Focus:	Ruggedness (Section 2.3.8).
Search Indep.:	High: based on random walks through a binary fitness landscape.
Assumptions:	As for Technique 2. Also assumes that the autocorrelation function is a decaying exponential.
Description:	Using the autocorrelation function $\rho(s)$ for step size s, calculate the correlation length using the formula: $\tau = -1/\ln(\rho(1))$.
Result:	A single value (the distance beyond which the majority of points become uncorrelated: a smaller value indicates a more rugged landscape).
Critique:	As for Technique 2.
Technique 4:	**Correlation length** by Lipsitch [80], 1991.
Focus:	Ruggedness (Section 2.3.8).
Search Indep.:	High: based on random walks through a binary fitness landscape.
Assumptions:	Assumes the problem has a binary representation.
Description:	Given 600 random initial points in the search space, calculate the standard correlation coefficient (c_i) between the fitness of points and the fitness of each of 30 i-mutant neighbours of the points. The correlation length is one less than the value of i at which c_i first becomes non-positive.
Result:	A single value (from 0 to 30, inclusive), where smaller values are indicative of a more rugged landscape.

Continued on Next Page…

Table 2.1 – Continued

Technique 5:	**Epistasis variance** by Davidor [27] with extensions [106, 109, 130], 1991.
Focus:	Epistasis (Section 2.3.1).
Search Indep.:	Complete: based on fitness function alone.
Assumptions:	Assumes a binary representation.
Description:	A measurement of epistasis is calculated based on a linear composition of a string solution from its bits. The level of inaccuracy (the epistasis variance) of the linear decomposition of the function is used as an estimate of the amount of non-linearity in the function.
Result:	A single value (from 0 to a non-normalized positive number), where 0 indicates no dependency between genes.
Critique:	Criticisms of epistasis variance are that it detects the absence rather than the presence of epistasis, requires the use of all solutions in the problem space, is computationally intensive ($O(n^2)$ to compute the exact value [65]) and that it has limited value as a measure of GA-Hardness [65, 69, 106, 107, 129, 130].
Technique 6:	**Formae variance** by Radcliffe and Surry [123], 1995.
Focus:	Fitness variance of formae (Section 2.3.3).
Search Indep.:	Medium: based on evolutionary notion of formae.
Assumptions:	Assumes a discrete representation.
Description:	Given a discrete fitness function and a sample of randomly generated formae (generalised schemata) at each order, calculate the variance of fitness values for each forma order. The premise is that lower variance will provide more exploitable information for evolutionary search algorithms.
Result:	Plot of fitness variance of formae against forma order, where a plot in which variance falls more quickly is indicative of more exploitable information.

Continued on Next Page. . .

Table 2.1 – Continued

Technique 7:	**Fitness distance correlation** and scatter plots by Jones and Forrest [68] with extensions [3, 67, 162], 1995.
Focus:	Deception with respect to local search (Section 2.3.6).
Search Indep.:	High: uses Hamming distance as basis of measure.
Assumptions:	Requires knowledge of global optima. Assumes the existence of a measure of distance between solutions.
Description:	Given a random sample of points in the search space, each point i generates a pair (f_i, d_i), where f_i is the fitness of point i and d_i is the distance of point i to the nearest global optimum. The fitness distance correlation is calculated as the correlation coefficient of this set of (fitness, distance) pairs.
Result:	A single correlation value r (between -1 and +1, inclusive), where for maximisation problems, low values ($r \leq -0.15$) are easy, values around 0 ($-0.15 < r < 0.15$) are difficult and higher values ($r \geq 0.15$) are misleading. A scatter plot of fitness against distance is used when r is insufficient as a measure of the relationship between fitness and distance.
Application:	Fitness distance correlation has been widely used to study problems for different purposes including analysing neural network error surfaces [39], investigating the effect of different representations on the multidimensional knapsack problem [153], analysing the fitness landscape of heuristics to be used by hyper-heuristics [111], characterising the CEC 2005 benchmark suite [104] and with other landscape features for predicting the performance of covariance matrix adaptation evolution strategy (CMA-ES) algorithms [101]. Fitness distance correlation is investigated further in Section 5.1.1 of this thesis.
Critique:	A significant limitation of fitness distance correlation is that the optimal solution(s) must be known beforehand. It is also computationally intensive to compute ($O(n^2)$ [65]) and many have shown that it is not a reliable predictor of problem difficulty [3, 65, 106, 107, 121, 129].

Continued on Next Page...

Table 2.1 – Continued

Technique 8:	**Static-ϕ metric** by Whitley *et al.* [179] with extensions [57, 125], 1995.
Focus:	GA deception (Section 2.3.6).
Search Indep.:	Medium: based on schemata, applicable to recombinative algorithms.
Assumptions:	Requires knowledge of global optima. Restricted to binary representations.
Description:	Given a binary fitness function with all hyperplane partitions, the static-ϕ metric calculates the degree of consistency between a ranking of schemata within hyperplane partitions based on average fitness values and a ranking based on the distance from the global optimum (using a form of match counting).
Result:	A single value, from 0 to a positive value (could be normalized).
Technique 9:	**Density of states** by Rosé *et al.* [136], 1996.
Focus:	Fitness distribution (Section 2.3.3).
Search Indep.:	Complete: based on fitness function alone.
Assumptions:	None.
Description:	Given a sample of points (the Boltzmann ensemble method of sampling was used in the original study), the density of states quantifies the number of solutions with a given fitness value. The shape of the density of states graph (when plotted over a range of fitness values) can serve as a classifier of fitness functions. For example, maximisation problems with a fast decay of the density of states graph is indicative of the fast decay of the probability of finding a better solution, so should be harder to solve [136].
Result:	A plot of the number of solutions per fitness value.
Application:	Examples of problems that have been studied using density of states include maximal constraint satisfaction problems [8] and road network optimisation problems [139].
Critique:	Reeves [129] criticizes the density of states measure in that it gives no information about how the fitness values are topologically related to each other.

Continued on Next Page…

Table 2.1 – Continued

Technique 10:	**First entropic measure (FEM) and second entropic measure (SEM)** by Vassilev *et al.* [169, 170, 171, 172], 1997.
Focus:	Ruggedness and smoothness with respect to neutrality (Section 2.3.8 and Section 2.3.9).
Search Indep.:	High: based on a random walk through the fitness landscape.
Assumptions:	Assumes a discrete representation.
Description:	Based on a random walk, a sequence of three-point objects are generated. These objects are classified as rugged, smooth or neutral, based on the change in fitness values between neighbouring points. The ruggedness/smoothness of the landscape is estimated using a measure of entropy with respect to the probability distribution of the rugged/non-rugged elements within the sequence.
Result:	A graph illustrating how ruggedness/smoothness changes with an increase in landscape neutrality. Ruggedness/Smoothness values are in the range $[0, 1]$ where 1 indicates maximal ruggedness/smoothness.
Application:	Example applications of FEM and SEM include analysis of the problem of learning robot soccer goal-scoring behaviour [133] and characterisation of continuous optimisation problems sampled during the search process [105]. FEM is investigated further in Section 4.4.1 of this thesis.
Technique 11:	**Amplitude Spectra** by Hordijk and Stadler [62], 1998.
Focus:	Ruggedness (Section 2.3.8).
Search Indep.:	High: based on any notion of neighbourhood.
Assumptions:	Assumes a discrete representation.
Description:	Using a form of Fourier analysis, the fitness landscape is decomposed into elementary landscapes. The resulting amplitude spectrum provides a summary of the properties of a landscape.
Result:	A graph of amplitude values for different interaction orders.

Continued on Next Page…

Table 2.1 – Continued

Technique 12:	**Bit-wise epistasis** by Fonlupt *et al.* [36], 1998.
Focus:	Epistasis (Section 2.3.1).
Search Indep.:	Complete: based on fitness function alone.
Assumptions:	Assumes a binary representation.
Description:	For each bit position i calculate the variance of the fitness differences at that position by comparing the fitness values of the genotypes with 0 in bit position i and 1 in bit position i, with the other bit position values staying the same. The computation is based on a full enumeration of the search space if feasible. If not, bit-wise epistasis is approximated on a sample of schemata.
Result:	A plot of bit-wise epistasis values (in range $[0, 1]$) for each bit position, where a value of 0 for all bit positions indicates no dependency between variables.
Critique:	Jansen [65] shows that bit-wise epistasis is in general extremely computationally expensive and that estimates are unpredictable.
Technique 13:	**HDIL and HDBL** by Belaidouni and Hao [9], 2000.
Focus:	Fitness distribution layout (Section 2.3.4).
Search Indep.:	High: uses Hamming distance as basis of measure.
Assumptions:	Assumes a binary representation.
Description:	Iso-cost levels are defined (sets of solutions with the same fitness values). The HDIL (Hamming Distance In a Level) measures the similarity of solutions within a given iso-cost level, based on the average Hamming distance between solutions in the set corresponding to that iso-level. The HDBL (Hamming Distance Between a Level) quantifies the distance between two iso-cost level sets C and C', based on the average Hamming distance required for solutions from C to reach any solution in set C'.

Continued on Next Page…

Table 2.1 – Continued

Result:	A single HDIL value for each iso-cost level, where a low value indicates that solutions with the same fitness value are clustered together in the search space and a high value that the solutions are spread out. A single HDBL value for each pair of iso-cost levels, where a low value indicates that on average a small Hamming distance has to be covered to move from a solution in one iso-cost level to a solution in the other iso-cost level.
Technique 14:	**Neutral walk** by Reidys and Stadler [131], 2001.
Focus:	Neutrality (Section 2.3.9).
Search Indep.:	High: based on generic notion of neighbourhood.
Assumptions:	Assumes a discrete representation.
Description:	From a random starting position x_0 in the search space, perform a neutral walk as follows: generate all neutral neighbours of x_0. Find one neutral neighbour for which the total distance from the starting point will increase with the step. This process is continued until there are no neutral neighbours that result in the total distance increasing.
Result:	A single value (the number of steps in the neutral walk).
Technique 15:	**Fitness evolvability portraits** by Smith *et al.* [142], 2002.
Focus:	Evolvability (Section 2.3.11).
Search Indep.:	Medium: quantifies evolvability of a solution with reference to a particular operator (mutation in the original study).
Assumptions:	Assumes a discrete representation.
Description:	For all solutions in a sample, calculate the evolvability metrics (such as the expected fitness of the top C_{th} percentile of offspring fitnesses). Determine the average metrics for solutions with the same (or similar) fitness values.
Result:	Plots of average evolvability metrics against fitness.

Continued on Next Page...

Table 2.1 – Continued

Technique 16:	**Fitness cloud** by Verel *et al.* [173] with extensions [164], 2003.
Focus:	Evolvability (Section 2.3.11).
Search Indep.:	Medium: illustrates evolvability with reference to a particular search operator.
Assumptions:	Assumes the existence of a neighbourhood function.
Description:	For every solution x in the search space S of all possible solutions, determine a neighbour $x' \in S$ based on some search operator and plot the points $(f(x), f(x'))$ for every $x \in S$, where f is the fitness function.
Result:	Scatterplot showing the relationship between fitness values of parents and offspring.
Application:	Fitness clouds are investigated further in Section 5.1.3 of this thesis.
Critique:	Lu, Li an Yao [83] show that the neighbourhood sample size has a drastic influence on the fitness cloud generated and so argue that fitness clouds are an unreliable characterisation of evolvability.
Technique 17:	**Negative slope coefficient** by Vanneschi *et al.* [162, 164] with extensions [118, 166], 2004.
Focus:	Evolvability (Section 2.3.11).
Search Indep.:	Medium: based on evolvability with reference to a particular search operator.
Assumptions:	Assumes the existence of a neighbourhood function.
Description:	Given a fitness cloud (Technique 16) partitioned into discrete bins, line segments are defined between the centroids of adjacent bins. The negative slope coefficient is the sum of all negative slopes between segments.
Result:	A single value (in the range (-∞,0], where 0 indicates an easy problem and smaller values indicate more difficult problems).
Application:	Negative slope coefficient was shown to be a reliable indicator of problem difficulty in a real world pharmaceutical application [163]. Negative slope coefficient is investigated further in Section 5.1.4 of this thesis.
Critique:	Problems with the negative slope coefficient are that the result is not normalized [162], the measure is highly influenced by the choice of parameters (it has been shown that the NSC measure tends to zero as the minimum number of points in a bin increases [168]) and is not always a reliable measure of problem difficulty [117, 156, 157].

Continued on Next Page…

Table 2.1 – Continued

Technique 18:	**Information landscape hardness measure** by Borenstein and Poli [14, 15] with extensions [17], 2005.
Focus:	Deception in terms of difference from a landscape with perfect information for search (Section 2.3.6).
Search Indep.:	High: based on a comparison to an optimal landscape, which assumes the same neighbourhood structure.
Assumptions:	Requires knowledge of global optima. Assumes a discrete representation.
Description:	Given a discrete problem, compute the information landscape (matrix of probabilities of superiority of every solution with respect to every other solution). Determine an optimal information landscape, which presents perfect information to guide search. Calculate the distance between the optimal information landscape and the information landscape of the problem.
Result:	A single value in the range $[0, 1]$, where a value of 0 indicates no misleading information and 1 indicates maximal misleading information (difference from the optimal information landscape).
Application:	Information landscape hardness measure is investigated further in Section 5.1.2 of this thesis.
Technique 19:	**Dispersion metric** by Lunacek and Whitley [84], 2006.
Focus:	Global topology or presence of funnels (Section 2.3.7).
Search Indep.:	High: requires the calculation of distances in the solution space.
Assumptions:	Assumes the existence of a measure of distance between solutions.
Description:	Given a sample of points below a fitness threshold: if a decrease in threshold (assuming a minimisation problem) results in an increase in the dispersion of the points from the sample that are below the threshold, then this indicates the presence of multiple funnels in the landscape. Dispersion is calculated as the average pairwise distance in solution space between all points in a sample. The dispersion metric is calculated as the dispersion of a sample of points subtracted from the dispersion of a subset of the fittest points from the same sample.
Result:	A single value where smaller values (negative values) indicate a simpler global topology and larger values (positive values) indicate the presence of funnels. The magnitude of the dispersion metric is dependent on the scale of the distances in the search space.

Continued on Next Page. . .

Table 2.1 – Continued

Application:	Muñoz *et al.* [101] used dispersion metric as one of the landscape features for predicting the peformance of CMA-ES algorithms. Dispersion metric is investigated further in Section 4.5.1 of this thesis.
Critique:	Müller *et al.* [103] showed that dispersion metric was not always a reliable predictor of the presence of multi-funnels in a landscape.
Technique 20:	**Measures on neutral networks** by Vanneschi *et al.* [165] with extensions [167], 2006.
Focus:	Neutrality (Section 2.3.9).
Search Indep.:	Medium: based on a notion of neighbourhood as defined by a search operator.
Assumptions:	Assumes a discrete representation.
Description:	Given a discrete fitness landscape, determine the set of all neutral networks (plateaus formulated as connected graphs of solutions with equal fitness neighbours). Measures are defined based on this set: average neutrality ratio, average fitness gain, non-improvable and "non-worsenable" solutions ratios.
Result:	Scatterplots of measures with respect to fitness values of neutral networks.
Technique 21:	**Fitness-probability cloud** by Lu, Li and Yao [83], 2011.
Focus:	Evolvability (Section 2.3.11).
Search Indep.:	Medium: based on evolvability with reference to a particular search operator.
Assumptions:	Restricted to problems with a discrete representation since the technique is based on the notion of an escape rate [94], which assumes discrete steps through the search space.
Description:	Using a sample of n solution points (Metropolis-Hastings sampling used in the original study) with associated fitness values $f_1, \ldots, f_n$, generate a sample set of neighbours for each point through one application of a given search operator. Calculate the proportion P_i of neighbours with improved fitness for each f_i. The fitness-probability cloud is the set of (f_i, P_i) points.
Result:	A plot of (f_i, P_i) pairs, where f_i is a fitness value and P_i is the estimated escape probability of the sampled point i.

Continued on Next Page...

Table 2.1 – Continued

Technique 22:	**Accumulated escape probability** by Lu, Li and Yao [83], 2011.
Focus:	Evolvability (Section 2.3.11).
Search Indep.:	Medium: based on evolvability with reference to a particular search operator.
Assumptions:	As for Technique 21.
Description:	Given a fitness-probability cloud as defined in Technique 21: $fpc = (f_1, P_1), \ldots, (f_n, P_n)$, the accumulated escape probability is defined as the mean of all P_i values in fpc.
Result:	A single value in the range $[0, 1]$, where a higher value indicates higher evolvability.

2.5 Discussion of survey of fitness landscape analysis techniques

The aim of this study was to make sense of the body of work outlined in Table 2.1 in order to better utilise these techniques in practical ways. This section highlights what the survey reveals: where the focus has been, where the gaps are and possible ways in which techniques can be adapted to be more usable or relevant. The main points of the discussion in this section are summarised as possible ways forward in Table 2.2.

2.5.1 The focus of techniques

Scanning the Focus attribute in Table 2.1 reveals how the techniques for characterising problems have evolved over time. Studies starting in the late 1980s through to the 90s had a strong focus on ruggedness, with some focus on other themes including deception, epistasis and fitness variance/distribution. The late 90s saw the emergence of neutrality as one of the new focus areas, with a number of studies highlighting the fact that there were problems that were not rugged and yet were hard to solve and many of these had high neutrality. The 2000s see evolvability emerge as a new focus of many techniques, with other themes including global topology and fitness statistics.

The many different factors on which the techniques focus highlight the wide range of features that can influence problem difficulty. Each factor is clearly important to some degree and it opens the question: Are there characteristics which are also important, but are missing from the list of available techniques? For example, symmetry is known to influence problem difficulty [26, 108, 137, 178], but to the authors' knowledge there are no known techniques for measuring symmetry in fitness landscapes. Another example is the degree of variable interdependency. Although there are techniques for measuring epistasis that appear in the survey, these all only apply to discrete representations. These and other potential factors point to possible areas for future work.

There are four techniques in Table 2.1 that focus on measuring deception, Goldberg's GA-deception (Technique 1), Jones and Forrest's fitness distance correlation (Technique 7), Whitley *et al.*'s static-ϕ metric (Technique 8), and Borenstein and Poli's information landscape hardness measure (Technique 18). These four techniques are also the only ones listed that require knowledge of the global optima. This is because it only makes sense to talk of deception in reference to finding the optimal solution(s). Characterising a problem based on deception is not useful in practice for two reasons:

- If the aim is to obtain *a priori* information on the problem, the global optima will not be known.
- In many cases it may be infeasible to expect an algorithm to find a global optimum and if a problem guides an algorithm to a reasonable solution, then this may be sufficient.

Assuming a broader notion of success and failure than finding an optimal solution, an alternative aim could be to measure the ease or difficulty with which a search process will progress towards a place of better fitness. This is equivalent to a shift in focus from optimality to searchability (or evolvability as it is more commonly known). If this position is taken, it becomes possible to use techniques such as fitness distance correlation (Technique 7) even when the global optimum is not known. The fitness distance correlation measure was based on the premise that *if the fitness function correlates well with the distance to the optimum, then search will be easier* (assuming a minimisation problem). If the measure is changed to focus on searchability, rather than deception or problem difficulty, and the premise is re-stated as: *if the fitness function correlates well with the distance to a position of higher fitness, then search will progress more easily*, then the technique can be used with the most fit value from a sample in place of the optimum. For example, given an unknown problem, a random sample of solutions can be generated and the fitness values determined. From this sample, the most fit solution is determined and is used as the basis for the fitness distance correlation calculation of Technique 7. The result would no longer be a measure of deception for local search, but would instead be a measure of how easy or hard it would be for a local search algorithm to progress to a place of better fitness. In this way, a technique for measuring deception or problem difficulty is converted into a technique for measuring problem searchability. This idea is pursued further in Section 5.2 where the fitness distance correlation and information landscape hardness measures are adapted to measures of searchability.

Table 2.2: Summary of some possible ways forward

Way forward 1:	New techniques for features not covered by existing techniques.
Description:	There are features that are known to influence problem difficulty, but for which there are no known predictive measures that can be used to obtain *a priori* information on the problem.
Examples:	Symmetry (Section 2.3.10) and variable interdependency for continuous functions (Section 2.3.1).
Way forward 2:	Shifting focus from optimality to searchability (or evolvability).
Description:	Techniques which measure deception assume knowledge of the global optima, which is not known for unseen problems. These techniques can be converted to instead measure searchability (or evolvability), by basing the analysis or calculation on the fittest solution from a sample, instead of the global optimum.
Examples:	This approach could apply to GA-deception (Technique 1), fitness distance correlation (Technique 7), static-ϕ metric (Technique 8), or information landscape hardness measure (Technique 18). Technique 7 and 18 are adapted in this way in Chapter 5.
Way forward 3:	Generalising the notion of neighbourhood.
Description:	Techniques with medium search independence can be adapted to work with more general notions of neighbourhood and in this way be adapted to techniques with high search independence.
Examples:	Fitness cloud (Technique 16) and associated negative slope coefficient (Technique 17) could be adapted to work with a generic distance measure as a neighbourhood function.
Way forward 4:	Specialising the notion of neighbourhood.
Description:	Techniques with high search independence can be adapted to have medium independence by working with more specific notions of neighbourhood for a given search algorithm.

Continued on Next Page...

Table 2.2 – Continued

Example:	Correlation length (Technique 3), based on random walks, can be adapted to measure ruggedness of a search path of a particular search algorithm.
Way forward 5:	Adapting techniques for different representations.
Description:	Techniques that are defined for one representation (e.g. discrete) can possibly be adapted to be used for problems with a different representation (e.g. continuous representation).
Example:	Information landscape hardness measure (Technique 18) defined for discrete problems could be adapted for continuous problems based on a random sample of solutions and this is investigated in Section 5.2.2.
Way forward 6:	Scalarizing visual outputs.
Description:	Techniques that produce plots or graphs can form the basis for new numerical measures to facilitate automated analysis.
Examples:	Density of states (Technique 9), which results in a visual plot, could be condensed into a single measure that in some way captures the shape of the graph.

2.5.2 Search independence

Each technique in Table 2.1 is characterised as having complete, high, or medium search independence. Depending on the purpose and context, different kinds of techniques will be more suitable. On the one hand, where the choice of algorithm is set, an appropriate technique with medium search independence could be used to better understand the given problem with reference to that algorithm. For example, assuming an evolutionary algorithm is being used, the accumulated escape probability (Technique 22) could be used to guide the choice of appropriate parameters for the algorithm on the given problem. On the other hand, where the purpose is to choose an appropriate algorithm for a given problem, techniques for characterising the problem with complete or high search independence will be more useful.

In some cases a technique that is based on a particular search operator and therefore regarded as having medium search independence could be adapted to use a generic notion of neighbourhood, so that the analysis could apply to different algorithms. For example, the negative slope coefficient (Technique 17) is described as having medium search independence because the neighbourhood is defined in terms of a particular search operator (subtree mutation for genetic programming in the original study). If instead, the neighbourhood was defined using some generic notion of distance, such as Euclidean distance for a continuous problem, then the technique will be used with high search independence. Conversely, a technique with high search independence can be adapted into a technique with medium search independence. For example, the correlation length technique (Technique 3) originally based on random walks, could be instead based on the trajectory of a particle within a PSO swarm. In this way, the technique would be used to measure ruggedness from the particular viewpoint of a PSO search process.

2.5.3 Further proposed work

Many of the techniques outlined in the survey assume the fitness function (f) is a mapping from the binary space to real space ($f\colon \{0,1\}^n \rightarrow \mathbb{R}$), or from some discrete alphabet to real space. In some cases this is a restriction, because there is no obvious way of using the technique for other representations, such as continuous representations (where $f\colon \mathbb{R}^n \rightarrow \mathbb{R}$). In other cases, although the technique is described in terms of one representation, this is not necessarily a restriction. For example, the information landscape hardness measure (Technique 18) is defined for discrete representations and involves constructing a matrix of fitness superiority values of all solutions with respect to all other solutions. This approach could be adapted to a continuous representation by using a random sample of solutions. Without knowledge of the global optimum the technique would also have to be adapted in the way described in Way forward 2 of Table 2.2. This is investigated further in Section 5.2.2 of this thesis.

The Result attribute of Table 2.1 also presents opportunities for further work. Some of the techniques produce plots or graphs as results. While visual output is useful for human analysis, numerical output is more useful for facilitating automated analysis. An example of a numerical measure that is based on an existing technique is Vanneschi *et al.*'s negative slope coefficient (Technique 17), which is a numerical output measure based on Verel *et al.*'s fitness cloud scatterplot (Technique 16). In Section 4.4 a single measure of ruggedness is proposed based on the first entropic measure output graph by Vassilev *et al.* (Technique 10).

In similar ways, other measures with non-numerical output could form the basis of new numerical measures. For example, the result of the density of states technique (Technique 9) is a plot of the number of solutions against fitness. The tail end of the graph closer to optimal fitness values is the more significant part in terms of assessing the difficulty for search. A possible single measure of fitness distribution could in some way quantify the proportion of solutions at better fitness values in contrast to the number of solutions at other less fit fitness values.

2.6 The link between problems and algorithms

Other than developing new techniques or adapting existing techniques for fitness landscape analysis, there is also further research required in understanding the link between problem characteristics and algorithm performance. Although there has been extensive research into the development of new algorithms for solving optimisation problems, there has been relatively little focus on understanding how these new algorithms behave with respect to particular problems. When a publication introduces a new algorithm or variation on an existing algorithm, the approach is typically to demonstrate empirically that the algorithm out-performs other algorithms on a number of selected benchmark problems. The proposers of a new algorithm will usually neglect to provide any analysis of problems on which the proposed algorithm will perform poorly and why.

There have been some recent studies attempting to address this gap in understanding of algorithm behaviour on problems. These include: analysing the link between known problem features (such as modality and separability) and algorithm performance [18]; analysing which problems are hard for particular algorithms [23, 56, 113]; analysing the correlation between problem difficulty measures and hybrid evolutionary algorithms [115]; analysing problem attraction basins to understand the behaviour of differential evolution and covariance matrix adaptation algorithms [19]; and analysing which problems require smaller population sizes in evolutionary algorithms [24]. An interesting recent approach to understanding algorithms and their behaviour on problems, proposed by Morgan and Gallagher [100], is to generate two dimensional problem instances that maximise the performance difference between two algorithms. The problems can then be visualised to try to better understand algorithm behaviour.

In addition, a number of studies have used problem characteristics to guide algorithm choices. Examples include: the use of correlation length and fitness distance correlation to design memetic algorithms [95]; the use of a ruggedness coefficient with dominance to guide the choice of algorithm for the quadratic assignment problem [4]; and using the negative slope coefficient to choose the most appropriate genetic programming configuration to solve real life applications [163].

This thesis extends this work by proposing a wider range of measures and investigating the possibility of predicting algorithm performance (PSO performance in particular) based on multiple characteristics. The next two chapters propose a number of measures for approximating problem characteristics and Chapter 6 investigates predicting algorithm performance based on considering multiple characteristics together.

2.7 Summary

This chapter provided a survey of existing techniques for characterising problems. Each technique was described in terms of the focus (what is measured), the level of search independence, assumptions on which the technique is based, and the result produced. The survey reveals how the focus has changed over the last two decades. Some characteristics, such as ruggedness, are the focus of many different techniques, but others, such as symmetry, are not well represented. Suggestions are made for ways in which existing techniques can be adapted to be more usable or relevant. Techniques that require knowledge of the global optima can be used without this knowledge by shifting the focus from optimality to searchability. The same fitness analysis technique can also be used in multiple ways by changing the search independence and in effect analysing different fitness landscapes for the same problem. In addition, fitness landscape analysis techniques that are based on the assumption that the search space is discrete can be adapted to also apply to continuous representations. A further suggested way forward is to find ways of scalarizing visual outputs.

The following two chapters investigate some of these suggested ways forward. In Chapter 4 an existing fitness landscape analysis technique (Technique 10), defined for discrete spaces and producing graphical output, is adapted to work in continuous spaces and to produce a single numerical approximation of ruggedness. Chapter 4 also investigates measures for gradients, which is a feature not covered by existing techniques. Chapter 5 adapts two existing measures (Techniques 7 and 18) by shifting focus from optimality to searchability and investigates ways of scalarizing fitness clouds (Technique 16) based on PSO updates.

Chapter 3

Benchmarks, Algorithms and Performance

Section 1.1 described the algorithm selection problem and stated the prerequisites for solving the algorithm selection problem [144]:

1. A large number of problem instances with different levels of difficulty;
2. A large number of different algorithms for solving these problem instances;
3. Metrics for evaluating the performance of algorithms; and
4. The existence of features that can be used to suitably characterise the properties of problems.

This chapter focuses on the first three prerequisites. Section 3.1 describes a number of continuous benchmark problems with different characteristics and hence different levels of difficulty. Section 3.2 describes seven variations of the PSO algorithm, which have different search behaviours. Section 3.3 addresses the third requirement of evaluating performance of algorithms on problems and proposes a number of metrics that can be used to contrast algorithm performance across different known problems. Chapters 4 and 5 then focus on the last requirement by proposing feature metrics for characterising continuous problems.

3.1 Benchmarks

Before a new optimisation algorithm or variation on an existing algorithm can be shown to be successful, the performance of the algorithm is usually tested on a range of problems and contrasted with the performance of some other algorithm(s) on the same set of problems. In these empirical comparative studies, knowledge of the true optima of the problem is not normally needed, because the best solutions found are sufficient for showing whether one algorithm performed significantly better than another. For the purposes of this study, however, knowledge of the optimum is needed, so that performance of an algorithm can be quantified for different problems to be used as the basis for training a performance predictor.

A number of papers have summarised existing benchmark problems or have proposed new problems that could serve as suitable benchmarks. Some useful sources of benchmark functions include Whitley *et al.* [178], Yao *et al.* [187], Suganthan *et al.* [150], Price *et al.* [119], Mishra [96, 97] and Rahnamayan *et al.* [124]. A selection of functions from these sources are listed in Appendix A in Table A.1. Only functions with known global optima are included. In selecting functions, the aim was to cover a wide range of different characteristics. Most functions are defined for different dimensions, but some are only specified for two dimensions. In each case, the optimum value is specified for the given domain.

Figures A.1 to A.3 in Appendix A provide visual plots of the functions. Plots are either of one-dimensional or two-dimensional versions of the functions. In some cases more than one plot is provided for a single function to provide additional insight into the function. In many cases, the 1D or 2D plots reveal characteristics of the problems, but these characteristics sometimes differ in higher dimensions.

From the visual plots it can be seen that some functions are unimodal, whereas others are multimodal. Examples of unimodal functions include Beale, Quadric, Quartic, Schwefel 2.22 and Spherical. Although functions like Bohachevsky may appear unimodal from the plot of the full domain, a plot of a smaller domain shows that the function is multimodal. The multimodal functions differ in the degree of multimodality. Some functions have only a few local optima, such as Goldstein-Price, Michalewicz (in 2D) and Six-hump camel-back. Others have many more local optima with a rugged landscape for the given domain, such as Alpine, Rastrigin and Schwefel 2.26. Still others have huge numbers of local optima with an extremely rugged landscape, such as Ackley, Griewank and Salomon.

In many cases, the 1D or 2D plots reveal characteristics of the functions which are similar for higher dimensions, but in some cases, the characteristics are known to differ in higher dimensions. For example, Griewank is very rugged on a micro scale in low dimensions, but the function becomes relatively easier to solve as D increases, because the underlying unimodal shape begins to dominate the local optima created by the cosine term [81]. Another example is the Rosenbrock function, which is unimodal up to 3D, but shown to have a local (non-global) minimum for $4 \leq D \leq 30$ [140].

The benchmarks in Appendix A, Table A.1 are revisited a number of times in subsequent sections and chapters of the thesis. Some of the problems are solved using PSO algorithms in Section 3.3 to illustrate the use of proposed performance metrics. Chapters 4 and 5 use selected problems to test proposed fitness landscape metrics. Finally, the full set of function are used in Chapter 6 to generate data sets for training and testing PSO performance predictors.

3.2 PSO algorithms

Particle swarm optimisation (PSO) [31, 75] is a stochastic population-based optimisation technique. Starting with a random swarm of solutions, called particles, the positions of particles in the search space are adjusted at each iteration of the algorithm. The adjustment has random elements, but is largely determined by the distance to the best solution found in the neighbourhood of the particle (called the neighbourhood best) and the distance from best solution found by the particle itself during the search process (called the personal best or *pbest*). In this way, the swarm of particles move around, exploring the search space, in time hopefully converging on a good solution. There are many different varieties of PSO algorithms. This section describes seven basic variations that differ in their exploration and exploitation behaviours [35]. These seven variations are used later in Chapter 6 when the link between fitness landscapes characteristics and PSO performance is investigated.

3.2.1 Traditional global best PSO

Traditional global best PSO model (gbest PSO for short) [31, 75], also known as 'vanilla' PSO, determines the multidimensional position of a particle, $\mathbf{x}_i$, at time step $t+1$ by adding a multidimensional step size (called the *velocity* of the particle), $\mathbf{v}_i$, at time step $t+1$, to the position of the particle at time t, using the equation:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{3.1}$$

The velocity at time step $t+1$ is given as:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 \cdot \mathbf{r}_1(t) \odot (\mathbf{y}_i(t) - \mathbf{x}_i(t)) + c_2 \cdot \mathbf{r}_2(t) \odot (\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)) \tag{3.2}$$

where w is the inertia weight (introduced by Shi and Eberhart [141]), $\mathbf{v}_i(t)$ is the velocity of particle i at time stamp t, c_1 and c_2 are the cognitive and social acceleration constants, respectively, $\mathbf{r}_1(t), \mathbf{r}_2(t) \sim U(0,1)^D$ where D is the dimension of the problem, $\odot$ denotes element-by-element vector multiplication, $\mathbf{y}_i(t)$ refers to particle i's personal best position and $\hat{\mathbf{y}}(t)$ refers to the global best position at time step t, being the best solution from the set of personal best positions of all particles.

Equation 3.2 shows that the position of a particle is influenced by three terms: the particle's previous velocity, the relative position of the *pbest* particle (cognitive component) and the relative position of the swarm's *gbest* (social component). Since the *gbest* is determined from the fittest *pbest* of all particles, the neighbourhood of each particle is the entire swarm. This is known as a star topology, because all particles are 'connected' (share information) with all other particles. The behaviour of the traditional gbest PSO is influenced by the relative weights of these three terms, set using the constants w, c_1 and c_2. The choice of values of these constants has to be made together to ensure convergence of the swarm [160]. While the social terms pull particles towards the same point, the cognitive terms pull particles in potentially different directions. When the cognitive and social acceleration constants are given the same values, the traditional gbest PSO model results in a balance of exploration and exploitation behaviour, but this depends on the value of w. Although the optimal choice of parameters is problem dependent, a common choice that works reasonably well for many problems is 0.7298 for w and 1.496 for both acceleration constants [32].

3.2.2 Cognitive PSO

The position update of the cognitive PSO variation [73] is the same as for the traditional PSO (Equation 3.1), but the social component is removed from Equation 3.2, giving:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 \cdot \mathbf{r}_1(t) \odot (\mathbf{y}_i(t) - \mathbf{x}_i(t)) \tag{3.3}$$

The cognitive PSO variation results in higher exploration than the traditional PSO model, since each particle is pulled in the direction of its own best position, resulting in essentially a swarm of individual local search optimizers (hill climbers).

3.2.3 Social PSO

The position update of the social PSO variation [73] is the same as for the traditional PSO (Equation 3.1), but the cognitive component is removed from Equation 3.2, giving:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_2 \cdot \mathbf{r}_2(t) \odot (\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)) \tag{3.4}$$

The social PSO variation results in faster exploitation than the traditional and cognitive variations, since all particles are pulled in the direction of the same global best particle at each time step.

3.2.4 Local best PSO

The traditional gbest PSO model uses a star neighbourhood structure where all particles share knowledge with all other particles in the form of the position of the global best particle. In an alternative model, called local best PSO (lbest PSO for short) [31] each particle is part of a smaller neighbourhood and each neighbourhood has its own neighbourhood best particle. In its simplest form, the membership of particles to neighbourhoods is determined using particle indices, which forms a ring structure. For example, for a neighbourhood size of two, a particle $\mathbf{x}_i$ will be in a a neighbourhood with $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i-1}$.

The position update of particle $\mathbf{x}_i$, at time step $t+1$ is the same as Equation 3.1, but the velocity update is given as:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 \cdot \mathbf{r}_1(t) \odot (\mathbf{y}_i(t) - \mathbf{x}_i(t)) + c_2 \cdot \mathbf{r}_2(t) \odot (\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) \tag{3.5}$$

where $\hat{\mathbf{y}}_i(t)$ is the best particle in the neighbourhood of particle $\mathbf{x}_i$ at time step t.

Lbest PSO with a ring structure has overlapping neighbourhoods and all neighbourhoods are indirectly connected to all other neighbourhoods. This means that the global best position will affect the positions of all particles, but this will be a delayed effect, promoting more exploration than the gbest PSO model.

3.2.5 Asynchronous global best PSO

In the case of the traditional gbest PSO model, at the end of each iteration (or the start of the next iteration) of the algorithm, the personal best positions of all particles and the global best position of the swarm are determined, after which the new velocities and positions of all particles are updated. In the asynchronous global best PSO model [21], the personal best and global best positions are updated after the position update and new fitness evaluation of each particle. This means that when there is an improvement in the gbest this information is immediately known to the next particle to be updated. Information on the position of the current gbest is therefore shared faster than with synchronous updates.

3.2.6 Bare bones PSO

The bare bones PSO variation [74], instead of a two-step process of calculating the velocity, then adding it as a step size to the previous position of the particle, updates the new position as a single step. The new position of each component j of $\mathbf{x}_i$ at time stamp $(t + 1)$ is sampled from the following Gaussian distribution:

$$x_{ij}(t+1) \sim N\left(\frac{y_{ij}(t) + \hat{y}_j(t)}{2}, \sigma\right) \tag{3.6}$$

with deviation $\sigma = |y_{ij}(t) - \hat{y}_j(t)|$. This means that instead of particles moving in the direction of the global best or personal best particle (or somewhere inbetween), particles ‘jump’ to a random position centred around the point half-way between the personal and global best position (the position $\frac{\mathbf{y}_i + \hat{\mathbf{y}}}{2}$ has been proven to be the point of convergence for particles in the traditional gbest PSO model [155, 159]). As the distance between the personal and global best reduces, so the deviation of the Gaussian distribution reduces, resulting in more exploitation.

3.2.7 Modified bare bones PSO

The modified bare bones PSO [74] is a variation on the barebones PSO model, with extra exploration abilities. As with the bare bones PSO, the model has only a position update, but the new position of each component j of $\mathbf{x}_i$ at time stamp $(t+1)$ is calculated as follows:

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < 0.5 \\ N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma\right) & \text{otherwise} \end{cases} \tag{3.7}$$

Equation 3.7 implies that there is a 50% chance of each component of the new position being set to the equivalent component of the personal best position, otherwise it is set to a random position centred around the point half-way between the equivalent components of the personal and global best positions. This results in more exploration than in the bare bones PSO, particularly early on, since personal best positions are initially widely spread throughout the search space.

3.3 Measuring algorithm performance

Solving the algorithm selection problem in general requires suitable metrics for evaluating the performance of algorithms on a given set of known problems. The performance metrics are used to generate a data set that can be used as the basis for finding a mapping from feature space to algorithm or performance space, as illustrated in Figure 1.1. For any two problems, the metrics should distinguish the relative difficulty of solving the problems by a given algorithm. Equally, given two algorithms, the metrics should distinguish the relative difficulty of the algorithms solving the same problem. This section reviews some existing approaches to measuring performance of optimisation algorithms and then proposes the use of three normalised metrics that can be used for quantifying the performance of algorithms on different problems.

3.3.1 Existing approaches to measuring performance

The most common way of measuring the performance of an optimisation algorithm is in terms of the quality of the solution found in comparison to the quality of the solution found by some other algorithm. Given sufficient independent runs of both algorithms, it is reported whether there is a statistically significant difference in the quality of the solutions found by the algorithms. In the case of benchmark problems with known optimal solutions, the measure of the quality is usually simply the difference in fitness value between the solution found by the algorithm and the optimal solution (sometimes called the fitness error value). To ensure reasonably fair comparisons between different algorithms, the fitness error value is usually based on the best or average solution found after a set number of function evaluations by the algorithm. Using an absolute fitness error value as a measure of performance is suitable when comparing algorithms on the same problem, but cannot be used when comparing the performance of algorithms on different problems with varying fitness ranges. For example, a mean fitness error value of 0.08 on one problem could be regarded as a high performance result, whereas a mean fitness error value of 0.0003 may be regarded as a low performance result for a different problem.

Another common approach to measuring performance is to quantify the percentage of successful runs over a number of runs (frequently called success rate). What constitutes a ‘successful run’ has to be properly defined. Some arbitrary cut-off like “within 10^{-8} of the global optimum” could be sensible in the case of a problem with a fitness range of $[0, 1]$, but would not make sense in the case of a problem with a fitness range of $[0, 10^{30}]$ (such as with the Schwefel 2.22 benchmark function in 30 dimensions). In some cases a fixed accuracy level is specified for each benchmark function [119, 150]. For example, Suganthan *et al.* [150] define fixed accuracy levels for each benchmark function, such as 10^{-6} for F_1 (Shifted Sphere Function) and 10^{-2} for F_6 (Shifted Rosenbrock’s Function). They define a successful run as one during which “the algorithm achieves the fixed accuracy level within the Max_FES for the particular dimension”[150], where Max_FES specifies the maximum number of function evaluations and is defined as $10000 \times D$. In this way, functions with higher dimensions are given more function evaluations to reach the fixed accuracy level.

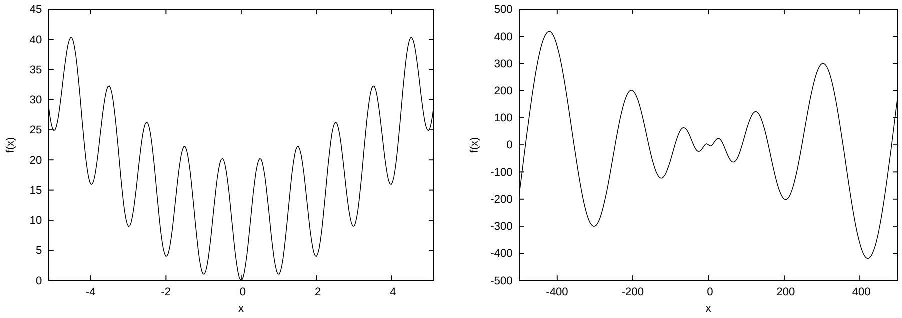
Yet another approach to measuring performance is to quantify how quickly an algorithm is able to find an acceptable solution. For example, the average number of evaluations to a solution (AES) measure, defined over those runs that reach a solution to within a given fixed accuracy level [34].

3.3.2 Proposed performance metrics

This section proposes the use of three normalised algorithm performance measures. Note that the purpose of these measures is to generate data on known problems to be used for the training of a predictor or classifier of algorithm performance on unknown problems. The measures quantify solution quality, rate of success and speed of reaching a solution and can be used to contrast the performance of a single algorithm on multiple problems or multiple algorithms on the same problem. All three measures require knowledge of the range of fitness values of a problem so that the measures can be normalised across different problems.

Estimating fitness range

The range of fitness values for benchmark problems is not always known. In the case of simple functions like the Spherical function, it is obvious that the maximum values lie on the boundaries of the search space. The range of fitness values is then simply the difference between the fitness values on the boundary and the fitness of the known optimal solution. In the case of many other benchmark problems, however, the maximum point is somewhere else in the search space. For example, Figure 3.1 illustrates two common benchmark functions in one dimension. A simple visual inspection of a plot of the functions in one dimension can be used to estimate the range of fitness values of the function. However, in higher dimensions it is not as easy to estimate the fitness range, as the position of the maximum point cannot be assumed to be at the same x position as for the one dimensional version of the function. A simple way of estimating the maximum fitness value of a benchmark problem is to optimise the minimisation problem as a maximisation problem using a suitable optimisation algorithm. The result of the optimisation is the estimated maximum fitness value, called $\hat{f}$. It is important that during the maximisation, the algorithm be confined to the bounds of the search space,



(a) Rastrigin function (b) Schwefel 2.26 function

Figure 3.1: Example one-dimensional minimisation benchmark functions.

because many of the benchmark functions (like those illustrated in Figure 3.1) continue increasing outside the bounds of the problem domain. For a benchmark function, given an estimated maximum fitness $\hat{f}$ and a known minimum fitness value, f^*, the estimated range of fitness values is then defined as $\hat{f} - f^*$.

Definition of a successful run and determining fixed accuracy levels

A *run* of an algorithm on a problem is a single execution of the algorithm with a given maximum number of fitness evaluations (MaxFES) as the stopping condition. For all runs in this study, the MaxFES was set to $10000 \times D$ (dimension), equivalent to $200 \times D$ iterations using a swarm of 50 particles. A *successful run* of an algorithm on a problem is a run that finds a solution with a fitness value that is within a set fixed accuracy level from the objective function value of the global optimum.

Similar to [119] and [150], fixed accuracy levels are specified for different problems. A function with a smaller range of fitness values should have a smaller fixed accuracy level than a function with a larger fitness range. However, rather than using a single fixed accuracy level for each benchmark function (as in [119, 150]), it is proposed that a fixed accuracy level be defined for each function/dimension combination. If the range of fitness values of the same benchmark function increases as the dimension increases, then the fixed accuracy level should also increase. The proposed method for determining the fixed accuracy level of a given benchmark function and dimension is as follows:

1. The estimated fitness range of the problem ($\hat{f} - f^*$) is rounded down to the nearest

10^n (called the *fitness range order*), where n is an integer. Rounding the fitness range down (rather than up) results in a smaller fixed accuracy level and hence higher requirement in terms of accuracy of solution.

2. The *fixed accuracy level* is computed as the fitness range order multiplied by 10^{-8} up to a maximum fixed accuracy level of 10^{-3}. The use of 10^{-8} is specifically chosen to align with error values in other sources [119, 150].

For example, the Ackley benchmark function in one dimension with domain $[-32, 32]$ has a rounded down fitness range of 10^1, resulting in a fixed accuracy of 10^{-7}. Some example benchmark problems with proposed fixed accuracy levels are listed in Table 3.1. The definition of these functions is given in Appendix A, Table A.1. Notice how with some functions, such as Ackley, the fixed accuracy levels stays the same as the dimension increases, but in the case of other functions, such as Griewank, the fitness range increases with an increase in dimension, resulting in a decrease in the fixed accuracy level.

QMetric

Given a run of an optimisation algorithm on benchmark function f with resulting best fitness found f^{min}, the difference in fitness between the best found solution and the optimal solution is quantified as $f^{min} - f^*$. This difference is an absolute measure of fitness error, where 0 is the minimum error and corresponds with the highest possible solution quality. To convert the fitness error into a positive measure of quality, the found solution, f^{min}, is subtracted from the estimated maximum, $\hat{f}$ and scaled by the estimated range of the problem as follows:

$$q = \frac{\hat{f} - f^{min}}{\hat{f} - f^*} \ . \tag{3.8}$$

The normalized measure q is a value in the range $[0, 1]$ where 1 indicates the highest quality, where the fitness of the found solution exactly matches the fitness of the known optimal solution, and 0 indicates the worst possible solution quality of finding the maximum fitness. In order to better distinguish between q values closer to 1, the value of q is scaled exponentially to produce the proposed QMetric measure as follows:

$$\text{QMetric} = 2^{q^{10^4}} - 1 \ . \tag{3.9}$$

Table 3.1: Some example benchmark problems (defined in Appendix A, Table A.1) in different dimensions with estimated maximum fitness ($\hat{f}$), known minimum (f^*), fitness range order and fixed accuracy level.

Function	Dimension	$\hat{f}$	f^*	Fitness Range Order	Fixed Accuracy Level
Ackley	1	22.31	0	10^1	10^{-7}
Ackley	15	22.31	0	10^1	10^{-7}
Ackley	30	22.31	0	10^1	10^{-7}
Griewank	1	92.00	0	10^1	10^{-7}
Griewank	15	1351.00	0	10^3	10^{-5}
Griewank	30	2701.00	0	10^3	10^{-5}
Rosenbrock	2	3905.93	0	10^3	10^{-5}
Rosenbrock	15	54682.97	0	10^4	10^{-4}
Rosenbrock	30	113271.86	0	10^5	10^{-3}
Schwefel 2.26	1	418.98	-418.98	10^2	10^{-6}
Schwefel 2.26	15	6284.74	-6284.74	10^4	10^{-4}
Schwefel 2.26	30	12569.49	-12569.49	10^4	10^{-4}

Figure 3.2 illustrates the relationship between q and QMetric.

For example, given a problem with a fitness range of $[0, 1]$ (with associated fixed accuracy level of 10^{-8}) and global optimum of 0, a best found solution of 10^{-8} would be regarded as a successful run. The resulting QMetric value would be 1.000 (rounded to 3 decimal places), indicating the highest rounded solution quality. On the other hand, a solution of 10^{-5} would result in a q value of 0.99999 and an associated QMetric value of 0.872, indicating a lower solution quality. Any solution with fitness 0.001 and larger will result in a QMetric value of 0 (rounded to 3 decimal places).

SRate

To compare the rate of success of different algorithms on different problems, each problem/algorithm combination is run using MaxFES as the terminating condition. The success rate (SRate) is defined as the number of successful runs that reach a solution

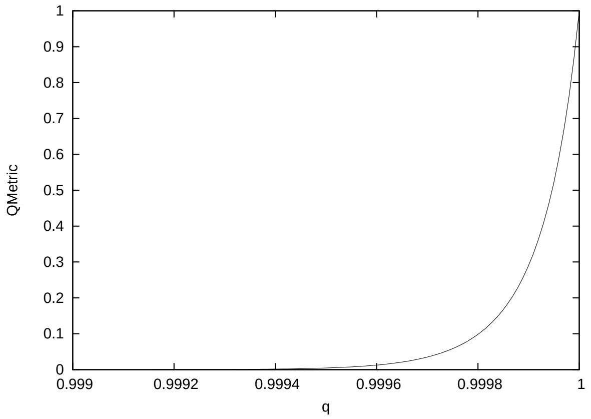


Figure 3.2: Function used to scale fitness quality measure q to QMetric.

within the fixed accuracy level of the global optimum divided by the total number of runs [150]. Like the QMetric, SRate is a value in the range $[0, 1]$ where 1 indicates the highest possible rate of success.

SSpeed

The number of function evaluations taken to reach the global optimum (within the fixed accuracy level) for a given run r is known as FES_r. A proposed metric called the success speed of a run r (SSpeed_r) is defined as:

$$\text{SSpeed}_r = \begin{cases} 0 & \text{if the run is not successful} \\ \frac{\text{MaxFES}-(\text{FES}_r-1)}{\text{MaxFES}} & \text{otherwise.} \end{cases} \tag{3.10}$$

The metric SSpeed_r is a value in the range $[0, 1]$. The highest value for SSpeed_r can only be obtained if the global minimum is reached in the first function evaluation (if FES_r is 1) and this would indicate the highest possible performance in terms of speed. The success speed (SSpeed) over ns successful runs, is defined as:

$$\text{SSpeed} = \begin{cases} \frac{\sum_{r=1}^{ns} \text{SSpeed}_r}{ns} & \text{if } ns > 0 \\ 0 & \text{if } ns = 0\,. \end{cases} \tag{3.11}$$

3.3.3 Performance metrics applied to PSO

To illustrate the use of the proposed three performance metrics, Table 3.2 shows the results of minimizing a number of benchmark functions using a traditional gbest PSO algorithm (Section 3.2.1). The final column in the table refers to the performance class and is described in Section 3.3.4.

The following parameter values were used: 50 particles, 1.496 for both the cognitive and social acceleration constants and 0.7298 for the inertia weight [32]. Benchmark functions and dimensions were selected to illustrate features of the metrics. For each benchmark function/dimension combination, 30 independent runs of the algorithm were performed. The QMetric values are the means of the 30 runs and the value for MaxFES was set at $10000 \times D$, or $200 \times D$ iterations for 50 particles.

In Table 3.2, the easiest function to minimize is the simple unimodal Spherical function and this is reflected in the high values for all performance metrics. The PSO algorithm achieved a QMetric mean value of 1 in dimensions 1, 15 and 30, indicating that the average of the best fitness values found was of the highest possible quality. Similarly, the SRate of 1 indicates that the algorithm found the optimal solution (to within the fixed accuracy level) in all 30 runs. The high SSpeed values indicate that the algorithm found the solution quickly (needing relatively few function evaluations). Recall that the SSpeed metric is a measure of how quickly the solution is found in relation to the maximum number of function evaluations (MaxFES), which increases with dimension (MaxFES is set at $10000 \times D$). For example, an SSpeed value of 0.941 for Spherical in 15 dimensions does not imply that the solution was found in fewer iterations than for 1 dimension (with an SSpeed of 0.845), but rather that the solution was found in a smaller percentage of the maximum number of iterations allowed for that dimension.

From the results, it can be seen that the Spherical problem did not become harder for the PSO algorithm as the dimensions increased. In contrast, the PSO algorithm found it harder to minimize the multimodal Ackley function as the dimensions increased. In 30 dimensions, the algorithm was only able to find the optimal solution in 9 out of the 30 runs (SRate of 0.3).

In many cases the QMetric and SRate have the same value. This indicates that the quality of the solutions and the success rate are essentially reflecting the same informa-

Table 3.2: Results of minimizing a number of benchmark problems using a traditional gbest PSO algorithm, based on 30 runs.

Function	Dimension	QMetric	SRate	SSpeed	Class
Ackley	1	1.000	1.000	0.471	S+
Ackley	15	0.933	0.933	0.855	S
Ackley	30	0.300	0.300	0.862	S
Griewank	1	1.000	1.000	0.721	S++
Griewank	10	0.352	0.033	0.019	S
Griewank	15	0.691	0.100	0.910	S
Griewank	30	0.902	0.367	0.921	S
Rosenbrock	2	1.000	1.000	0.843	S++
Rosenbrock	5	0.931	0.067	0.321	S
Rosenbrock	15	0.888	0.000	0.000	S–
Rosenbrock	30	0.482	0.000	0.000	S–
Salomon	1	1.000	1.000	0.574	S++
Salomon	5	0.000	0.000	0.000	F
Schwefel 2.26	1	1.000	1.000	0.816	S++
Schwefel 2.26	5	0.400	0.400	0.826	S
Schwefel 2.26	15	0.000	0.000	0.000	F
Spherical	1	1.000	1.000	0.845	S++
Spherical	15	1.000	1.000	0.941	S++
Spherical	30	1.000	1.000	0.940	S++

tion. If the quality of the solution is low, it will result in a QMetric value of 0 and this will also be reflected in an unsuccessful run. There are cases, however, where the QMetric differs from the SRate measure. Consider for example the Rosenbrock function in 15 dimensions. None of the 30 runs of the PSO algorithm found the optimal solution, but a QMetric value of 0.888 indicates that the best solutions found were still of a relatively good quality (relatively close to the global optimum fitness value).

It is interesting to notice that, in the case of the Griewank benchmark function, the performance metrics are high in 1 dimension, decrease in 10 dimensions, but then increase in 15 dimensions and increase further in 30 dimensions. This indicates that the problem becomes easier in higher dimensions, which is a known characteristic of the Griewank benchmark function [81].

3.3.4 Performance classes

The three normalized metrics proposed in this section make it possible to compare data from different benchmarks and algorithms. When viewed together the three values provide information on the accuracy, predictability and speed of the algorithm in solving the given problem. In some cases, it is convenient to have a simpler indication of performance. For example, when a number of algorithms and problems are considered together and data mining is used to investigate the link between problem features and algorithm performance (as is done in Chapter 6), it is easier to extract meaning when there is a single performance class for a given problem/algorithm combination. When it is desirable to have a single performance class, the QMetric, SRate and SSpeed values are used to allocate performance into one of the following classes:

- *Always solved and fast* (class symbol S++): problems with an SRate of 1, indicating that the solution was found for all 30 runs of the algorithm, and an SSpeed > 0.5, indicating that the algorithm was able to find the solution in less than half of the allowable time (maximum number of function evaluations) on average.

- *Always solved* (class symbol S+): problems with an SRate of 1 and an SSpeed ≤ 0.5, indicating that the solution was found for all 30 runs of the PSO algorithm, but that more than half of the allowable function evaluations were used to find the solution on average.

- *Sometimes solved* (class symbol S): problems with an SRate less than 1, but greater than 0, indicating that the solution was found for some of the runs.

- *Almost solved* (class symbol S–): problems with an SRate of 0, but a QMetric value greater than 0, indicating that although none of the runs found the solution to within the required fixed accuracy level, a solution was sometimes found that was very close to the optimum.

- *Not solved* (class symbol F): problems with all performance metric values equal to 0.

The final column of Table 3.2 gives the relevant performance class of the examples listed. The class information provides a very quick way of assessing which problems were easy for the algorithm, which were harder and on which problems the algorithm failed.

3.4 Summary

This chapter filled in a number of components of the algorithm selection model as depicted in Figure 1.1 applied to optimisation problems solved using PSO algorithms. The benchmark problems described in Section 3.1 provided a set of problems to define the problem space P. The set of seven PSO algorithms described in Section 3.2 define the algorithm space A and the performance metrics proposed in Section 3.3 define the performance space Y. Given these three components, an algorithm a can be applied to a problem p to produce performance measure y. The remaining aspects of the model include feature extraction (address in Chapters 4 and 5) and algorithm selection / performance prediction (addressed in Chapters 6).

Chapter 4

Ruggedness, Funnels and Gradients

4.1 Introduction

The modality of a fitness function refers to the number of optima and is frequently mentioned in relation to the difficulty of a problem. However, rather than the absolute number of optima, it is the distribution of basin sizes in the fitness landscape and the depth (or height) of the basins that may be more important factors to consider in determining problem difficulty [69]. Consider, for example, the two simple fitness functions illustrated in Figure 4.1. Both functions have three optima (two local and one global), but in the case of the top landscape, the basin containing the global optimum is larger than the two basins containing the local optima, which is not the case with the lower landscape. In the case of a PSO algorithm, if two particles are positioned as illustrated, the social velocity in the top function will pull the righmost particle towards the global optimum, whereas the social velocity in the bottom function will pull the leftmost particle away from the global optimum. The bottom landscape would therefore be regarded as more deceptive for PSO search, because the information from the landscape is pulling the search in the wrong direction. This is supported by Xin *et al.* [184] who studied the notion of deception for PSOs and concluded that the relative size of basins of attraction (local versus global) was the most important factor related to deception for PSOs.

Given an unknown multidimensional optimisation problem, there is no computationally cheap way of determining the relative sizes of basins of attraction. Instead,

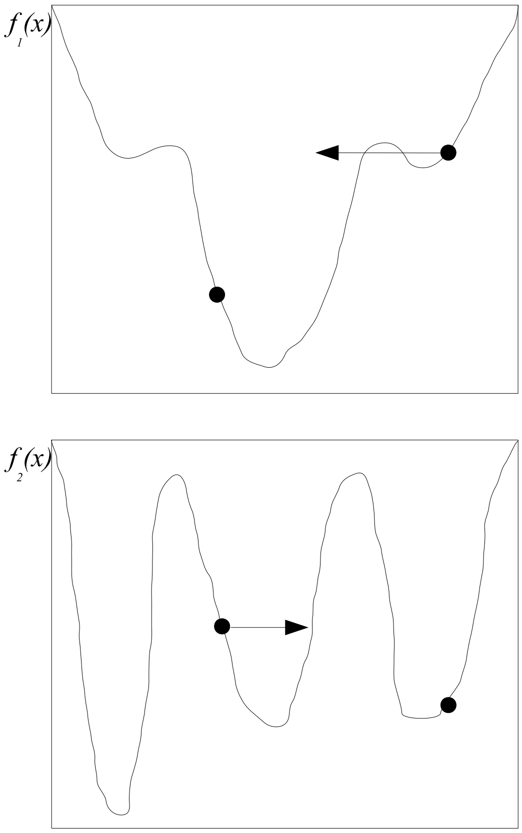


Figure 4.1: Sample fitness landscapes illustrating how the landscape structure can result in deception for a PSO algorithm.

approximate measures based on samples from the search space are used to predict particular features of the landscape, such as ruggedness, in the hope that this will in some way capture the level of deception of the landscape for a search algorithm. Techniques for measuring ruggedness include adaptive walks [71], autocorrelation measures [92, 177], correlation length [80], entropic measures [171, 172] and amplitude spectra [62]. All of these techniques assume a discrete representation. This chapter proposes an adaptation of Vassilev *et al.*'s [171, 172] entropic measures for continuous problems and a method for approximating a single measure of ruggedness from a fitness landscape.

The presence of funnels is related to a different aspect of the distribution of local optima that considers the ruggedness of the underlying landscape on a macro scale. A funnel is a global basin shape that consists of clustered local optima [151] (see Section

2.3.7). Sutton *et al.* [151] have shown empirically that PSO algorithms tend to stagnate early on multi-funnel landscapes. A technique for estimating the presence of funnels in a fitness landscape is Lunacek and Whitley's dispersion metric [84], which is described and implemented in this chapter.

A feature which is not normally considered in fitness landscape analysis is the steepness of gradients. Where ruggedness refers to whether there are variations in neighbouring fitness values or not, the steepness of gradients takes into account the magnitude of fitness changes of neighbouring points. The motivation for considering gradients is that a landscape with steep gradients should have a higher probability of being deceptive to search (as is the case in the bottom landscape of Figure 4.1). This chapter also proposes a technique for estimating gradients.

The outline of the chapter is as follows: Section 4.2 discusses algorithms for performing random walks in continuous spaces. Random walks are needed as a basis for ruggedness and gradient measures. Section 4.3 describes the one-dimensional benchmark problems that are used for testing the different techniques in the chapter to see if results are consistent with a visual inspection of the plotted functions. Section 4.4 describes a proposed approach to quantifying ruggedness of a fitness landscape based on entropy, Section 4.5 discusses a measure for predicting the presence of funnels and Section 4.6 explores gradient measures. Section 4.7 tests the techniques on higher dimensional benchmark problems and also investigates the link to PSO performance.

4.2 Random walk algorithms

Some fitness landscape analysis techniques are based on a random walk through the search space to capture a sequence of neighbouring solutions, such as autocorrelation techniques [177], correlation length [80] and entropic measures of ruggedness and smoothness with respect to neutrality [172]. Adapting these for real-valued problems requires a method for performing a random walk in a continuous space. This section proposes a random walk algorithm for continuous spaces with the following aims:

1. The points in a walk should be ordered based on some generic notion of neighbourhood, such as Euclidean distance.

2. Unlike a sample generated by a search algorithm, a random walk should be unbiased in that it should not use fitness information to direct the walk. This differs from the aim of other sampling strategies that use a particular search operator to define neighbourhood (as is the case with techniques that measure evolvability such as fitness clouds [173], negative slope coefficient [164], fitness-probability clouds [83] and accumulated escape probability [83]).

3. The set of walks used to sample the search space should attempt to provide as wide a coverage of the search space as possible within the constraints of acceptable computational cost. If the purpose of analysing a problem is to obtain information to guide the choice of an appropriate algorithm, then the computational cost of sampling should be significantly less than solving the problem with multiple algorithms using a trial-and-error approach.

This section looks at a simple random walk algorithm and shows that the coverage of the search space is not adequate. Section 4.2.2 proposes an alternative algorithm called a *progressive random walk* algorithm.

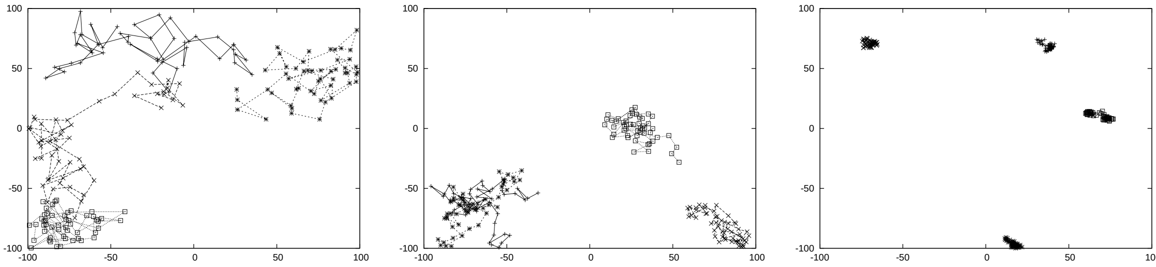
4.2.1 Simple random walk

A walk through a search space requires a notion of neighbourhood for any point in the space. For binary problems, a random walk could be implemented as follows [172]: start from a randomly chosen point, generate all neighbours of the current point by mutation (bit flip), choose randomly one neighbour as the next point, generate all neighbours of the new point, and so on. In the case of continuous search spaces, however, there is no equivalent set of all possible neighbours of any point. The neighbourhood of a multidimensional point $\mathbf{x}$ is normally defined as the set of points within the hypersphere with some small radius and centre $\mathbf{x}$ [152]. However, this approach requires Euclidean distance calculations to ensure that one point is in the neighbourhood of another point. To simplify the computational complexity of neighbourhood checking, a neighbourhood based on hypercubes is proposed. Formally, the neighbourhood set $N(\mathbf{x}_k)$ of an n-dimensional point $\mathbf{x}_k$ is defined as follows:

$$\mathbf{x}_j \in N(\mathbf{x}_k) \iff |x_{ki} - x_{ji}| < s, \ \forall i \in [1, \ldots, n] \tag{4.1}$$

where s is half of one length of the hypercube specifying the neighbourhood size. Note that this definition assumes that the neighbourhood size is equal in all dimensions. In some cases it may be desirable to define s, not as a scalar value, but as a multidimensional vector $\mathbf{s}$. For example, consider a two-dimensional search space where variables x_1 and x_2 have domains of $[0, 100]$ and $[0, 1]$, respectively. The neighbourhood could be specified with $\mathbf{s} = [10, 0.1]$, so that although the neighbourhood is technically a rectangle, it is a square relative to the search space, since each side of the neighbourhood is 10% of the domain.

Given the definition of neighbourhood in Equation 4.1, a simple approach to a random walk through a continuous space could be the following: start at a random position within the bounds of the multi-dimensional search space; take a step of random size and direction within the bounds of the multi-dimensional hypercube defining the neighbourhood, always ensuring that the walk stays within the outer bounds of the search space, until the required number of steps are reached. This simple random walk algorithm is expressed more formally in Algorithm 4.1 and Figure 4.2 plots the position vectors of sample runs for a two-dimensional space using different values of s. As can be seen, taking steps in random directions has a tendency for the points of a walk to be clustered in limited areas of the search space, which becomes more pronounced as the value of s decreases. This can result in poor coverage of the search space.



(a) $s = 20$ (b) $s = 10$ (c) $s = 2$

Figure 4.2: Plots of the position vectors of sample simple random walk runs for a two-dimensional space with differing values of s (step bound). Each sub-figure shows four independent sample walks of 50 steps each.

Algorithm 4.1 Simple Random Walk Algorithm

```
Let $p$ be the problem on which to base the walk, which encapsulates the number of dimensions, $n$, and domain for each dimension $[x_1^{min}, x_1^{max}], \ldots, [x_n^{min}, x_n^{max}]$
Let $numSteps$ specify the number of steps in the walk
Let $s$ specify the bound on the step size for each dimension
Create an array of $n$-dimensional vectors for storing the walk (called $walk$) of size $numSteps + 1$
for all dimension $i$ of $n$ do
    Generate a random number $r$ in the range $[x_i^{min}, x_i^{max})$
    Set $walk[0]_i$ to $r$
end for
for all step $s$ from 1 to $numSteps$ do
    for all dimension $i$ in $n$ do
        repeat
            Generate a random number, $r$, in the range $[-s, +s)$
        until $walk[s-1]_i + r$ is in bounds of search space
        Set $walk[s]_i = walk[s-1]_i + r$
    end for
end for
```

4.2.2 Progressive random walk

An alternative approach to a simple random walk, called a *progressive random walk* is proposed with the following basic idea: A walk starts on the edge of the multi-dimensional search space, progresses in a random way, but with a bias in direction towards the opposite side of the search space. If a search space boundary is reached, the bias is changed to the opposite direction. Multiple walks are generated from different random starting positions on the outer boundaries of the search space. The details of the approach are specified below.

Starting zones for progressive random walks

For a one-dimensional search space, there are only two possible starting positions on the edge of the search space: the minimum and maximum points defining the problem domain. In two dimensions, there are four lines of boundary points; in three dimensions there are six planes of boundary points; and so on. In an attempt to provide a wide coverage of the search space, the set of points on the boundary is divided into non-overlapping zones, so that multiple random walks can start in different portions of the outer boundary. The proposed approach to dividing the boundary points into zones is described in Table 4.1, where the domain for each dimension i is specified as $[x_i^{min}, x_i^{max}]$. The starting zones for two-dimensional and three-dimensional spaces are illustrated in Figures 4.3 and 4.4, respectively.

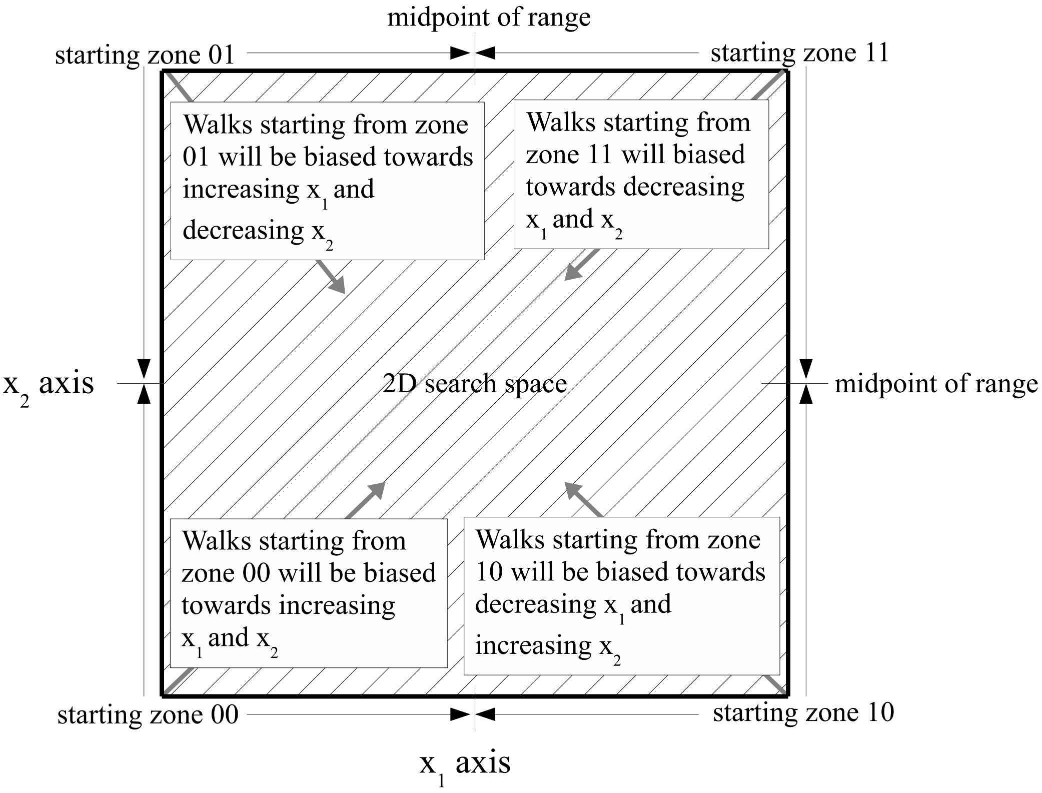

midpoint of range
starting zone 01
starting zone 11
Walks starting from zone 01 will be biased towards increasing x1 and decreasing x2
Walks starting from zone 11 will biased towards decreasing x1 and x2
x2 axis
2D search space
midpoint of range
Walks starting from zone 00 will be biased towards increasing x1 and x2
Walks starting from zone 10 will be biased towards decreasing x1 and increasing x2
starting zone 00
starting zone 10
x1 axis

Figure 4.3: Four starting zones for a two-dimensional search space.

Table 4.1: Proposed starting zones of progressive random walks for different dimensions.

	Description of starting zones and random walk progression
1-D	Two starting zones, which are the single points (x^{min}), starting zone 0, and (x^{max}), starting zone 1.
2-D	Four starting zones, corresponding to each corner of the rectangular search space. Each starting zone is defined as the set of points falling on the two lines extending from the corner point up to the midpoints of the ranges along the axes, as illustrated in Fig. 4.3. Starting zone 00 refers to the lines extending from corner (x_1^{min}, x_2^{min}). Similarly, zones 01, 10 and 11 refer to the lines extending from corners (x_1^{min}, x_2^{max}), (x_1^{max}, x_2^{min}), and (x_1^{max}, x_2^{max}), respectively.
3-D	Eight starting zones (identified as zones 000 to 111 in binary) corresponding to the eight corners of the cuboid defining the search space. Each starting zone is defined as the set of points on the planes falling on the outer boundaries of the search space extending from the given corner up to the midpoints of the ranges of the axes, as illustrated in Figure 4.4.
n-D	2^n starting zones corresponding with the 2^n corner points, where each corner point is of the form $(c_1, \ldots, c_n)$, such that $\forall i \in [1, \ldots, n], c_i \in \{x_i^{min}, x_i^{max}\}$. Each starting zone is defined as the set of points falling on the outer boundary of the search space extending from the given corner point up to the midpoints on the ranges of the axes.

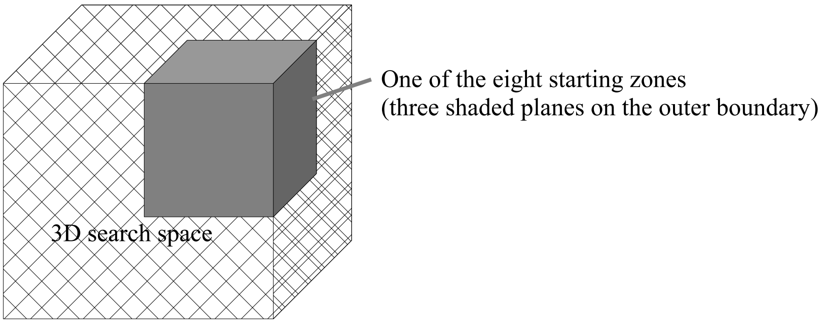

One of the eight starting zones
(three shaded planes on the outer boundary)
3D search space

Figure 4.4: One of the eight starting zones for a three-dimensional search space.

As described in Table 4.1, for an n-dimensional search space, there are 2^n non-overlapping starting zones. Each starting zone is identified using an n-bit string $b_1 \dots b_n$, which specifies the corner point $(c_1, \dots, c_n)$ of the starting zone as follows:

$$c_i = \begin{cases} x_i^{min} & \text{if } b_i = 0 \\ x_i^{max} & \text{if } b_i = 1 \end{cases} \tag{4.2}$$

A point $(x_1, \dots, x_n)$ is defined as being in the starting zone identified by the binary number $b_1 \dots b_n$ with corner point $(c_1, \dots, c_n)$ when the following hold:

$$\begin{aligned} &\forall i \in [1, \dots, n], \quad |x_i - c_i| < \tfrac{x_i^{max} - x_i^{min}}{2}, \text{ and} \\ &\exists i \in [1, \dots, n], \quad \text{where} \quad x_i = c_i \end{aligned} \tag{4.3}$$

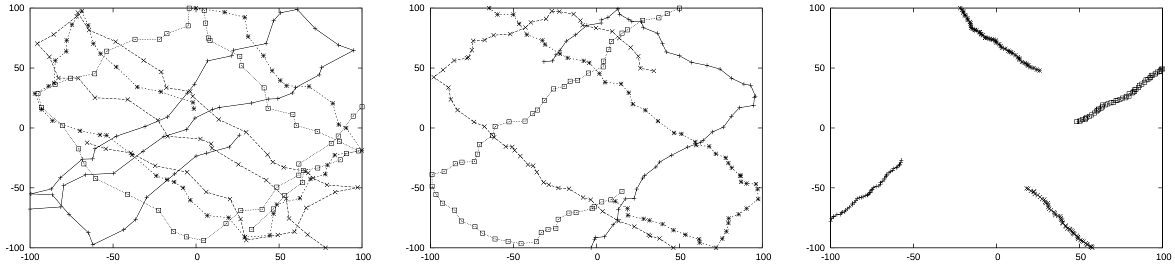
Progressive random walk algorithm

Given a binary number specifying the starting zone, a random progressive walk starts by generating a random position in the specified starting zone. Steps are then based on random offsets in each dimension, within the neighbourhood, in the direction of the opposite boundaries. The algorithm is expressed more formally in Algorithm 4.2, while Figure 4.5 plots the position vectors generated by sample runs of the algorithm for a two-dimensional space using different values of s. The computational time complexity of Algorithm 4.2 (a single progressive random walk) is simply linear with respect to the number of steps in the walk. However, if multiple walks are performed in multi-dimensional space, where the number of walks equals the number of starting zones, then the time complexity of multiple walks increases exponentially with the dimension. An alternative approach to multiple walks with linear complexity with respect to the dimension is proposed in the next section.

It can be seen in Figure 4.5, in contrast to Figure 4.2, that a better coverage is obtained by the progressive random walks than the simple random walks in two dimensions. It can also be seen that a step bound of 20 (10% of the range of the domain on a single dimension) provides a better coverage of the space than smaller bounds, while still maintaining a reasonably close proximity between points on the walk. Figure 4.6 contrasts sample runs from simple random walks and progressive random walks for a three-dimensional problem space.

Algorithm 4.2 Progressive random walk algorithm

```
Let $p$ be the problem on which to base the walk, which encapsulates the number of dimensions, $n$, and domain for each dimension $[x_1^{min}, x_1^{max}], \ldots, [x_n^{min}, x_n^{max}]$
Let $numSteps$ specify the number of steps in the walk and $s$ the bound on the step size for each dimension
Let $startingZone$ be a binary array of size $n$ (a bit for each dimension) specifying the starting zone of the walk and how the walk should progress
Create an array of $n$-dimensional vectors for storing the walk (called $walk$) of size $numSteps + 1$
**for all** dimension $i$ of $n$ **do**
    Generate a random number $r$ in the range $[0, \frac{x_i^{max} - x_i^{min}}{2})$
    **if** $startingZone_i$ equals 1 **then** set $walk[0]_i$ to $x_i^{max} - r$
    **else** set $walk[0]_i$ to $x_i^{min} + r$
    **end if**
**end for**
Generate a random dimension, $rD$, in range $[0, \ldots, n]$
**if** $startingZone_i$ equals 1 **then** set $walk[0]_{rD}$ to $x_i^{max}$
**else** set $walk[0]_{rD}$ to $x_i^{min}$
**end if**
**for all** steps $s$ from 1 to $numSteps$ **do**
    **for all** dimension $i$ in $n$ **do**
        Generate a random number, $r$ in the range $[0, s)$
        **if** $startingZone_i$ equals 1 **then** set $r$ to $-r$
        **end if**
        Set $walk[s]_i = walk[s-1]_i + r$
        **if** $walk[s]_i$ is out of bounds **then**
            Set $walk[s]_i$ to mirrored position inside boundary
            Flip bit $startingZone_i$
        **end if**
    **end for**
**end for**
```



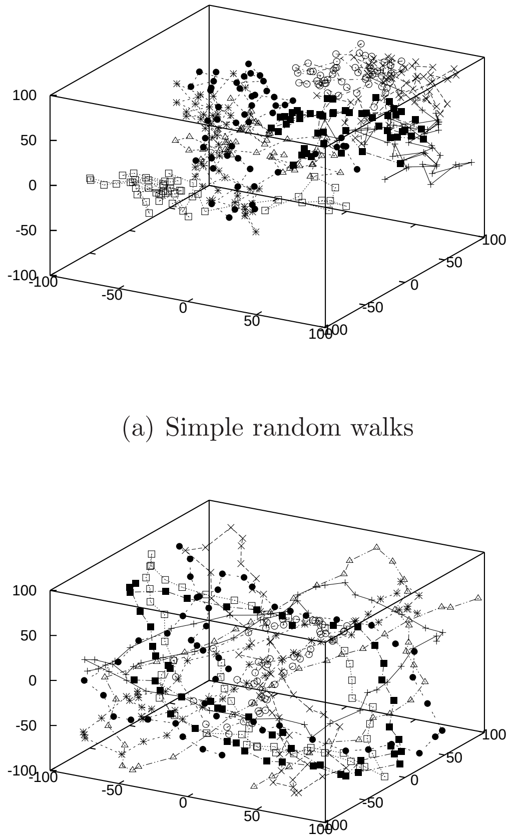
(a) $s = 20$ (b) $s = 10$ (c) $s = 2$

Figure 4.5: Plots of the position vectors of sample progressive random walk runs for a two-dimensional space with differing values of s (step bound). Each sub-figure shows four sample walks of 50 steps each, each starting in a different starting zone. Wider coverage of the search space is obtained than in the case of the simple random walks, as illustrated in Figure 4.2.

4.2.3 Testing coverage of walks

This section proposes a technique for quantifying the coverage of a sample in continuous space in terms of deviation from the distribution of a uniform sample. A histogram is a standard technique for visualising and estimating the distribution of a sample. For example, Figure 4.7 shows the distributions of three samples of 10 000 points in a two-dimensional space with domain $[-100, 100]$. The frequencies are based on 100 (10×10) bins of equal size, resulting in a mean of 100 points per bin. Figure 4.7(a) shows the distribution of a uniform random sample of the space. It can be seen that the frequencies deviate slightly from the mean of 100. Figures 4.7(b) and 4.7(c) show the distributions of samples resulting from simple random and progressive random walks respectively. In both cases the samples were generated from four equal-length walks with a step bound of 20 (10% of the domain), with the progressive random walks starting in different starting zones. As can be seen, the distribution of samples produced by the progressive random walks is more similar to a uniform distribution than the simple random walks. The problem of clustering of points in the search space is clearly evident in the histogram of the simple random walks.

To test the coverage of the random walk algorithms in higher dimensions, experiments were conducted on search spaces with domain $[-100, 100]$ for dimensions (D) 1, 2, 3, 4, 6 and 10. Sample sizes were chosen to be in the order of $10000 \times D$ (the maximum number of



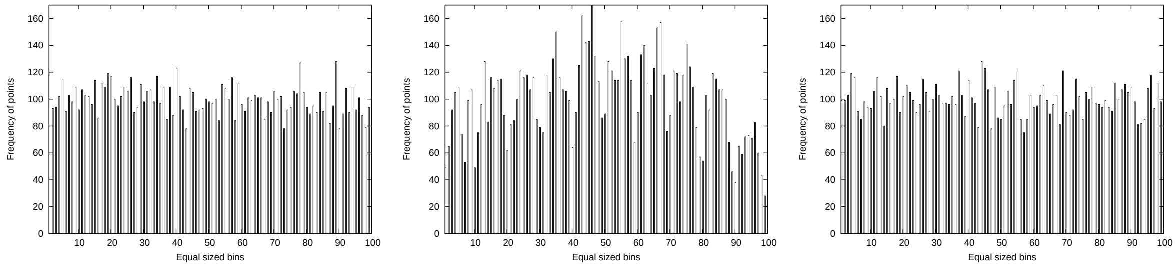
(a) Simple random walks

(b) Progressive random walks

Figure 4.6: Sample random walks in a three-dimensional space contrasting the coverage of simple random walks with progressive random walks. In both cases eight walks were generated of 50 steps each and with a maximum step size of 20.

function evaluations sometimes used in real-parameter optimisation competitions [150]), but adjusted to allow for a set mean of 100 points in a bin, while also ensuring equal sized bins (k^D for some integer k). The number of points and bins for each dimension is given in Table 4.2.

For the two random walk algorithms, 30 samples were generated through independent runs of the algorithm for each dimension. Each run consisted of 2^D walks using a step bound of 20 (10% of the domain). For the progressive random walk each walk of a sample



(a) Uniform sample (b) Simple random walks (c) Progressive random walks

Figure 4.7: Histograms showing the distribution of samples of 10 000 points in a two-dimensional space. Frequency is in terms of numbers of points in 10 × 10 equal-sized bins.

started in a different starting zone. For each run in a sample, the standard deviation of the frequency in the bins was calculated and the mean of the standard deviations over the 30 runs was calculated. The deviations based on uniform random samples were also calculated and the results are given in Table 4.2. As can be seen, a uniform random sample on average deviates by approximately 10 points in a bin from the mean of 100 in all dimensions. A sample generated by a simple random walk has much higher average deviations, ranging from approximately 20 in 1 dimension to over 30 in 10 dimensions. Samples generated by progressive random walks have significantly smaller deviations on average than simple random walks.

The results in Table 4.2 show that a progressive random walk sample is more uniformly distributed than a simple random walk sample, but not as uniformly distributed as a uniform random sample. The advantage of progressive random walk samples over uniform random samples is that additional information is provided in the form of the neighbourhood captured in the sequence of points in the sample.

Multiple random walks in multi-dimensional space

In multi-dimensional spaces, the size of the search space increases exponentially as the dimension increases. When performing a random walk, larger search spaces require more walks to sample the space. When sampling an n-dimensional space using a random walk, the number of walks should ideally be equal to the number of starting zones, that is 2^n, as described in Table 4.1. This is clearly infeasible for high dimensions (for example, for

Table 4.2: Quantifying coverage of a sample using average standard deviations from the mean frequency in a bin (mean equals 100). Values are based on 30 runs, with standard deviations shown in brackets. D is the dimension of the space.

D	Number of points in sample	Number of bins	Average standard deviations from the mean of 100		
			Uniform random sample	Simple random walk	Progressive random walk
1	10 000	100	10.21 (±0.55)	20.02 (±4.34)	9.59 (±0.69)
2	19 600	196 (14^2)	10.03 (±0.64)	24.75 (±2.48)	12.75 (±2.33)
3	34 300	343 (7^3)	10.06 (±0.40)	26.36 (±1.88)	16.47 (±2.57)
4	62 500	625 (5^4)	9.93 (±0.27)	26.56 (±1.35)	18.34 (±1.59)
6	72 900	729 (3^6)	10.07 (±0.25)	31.07 (±1.47)	23.86 (±1.17)
10	102 400	1024 (2^{10})	9.98 (±0.26)	33.61 (±0.89)	19.67 (±0.62)

a 30-dimensional space $2^{30} = 1073741824$ walks), especially since the aim is to provide a characterisation of a problem that is less computationally expensive than actually solving the optimisation problem.

To keep within a limit of $10^3 \times n$ sample points as the basis for fitness landscape measures (10 times less than the maximum number of function evaluations commonly used in real-parameter optimisation competitions), the following approach is proposed for n-dimensional space:

- The number of walks performed is equal to the number of dimensions, so n independent walks are performed with each walk starting in a different starting zone. This ensures a linear growth in computation time as the dimension increases.

- To distribute different walks across the starting zones, every $(\frac{2^n}{n})^{th}$ starting zone is used as the starting point for a walk. For example, for a 4-dimensional space with 16 starting zones, 4 walks will be performed starting in zones 0000, 0100, 1000 and 1100.

Note that any sampling of a high dimensional space will provide inadequate coverage of the search space. The aim is to have a random sampling technique that can be used as the basis of measures that are reasonably reliable (that is, the resulting characterisation measures do not have very large standard deviations).

4.3 Simple benchmark problems

The one-dimensional benchmark problems used to test the proposed measures for the remainder of this chapter are defined in Table 4.3 and are plotted in Figure 4.8. The nine functions exhibit different characteristics in terms of ruggedness, gradients and funnels. For example, the Table Legs function was constructed as a problem with high neutrality and therefore very little ruggedness. Despite two points with infinite gradients, these are offset by zero gradients for the rest of the domain. The two minimum plateaus at the extremes of the domain make the Table Legs function multi-funnelled. The functions and their characteristics are discussed in the relevant sections where they are used to test the proposed measures.

4.4 Ruggedness measures

Aspects of this section were published as a paper in the proceedings of the IEEE Congress on Evolutionary Computation [87].

4.4.1 Measuring ruggedness using entropy

Ruggedness refers to the number and distribution of local optima. Entropy is a measure of the uncertainty involved in sampling from a source [135]. Given a sample of fitness values resulting from a random walk on a problem landscape, and using a suitable encoding, an estimate of entropy of this sample could form the basis for an estimate of the ruggedness of the problem landscape. Vassilev *et al.* [170, 171, 172] propose such an information theoretic technique for studying the structure of discrete landscapes in terms of their smoothness, ruggedness and neutrality. This study is limited to discussing their approach to estimating ruggedness (and not smoothness) with respect to neutrality.

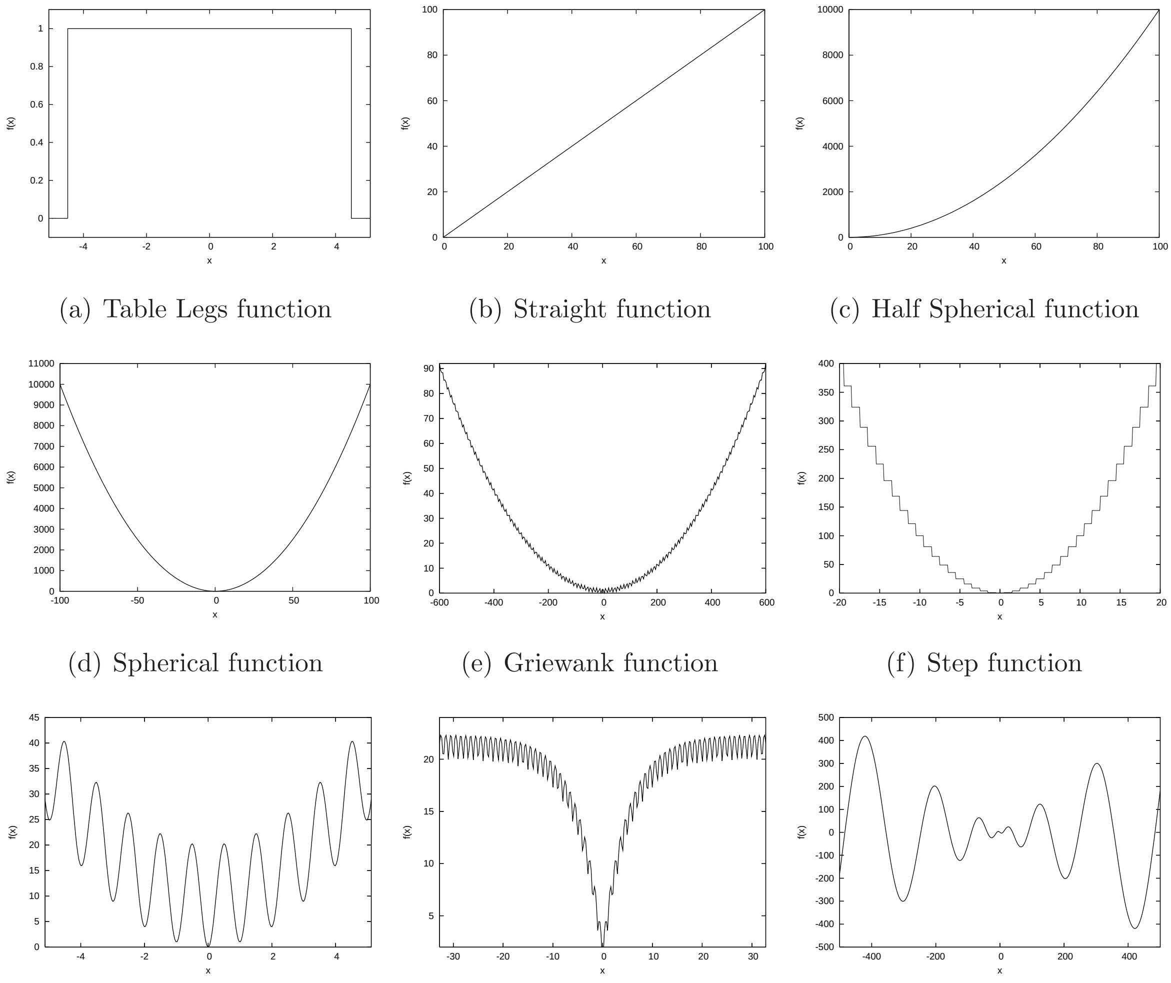
Table 4.3: One-dimensional benchmark functions for ruggedness, gradients and funnels. Plots of these functions are shown in Figure 4.8.

Table Legs	$f(x) = 0$, where $x \in [-5, -4.5)$ $= 1$, where $x \in [-4.5, 4.5]$ $= 0$, where $x \in (4.5, 5]$
Straight	$f(x) = x, \qquad x \in [0, 100]$
Half Spherical	$f(x) = x^2, \;\; x \in [0, 100]$
Spherical	$f(x) = x^2, \;\; x \in [-100, 100]$
Griewank	$f(x) = \frac{1}{4000}x^2 - \cos\left(\frac{x}{\sqrt{1}}\right) + 1, \;\; x \in [-600, 600]$
Step	$f(x) = (\lfloor x + 0.5 \rfloor)^2, \;\; x \in [-20, 20]$
Rastrigin	$f(x) = x^2 - 10\cos(2\pi x) + 10, \; x \in [-5.12, 5.12]$
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{x^2}\right) - \exp\left(\cos(2\pi x)\right) + 20 + e, \;\; x \in [-32, 32]$
Schwefel 2.26	$f(x) = -(x\sin(\sqrt{\lvert x \rvert}), \;\; x \in [-500, 500]$

The basic idea is to obtain a landscape path by performing a random walk on the landscape. This path is represented as an ensemble of three-point objects, where each object is a point on the path together with its neighbours. Each three-point object is classified as one of the following:

- neutral (a point together with its neighbours has equal fitness values);
- smooth (the fitness differences between the three points changes in one direction: a slope);
- rugged (the fitness differences between the three points changes in two directions: a peak, valley or step).

Table 4.4 gives all possible shapes of three-point objects with their classification. Deciding whether a given three-point object is neutral, smooth or rugged would depend



(a) Table Legs function (b) Straight function (c) Half Spherical function

(d) Spherical function (e) Griewank function (f) Step function

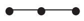
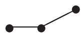
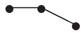
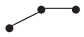
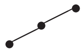
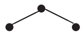
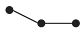
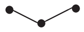
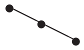
(g) Rastrigin function (h) Ackley function (i) Schwefel 2.26 function

Figure 4.8: One-dimensional benchmark functions for testing the ruggedness, dispersion and gradient measures. Definitions of these functions are given in Table 4.3.

on the margin of error used to determine whether two values are ‘equal’ or not. By increasing this margin of error, a larger number of objects (such as very small steps, or very shallow valleys) could be regarded as neutral.

Given a sample of three-point objects obtained from a random walk through the search space and a classification as described in Table 4.4, an information function is used to estimate the entropy of the subset of objects that are rugged. By increasing the margin of error used to determine whether two fitness values are ‘the same’ or not,

Table 4.4: Classification and encoding of three-point objects (a point on a path together with its neighbours) as neutral, rugged or smooth based on relative fitness values.

Object shape	Classification	Encoding
	neutral	0 0
	rugged	0 1
	rugged	0 $\bar{1}$
	rugged	1 0
	smooth	1 1
	rugged	1 $\bar{1}$
	rugged	$\bar{1}$ 0
	rugged	$\bar{1}$ 1
	smooth	$\bar{1}$ $\bar{1}$

the information function is used to reveal how the measure of entropy changes as the neutrality of the landscape is increased. The following section provides a formalisation of this approach.

Formalisation of approach

This section describes Vassilev *et al.*'s [172] approach to estimating ruggedness with respect to neutrality. Assume that a random walk on a landscape generates the time series of fitness values $\{f_t\}_{t=0}^{n}$. This time series is represented as a string, $S(\varepsilon) = s_1 s_2 s_3 ... s_n$, of symbols $s_i \in \{\bar{1}, 0, 1\}$, obtained by the function:

$$\begin{aligned} s_i = \Psi_{f_t}(i, \varepsilon) &= \bar{1}, \text{ if } f_i - f_{i-1} < -\varepsilon \\ &= 0, \text{ if } |f_i - f_{i-1}| \leq \varepsilon \\ &= 1, \text{ if } f_i - f_{i-1} > \varepsilon \end{aligned} \tag{4.4}$$

The parameter ε is a real number that determines the accuracy of the calculation of string $S(\varepsilon)$. A low value for ε would result in a high sensitivity to the differences between neighbouring fitness values. Based on this definition of the string $S(\varepsilon)$, an entropic measure $H(\varepsilon)$ is defined as follows:

$$H(\varepsilon) = -\sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]} \tag{4.5}$$

where p and q are elements from the set $\{\bar{1}, 0, 1\}$, $P_{[pq]}$ is defined as

$$P_{[pq]} = \frac{n_{[pq]}}{n} \tag{4.6}$$

and $n_{[pq]}$ is the number of sub-blocks pq in the string $S(\varepsilon)$.

The formula for $H(\varepsilon)$ is an instance of the more general b-ary entropy [135]. The result is a value in the range $[0, 1]$ and is an estimate of the variety of 'shapes' in the walk [171]. Note that Equation 4.5 only considers the subset of elements that are rugged (sub-blocks pq in the string $S(\varepsilon)$ where $p \neq q$). The base of the logarithmic function is 6, because there are 6 possible rugged shapes (see Table 4.4). For each rugged element, Equation 4.6 calculates the probability of that element occurring. Therefore, the higher the value of $H(\varepsilon)$, the more the variety of rugged shapes in the walk and so the more rugged the landscape.

The parameter ε determines how sensitive Ψ_{f_t} (Equation 4.4) is to differences in fitness values. By increasing the value of ε the neutrality of the landscape is increased. The smallest value of ε for which the landscape becomes flat (i.e. for which $S(\varepsilon)$ is a string of 0's) is called the information stability and is denoted by ε^*. The value of ε^* would therefore equate to the largest difference in fitness values between any two successive points on the path.

Unlike other statistical approaches to measuring ruggedness (such as correlation functions [147, 177]), $H(\varepsilon)$ is not a measure of the ruggedness of the landscape, but rather an indication of the relationship between the ruggedness and neutrality.

4.4.2 Modifications for continuous landscapes

The calculation of entropic measure $H(\varepsilon)$ is based on a random walk through the landscape resulting in a sequence of fitness values. The progressive random walk as described in Section 4.2.2 is proposed as the basis for the entropic ruggedness measure. The random walk generates a time series of fitness values $\{f_t\}_{t=0}^{n}$. This time series can be represented as a string $S(\varepsilon)$ of symbols taken from $\{\bar{1}, 0, 1\}$. In Vassilev *et al.*'s definition [171], the string $S(\varepsilon)$ is considered with periodic boundary conditions (based on the assumption that the landscape is statistically isotropic). This means that a walk of n steps results in a string $S(\varepsilon)$ of n symbols, because the last fitness value in the walk is compared back with the first fitness value of the walk to determine the final symbol of $S(\varepsilon)$. For continuous landscapes, the definition of $S(\varepsilon)$ was modified to not 'wrap around' in this way. A walk of n steps would therefore result in a string $S(\varepsilon)$ of $n - 1$ symbols.

4.4.3 Testing entropic measure on one-dimensional functions

The approach proposed in Section 4.4.2 was tested on seven of the one-dimensional benchmark functions defined in Section 4.3 (the Straight and Half Spherical functions were left out due to similar levels of ruggedness to the Spherical function). The following summarises the expected levels of ruggedness of the seven functions:

- Table Legs function: This benchmark is mostly neutral, so should give the lowest values for ruggedness.
- Spherical function: This function is very smooth, so should give relatively low values for ruggedness.
- Step function: This function has high neutrality (due to the flat sections) and high ruggedness.
- Rastrigin and Schwefel 2.26: Both of these functions have fairly high ruggedness.
- Griewank and Ackley: These functions are both extremely rugged on a fine scale.

For each of these seven one-dimensional benchmark problems the following was done:

- A progressive random walk (using Algorithm 4.2) of 1000 steps of size $\frac{xRange}{100}$ was performed, where $xRange$ is the difference between the maximum and minimum x values.

- The information stability measure (ε^*) for the given walk was estimated by trying increasingly smaller values of ε up to a precision of 0.001. Recall that ε^* is the smallest value of ε for which the landscape (as represented by the walk) becomes flat. For each function this estimated value of ε^* was used as the upper bound on the range of sensible values for ε.

- $H(\varepsilon)$ was calculated for increasing values of ε from 0 to ε^* with intervals of $0.05 \times \varepsilon^*$.

A graph showing $H(\varepsilon)$ for different values of ε is shown in Figure 4.9. This graph illustrates the trend of how ruggedness changes with respect to neutrality. The use of relative ε-values that depend on the value of ε^* for each function allows for comparisons between benchmarks occupying very different fitness ranges. The following observations can be made on the behaviour of $H(\varepsilon)$:

- The value of ε^* is defined as the smallest value of ε for which the landscape becomes flat. This is when $S(\varepsilon)$ is a string of 0's. As can be seen on the graph in Figure 4.9, the value of $H(\varepsilon^*)$ for all functions is 0, which is as expected.

- In the case of all benchmark functions, except for Table and Step, $H(\varepsilon)$ is an increasing function for small values of ε. The least rugged and most neutral of the functions is the Table Legs function, which has a low value for $H(0)$ and maintains this value until just before ε^*. The only other function with high neutrality is the Step function, which has its highest value for $H(\varepsilon)$ where $\varepsilon = 0$. Figure 4.10 illustrates, using a very simple example, how the value of $H(\varepsilon)$ can increase with an increase in ε. Assume a straight walk on function $f(x) = x^2$ produces the time series of fitness values: $f(-2) = 4, f(-1) = 1, f(0) = 0, f(1) = 1, f(2) = 4$. The value of $S(\varepsilon = 0) = \bar{1}\ \bar{1}\ 1\ 1$ (a landscape with a single rugged element $\bar{1}\ 1$). This results in a value for $H(0)$ of $-\frac{1}{3}\log_6\frac{1}{3} = 0.204$. When ε increases to 2, $S(\varepsilon = 2)$ changes to $\bar{1}\ 0\ 0\ 1$, since the difference between $f(-1)$ and $f(0)$ and the difference between $f(0)$ and $f(1)$ are less than ε, so are regarded as no change. The increase in neutrality (by increasing ε) has resulted in a landscape with more 'difference' than the landscape represented by $S(\varepsilon = 0)$ and hence the value of $H(\varepsilon)$ increases. It is therefore only in the case of functions with low neutrality that $H(\varepsilon)$ is an increasing function for small values of ε. If a function has low

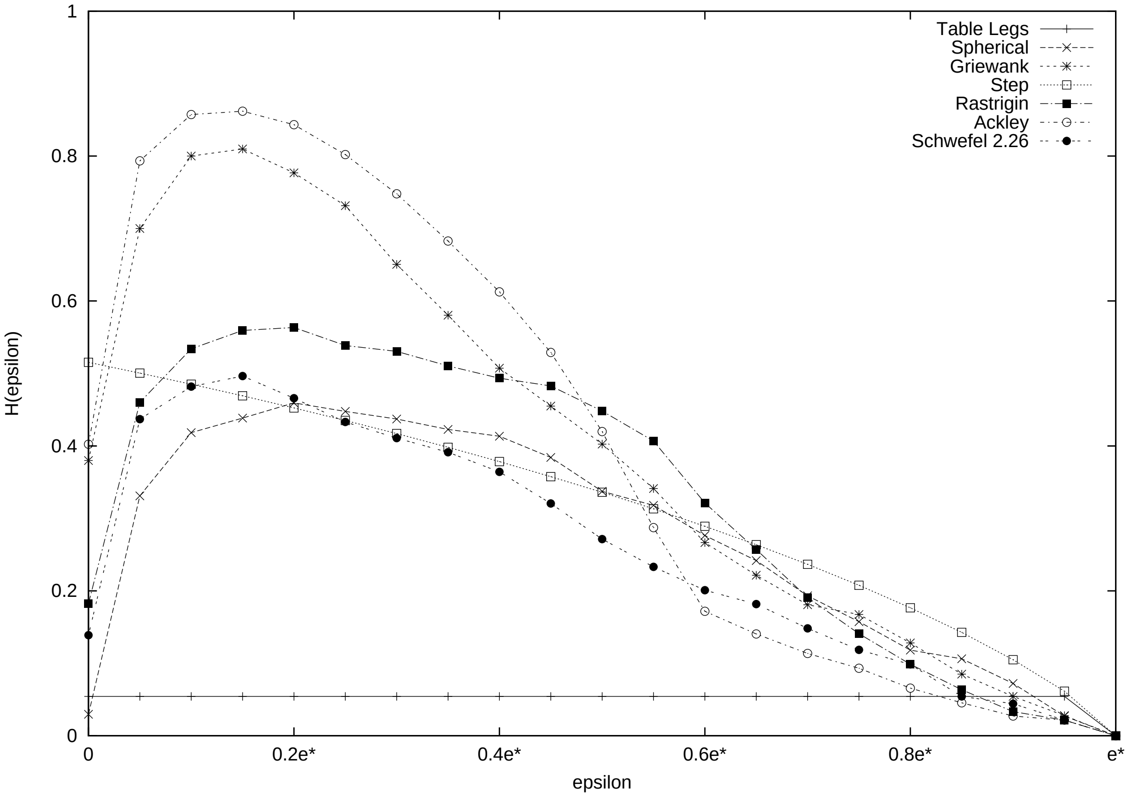


Figure 4.9: $H(\varepsilon)$ over different values of ε for seven different 1-dimensional functions, where the values of ε depend on ε^*

.

neutrality, increasing the neutrality (i.e. increasing the value of ε) introduces more rugged elements (steps) which were present in low proportions (or even absent) for lower values of ε.

- In the case of all functions with low neutrality, the value of $H(\varepsilon)$ peaks at a particular value for ε and then decreases after that. To understand this behaviour, consider the Spherical function: For small values of ε the path represented by $S(\varepsilon)$ is smooth except for the part where the progressive walk reaches the minimum point of the function. With larger values of ε the magnifying glass with which the discrete path through the landscape is investigated zooms out. Some successive points which previously were regarded as having different fitness values are now

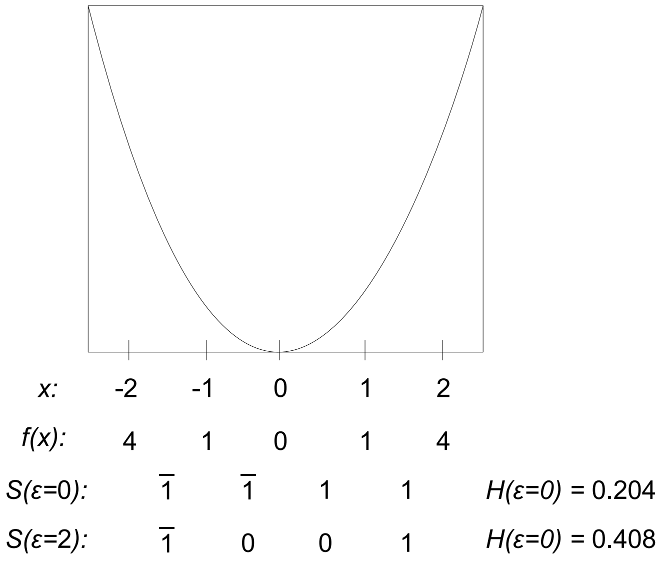


Figure 4.10: A simple illustration of how the value of $H(\varepsilon)$ can increase with an increase in ε

regarded as having the same fitness values. This results in the introduction of rugged elements (in the form of step shapes) into the analysis with a corresponding increase in the value of $H(\varepsilon)$. For greater values of ε, more and more neutral shapes are introduced until the path becomes perfectly neutral.

4.4.4 Proposed single measure of ruggedness

As discussed in Section 1.4.2, one of the requirements of each technique is that the result is numerical to facilitate data mining of generated data. For the characterisation of the ruggedness of a landscape as illustrated in Figure 4.9 to be useful, it needs to be reduced to a single scalar value. On inspection of the behaviour of $H(\varepsilon)$ for different values of ε, as shown in Figure 4.9, it would seem that a significant value for each function is the maximum value of $H(\varepsilon)$. The point at which the maximum of $H(\varepsilon)$ occurs would correspond to the level of magnification that produces the most 'difference' in the landscape in the form of the maximum number of rugged elements. To characterise the ruggedness of a function, the following single value measure of ruggedness, called FEM (first entropic measure) is proposed for a fitness landscape:

$$\text{FEM} \quad = \quad \max_{\forall \varepsilon \in [0, \varepsilon^*]} \{H(\varepsilon)\}$$

To approximate the theoretical value of FEM, the maximum of $H(\varepsilon)$ is calculated for all ε values in the set of values from 0 to ε^* at discrete intervals of $(0.05 \times \varepsilon^*)$, resulting in 21 values of ε (or 20 values, excluding the final value of ε^*, since $H(\varepsilon^*) = 0$). The detailed algorithm for FEM is given in Algorithm 4.3.

Algorithm 4.3 Algorithm for computing FEM based on the search space of a problem

```
Perform a progressive random walk through the search space of the problem with $ns$
    steps using a step bound $s$
Set $\varepsilon Current$ and $\varepsilon Next$ to 0
repeat
    Set $\varepsilon Current$ to $\varepsilon Next$
    Determine the string $S(\varepsilon Current)$ using Equation 4.4
    Increase $\varepsilon Next$ by some increment
until the string $S(\varepsilon Current)$ is all zeros (i.e. the landscape is flat)
Set $\varepsilon^*$ to $\varepsilon Current$
for all $\varepsilon$ from 0 to $\varepsilon^*$ (non-inclusive) in increments of $(0.05 \times \varepsilon^*)$ do
    Determine the string $S(\varepsilon)$ using Equation 4.4
    Compute $H(\varepsilon)$ using Equation 4.5
end for
Return the maximum of all $H(\varepsilon)$
```

4.4.5 The influence of the step bound parameter on FEM

To investigate the influence of the step bound on the FEM measure, 30 independent runs of the FEM algorithm were performed for six different step bounds on each of the seven one-dimensional problems used before (in Section 4.4.3). The results are shown in Table 4.5 and the mean values are plotted in Figure 4.11. From Table 4.5 and Figure 4.11 it can be seen that two of the functions, Ackley and Griewank, have the highest mean FEM values at the smallest step bound of 0.01 (1% of the domain). These are

Table 4.5: Mean FEM ruggedness values for one-dimensional benchmark functions using different step bounds. Standard deviations are shown in parentheses below the means.

Function	Mean FEM values for different step bounds (proportion of range)					
	0.01	0.025	0.05	0.075	0.1	0.25
Table Legs	0.059 (±0.004)	0.122 (±0.000)	0.206 (±0.003)	0.276 (±0.004)	0.331 (±0.005)	0.403 (±0.018)
Spherical	0.463 (±0.013)	0.479 (±0.015)	0.501 (±0.010)	0.522 (±0.012)	0.546 (±0.009)	0.664 (±0.011)
Griewank	0.785 (±0.014)	0.619 (±0.015)	0.528 (±0.014)	0.528 (±0.010)	0.545 (±0.012)	0.669 (±0.010)
Step	0.512 (±0.005)	0.691 (±0.006)	0.596 (±0.011)	0.547 (±0.012)	0.549 (±0.010)	0.671 (±0.012)
Rastrigin	0.543 (±0.009)	0.705 (±0.007)	0.879 (±0.007)	0.901 (±0.008)	0.881 (±0.007)	0.863 (±0.008)
Ackley	0.876 (±0.010)	0.837 (±0.013)	0.796 (±0.011)	0.786 (±0.011)	0.767 (±0.010)	0.754 (±0.015)
Schwefel 2.26	0.486 (±0.010)	0.577 (±0.010)	0.709 (±0.008)	0.782 (±0.008)	0.820 (±0.010)	0.862 (±0.008)

the functions that display the most ruggedness on a micro scale (evident in the plots of Figure 4.8). Other functions, such as Rastrigin and Schwefel 2.26 have higher FEM values at larger step bounds and these are the functions that display ruggedness on a more macro scale. It is interesting to observe in Figure 4.11 how the FEM values of the three functions Step, Griewank and Spherical converge on the same FEM value from step bound 0.1 (10% of the domain). It is very clear from the plots in Figure 4.8 that these three functions have identical macro shapes. The values of FEM at step bounds of 0.01 and 0.1 therefore capture different information on the ruggedness of the functions. To capture these different levels of detail of ruggedness, it is proposed that two FEM measures are used: one for estimating micro ruggedness and the other for estimating macro ruggedness. These two measures are summarised in Table 4.6.

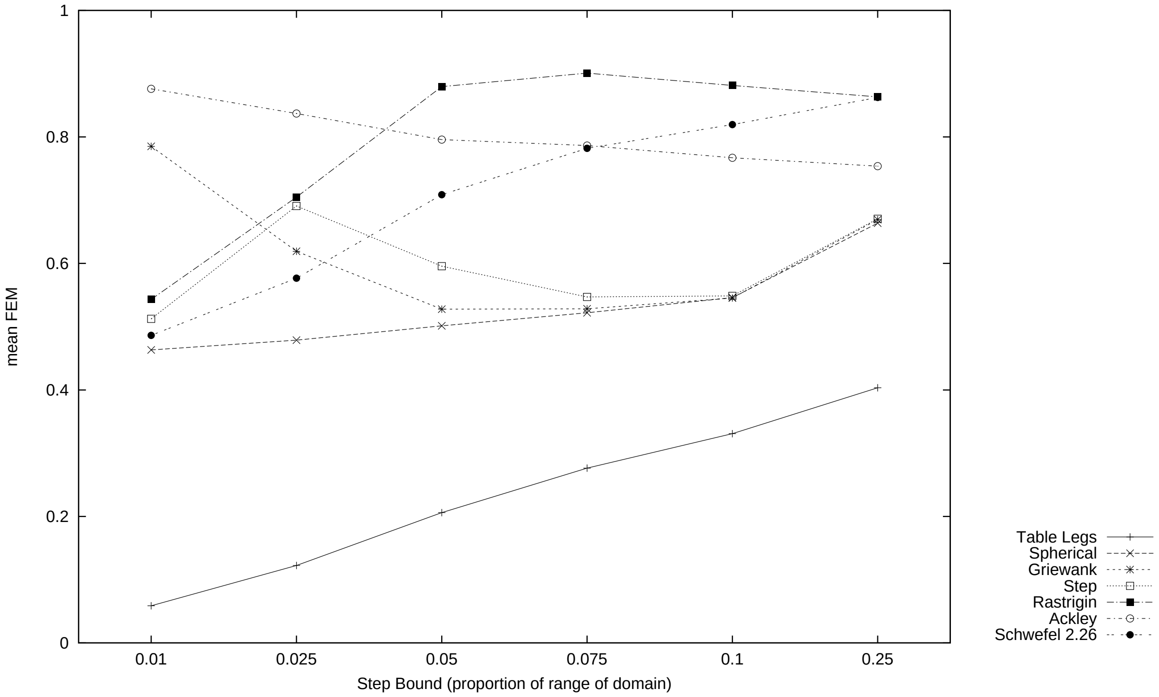


Figure 4.11: Plot of mean FEM ruggedness values for one-dimensional benchmark functions using different step bounds.

Table 4.6: Proposed measure of ruggedness based on random walks

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
$\text{FEM}_{0.01}$ $\text{FEM}_{0.1}$	First entropic measures of micro and macro ruggedness.	Number of steps in the random walk, ns. Step bounds of 1% and 10% of the range of the domain used for $\text{FEM}_{0.01}$ and $\text{FEM}_{0.1}$, respectively.	$[0, 1]$: where 0 indicates a flat landscape and 1 indicates maximal ruggedness.	$ns + 1$ function evaluations and 20 entropy calculations (for each discrete ε).

4.5 Estimating the presence of funnels

If one views the entire range of the fitness landscape of a one-dimensional problem, such as those illustrated in Figure 4.8, many of the standard benchmark functions have an underlying unimodal structure with a single funnel. A funnel in a landscape is a global basin shape that consists of clustered local optima [151]. Of the functions illustrated in Figure 4.8, only the Table Legs and Schwefel 2.26 functions have more than one funnel. This section investigates a measure of funnels.

4.5.1 Dispersion metric

Lunacek and Whitley [84] introduced the *dispersion metric* as a way of estimating the global topology of fitness landscape (and indirectly, the presence of multi-funnels). The *dispersion* of a sample of points refers to how spread out the points are (calculated as the average pair-wise distance between all points in the subset), while the dispersion function is based on the way in which the dispersion changes between a sample of solutions and a subset of solutions that are better [84]. A *low dispersion function* is one where the dispersion of a sample of points from the search space is larger than the dispersion of a subset of better solutions. This phenomenon is illustrated through the Rastrigin function in Figure 4.12. In the top left graph, a subset of the search space is considered below some threshold fitness value (horizontal line). A sample of points in this subset will have a larger dispersion (that is, they will be more spread out), than a sample of points below some lower threshold value (graph on the top right), being a sample of better solutions.

On the other hand, a *high dispersion function* is one where the dispersion of a sample of points is the same or less than the dispersion of a subset of better solutions. The two lower graphs in Figure 4.12 illustrate the Schwefel 2.26 function as an example of a high dispersion function. In the bottom left graph, a sample of points below the fitness threshold (horizontal line) will have a lower dispersion than a sample of points below the decreased fitness threshold in the bottom right graph. A low dispersion function is one which probably has a single funnel (an underlying unimodal structure), while a high dispersion function is one which is probably a multi-funnelled landscape (an underlying multimodal structure).

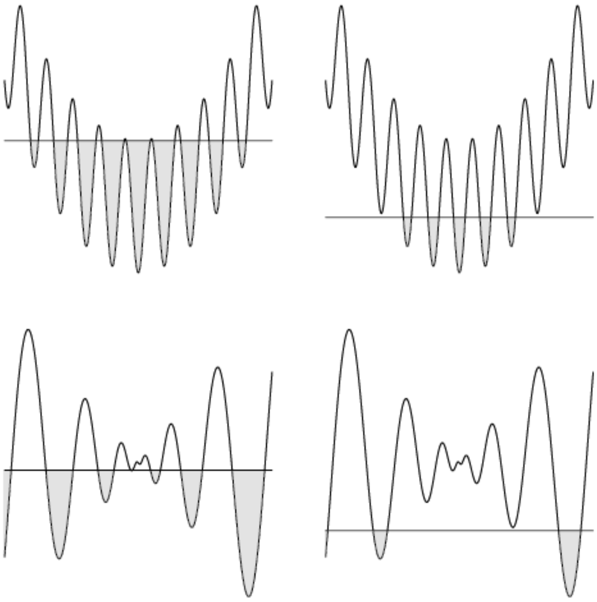


Figure 4.12: Replication of Figure 1 from [84] illustrating how a decrease in fitness threshold will result in a decrease in dispersion in the case of the Rastrigin function (top graphs) and an increase in dispersion in the case of the Schwefel 2.26 function (bottom graphs).

4.5.2 Proposed dispersion metric

The dispersion metric is a measure which captures the change in dispersion as the fitness threshold is decreased (assuming a minimisation problem). Algorithm 4.4 outlines the proposed approach to determine the dispersion metric of a problem and the properties of the measure are summarised in Table 4.7. The main difference from Lunacek and Whitley's [84] formalisation is that solution vectors are normalised to allow for comparison of dispersion metric values of problems with different domains. Note that step 6 of the algorithm and the result in Table 4.7 refer to a pre-determined constant value ($disp_D$) for the dispersion of a large uniform sample of a search space normalised to $[0, 1]$ in all dimensions. For example, the constant for a 1-dimensional space, $disp_1$ is approximately 0.33, while $disp_{30}$ is approximately 2.22. In the most extreme single-funnel scenario, the s best points are at the same position, so have a dispersion of 0, resulting in a dispersion metric (DM) value of $-disp_D$. In the most extreme multi-funnel scenario, s consists of

Algorithm 4.4 Algorithm for computing dispersion metric (DM)

```
Draw a uniform random sample $S$ of $n$ points (position vectors) from the $D$-dimensional search space of the problem.
Determine the fitness values of all $n$ points in $S$.
Determine the subset $S^*$ of the best $s$ points in $S$ based on fitness values.
Normalise the position vectors of the points in $S^*$, so that the domain of the search space is $[0, 1]$ for all dimensions $D$.
Calculate $disp(S^*)$ as the average pair-wise distance between the normalised position vectors in the subset $S^*$.
Return DM $= disp(S^*) - disp_D$, where $disp_D$ is a pre-determined constant value for the dispersion of a large uniform random sample in a $D$-dimensional search space, normalised to $[0, 1]$ in all dimensions.
```

only two points and these best points are positioned on two diagonally opposite points of the search space (any additional points in s will reduce the average pairwise distance). The length of the largest diagonal of a unit hypercube is $\sqrt{D}$, so the largest possible DM value is $\sqrt{D} - disp_D$.

Table 4.7: Proposed dispersion metric (DM)

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
DM	Dispersion metric.	(1) Size of sample, n, (2) Size of sub-sample of best solutions, s.	$[-disp_D, \sqrt{D} - disp_D]$: where $disp_D$ is the dispersion of a large uniform random sample of a D-dimensional space normalised to $[0, 1]$ in all dimensions. A positive value for DM indicates the presence of multiple funnels.	n fitness evaluations, sorting of n elements ($n\log(n)$) and $\frac{s^2-s}{2}$ Euclidean distance calculations.

4.5.3 Testing dispersion metric

Initial experiments were conducted on five of the one-dimensional benchmark functions illustrated in Figure 4.8, namely Table Legs, Spherical, Rastrigin, Ackley and Schwefel 2.26. The algorithm in Figure 4.4 was run using a sample size of 1000 with different values of s (the size of the subset of best solutions) from 0 to 1000. For each value of s the algorithm was run 30 times on each benchmark function and the mean DM values were calculated. The results are plotted in Figure 4.13. Notice how the value for DM converges on 0 when the percentage of the sample is 100 (s=1000, the full sample). From Figure 4.13 it can be seen that when the subset is above 10%, the DM value of the Table Legs and Schwefel 2.26 functions is positive. This is indicative of multiple funnels. All of the other functions have negative values for DM, which is as expected. When the percentage of the subset is too small, the multiple funnels of Schwefel 2.26 are not detected, as all the best solutions will be in the single funnel containing the global best (see Figure 4.12).

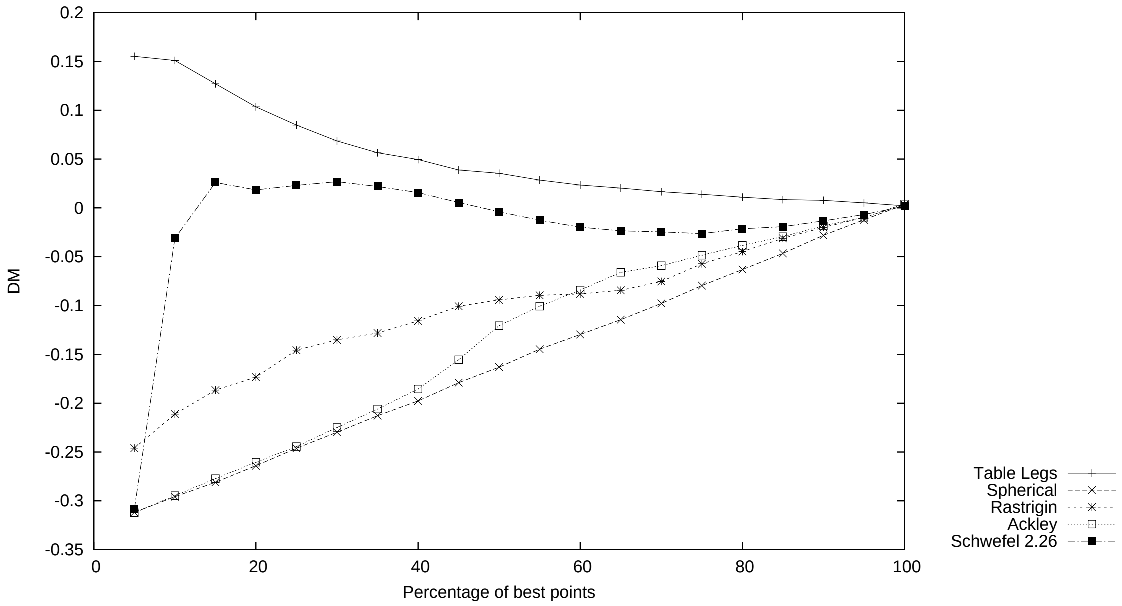


Figure 4.13: Mean DM values for one-dimensional benchmarks, showing the influence of s (the subset of best points in the sample).

Initial experiments on one-dimensional functions show that a sensible value for parameter s is approximately 10% of the full sample. This value is used in Section 4.7 when the proposed measures from the chapter are evaluated on higher dimensional benchmark problems.

4.6 Gradient measures

This section describes a proposed technique for estimating the steepness of gradients in a fitness landscape based on a particular form of random walk, called a Manhattan Progressive Random Walk, proposed in this study.

4.6.1 Proposed gradient measures

A walk through a multidimensional search space of a problem with fitness function f, starting at solution vector $\mathbf{x}(t)$ with T steps of equal size will result in the following sequence of $T+1$ points: $\mathbf{x}(t), \mathbf{x}(t+1), ..., \mathbf{x}(t+T)$. The norm of the gradient in fitness space between steps t and $t+1$ can be estimated by:

$$g(t) = \frac{f(\mathbf{x}(t+1)) - f(\mathbf{x}(t))}{dist(\mathbf{x}(t+1), \mathbf{x}(t))}. \tag{4.7}$$

where $dist(\mathbf{x}(t+1), \mathbf{x}(2))$ is the Euclidean distance between points $\mathbf{x}(t+1)$ and $\mathbf{x}(t)$.

A walk of T steps gives T gradients:

$$g(t), g(t+1), ..., g(t+(T-1)).$$

Given this sequence of gradient measures, a number of values can be calculated: The mean of all $g(t)$ values would give an indication of the average estimated gradient between neighbouring points on the walk, the standard deviation of $g(t)$ values from the mean would give an indication of how much variation there is in gradient estimations of the walk, and the maximum of all $g(t)$ values would quantify the biggest jump in fitness between neighbouring points. The definitions of these are as follows:

- The average estimated gradient within the walk is defined as:

$$\mathrm{G}_{avg} = \frac{\sum_{t=0}^{T-1} |g(t)|}{T} \tag{4.8}$$

The absolute value of each $g(t)$ is used, since the purpose is to quantify steepness, regardless of the direction of the slope. If the absolute values were not used, then negative slopes would cancel out positive slopes. If the walk provides a good sample of neighbouring points in the search space, the G_{avg} measure will provide an estimation of the gradient between neighbouring points across the whole search space.

- The standard deviation of gradient measures from the mean would give an indication of how much gradient measures on a walk differ from the average and is defined as:

$$\mathrm{G}_{dev} = \sqrt{\frac{\sum_{t=0}^{T-1}(\mathrm{G}_{avg} - |g(t)|)^2}{T-1}} \tag{4.9}$$

  The G_{dev} measure would be low if G_{avg} is a good estimator of the gradient on the walk. If, however, the G_{dev} measure is high, then it would be indicative of 'cliffs' or sudden steep 'valleys' or 'peaks' that stand out in contrast to the rest of the fitness landscape, or the presence of neutral areas that differ from the rest of the fitness landscape.

- The highest estimated gradient within the walk is defined as:

$$G_{max} = \max\{|g(t)|\}, \text{ for } t = 0, ..., T-1 \tag{4.10}$$

  If the value of G_{max} is much larger than the value of G_{avg}, it would be indicative of cliffs or sudden steep valleys/peaks that stand out from the rest of the landscape. If both G_{max} and G_{avg} are high, then this would be indicative of a highly rugged landscape where the fitness jumps are large.

Considered together, these three measures would provide some insight into the gradients of a fitness landscape as they could be experienced by an optimisation algorithm.

4.6.2 Manhattan progressive random walk

The gradient estimation measures require a random walk with steps of equal size. Enforcing steps of equal size across dimensions would involve computationally expensive

Euclidean distance calculations. A simple approach to ensuring a walk with steps of equal size is to take steps dimension-by-dimension, since a fitness change experienced by a step across dimensions would be made up of the individual fitness changes experienced in the individual dimensions. The progressive random walk proposed in Section 4.2.2 is based on steps of varying size. A modification of the progressive random walk is proposed, called a Manhattan progressive random walk with the following general idea: each step in a walk involves an offset of a predefined step size in only a single dimension, but this dimension is chosen randomly. The proposed algorithm for a Manhattan progressive random walk is given in Algorithm 4.5 and Figure 4.14 shows sample walks for a two-dimensional space. As with progressive random walks, Manhattan progressive random walks change direction when they ‘hit’ a boundary of the search space.

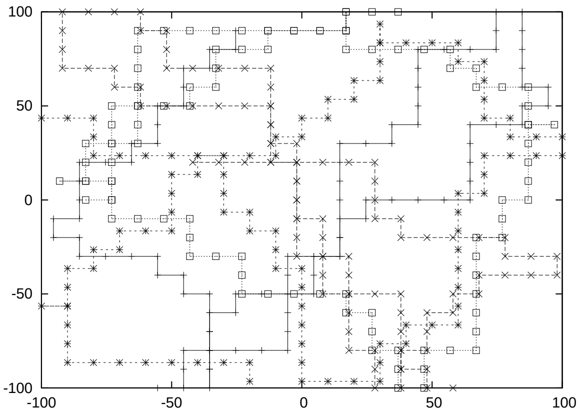


Figure 4.14: Plots of position vectors for four sample Manhattan progressive random walks starting in different starting zones. All walks are 100 steps long of step size 10.

Using steps of equal size in only a single random dimension at a time, Equation 4.7 can be replaced by:

$$g(t) = \frac{f(\mathbf{x}(t+1)) - f(\mathbf{x}(t))}{s}, \tag{4.11}$$

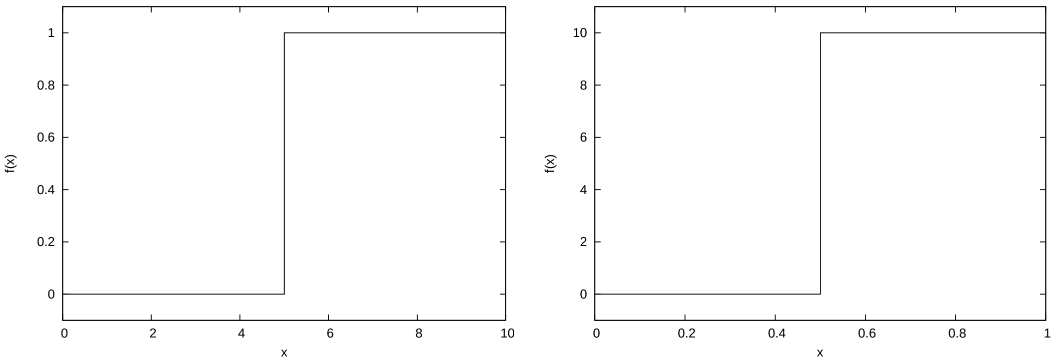
where s is the parameter defining the size of all steps in the Manhattan progressive random walk.

Algorithm 4.5 Manhattan progressive random walk algorithm

```
⋮
The first 14 steps of the algorithm match the progressive random walk algorithm in Algorithm 4.2
⋮
15: for all steps $s$ from 1 to $numSteps$ do
16:     Generate a random dimension, $rD$, in range $[0, \dots, n]$.
17:     for all dimensions $i$ in $n$ do
18:         if $i$ equals $rD$ then
19:             if $startingZone_i$ equals 1 then
20:                 Set $inc = -1$
21:             else
22:                 Set $inc = +1$
23:             end if
24:             Set $walk[s]_{rD} = walk[s-1]_{rD} + (inc * s)$
25:             if $walk[s]_{rD}$ is out of bounds then
26:                 Set $walk[s]_{rD} = walk[s-1]_{rD} - (inc * s)$
27:                 Flip bit $startingZone_{rD}$
28:             end if
29:         else
30:             Set $walk[s]_i = walk[s-1]_i$
31:         end if
32:     end for
33: end for
```

4.6.3 Normalising gradient measures

When calculating the gradient measure between any two points on a walk, it would be desirable for two functions with the same shape within the search space to give the same gradient measures. Consider the functions plotted in Figure 4.15. Although f_1 in Figure 4.15(a) has a larger domain and smaller fitness range compared with Figure 4.15(b), the gradient measures should give the same values for these two functions. To prevent



(a) f_1: Smaller fitness range and larger domain

(b) f_2: Larger fitness range and smaller domain

Figure 4.15: Two functions with identical landscape shapes, but different search space domains and ranges of fitness values.

.

gradient measures from being influenced by fitness ranges and function domains, fitness differences and distances between points are normalised before calculating the gradient measures. The fitness difference is divided by the range of fitness values as encountered on the walk and the s is divided by the total Manhattan distance between the position vectors defining the bounds of the search space. Equation 4.7 is therefore replaced by the following:

$$g(t) = \frac{(f(\mathbf{x}(t+1)) - f(\mathbf{x}(t)))/(f^{max} - f^{min})}{s/(\sum_{i=1}^{n}(x_i^{max} - x_i^{min}))} \tag{4.12}$$

where f^{max} and f^{min} are the maximum and minimum fitness values, respectively, as encountered on the walk and x_i^{max} and x_i^{min} are the bounds of the search space for dimension i.

4.6.4 Testing gradient measures on one-dimensional problems

To investigate the proposed gradient measures and the effect of the step size, the algorithm was run on the one-dimensional functions defined in Table 4.3 and illustrated in Figure 4.8. For each function the following was done:

- For the given function, eight step sizes of exponentially decreasing size were determined based on the range of x values, $xRange = x^{max} - x^{min}$, as follows:

$$\frac{xRange}{2^{11}}, \frac{xRange}{2^{10}}, \frac{xRange}{2^{9}}, \frac{xRange}{2^{8}}, \frac{xRange}{2^{7}}, \frac{xRange}{2^{6}}, \frac{xRange}{2^{5}}, \frac{xRange}{2^{4}}$$

A step size of $\frac{xRange}{2^{11}}$ implies that 2048 steps could fit into a single walk from the minimum x value to the maximum x value, while a step size of $\frac{xRange}{2^{4}}$ implies that only 16 steps would fit into a single walk across the one-dimensional space.

- For each of the eight step sizes, a walk was performed starting at the minimum x value and the measures of G$_{avg}$, G$_{dev}$ and G$_{max}$ were calculated for each walk.

The results are shown in Tables 4.8 to 4.10 and illustrated in Figures 4.16 to 4.18.

Table 4.8: Values of G$_{avg}$ for the one-dimensional benchmark functions illustrated in Figure 4.8 for decreasing step sizes.

Function	G$_{avg}$							
	$\frac{xRange}{2^{11}}$	$\frac{xRange}{2^{10}}$	$\frac{xRange}{2^{9}}$	$\frac{xRange}{2^{8}}$	$\frac{xRange}{2^{7}}$	$\frac{xRange}{2^{6}}$	$\frac{xRange}{2^{5}}$	$\frac{xRange}{2^{4}}$
Table Legs	2	2	2	2	2	2	2	2
Straight	1	1	1	1	1	1	1	1
Half Spherical	1	1	1	1	1	1	1	1
Spherical	2	2	2	2	2	2	2	2
Griewank	8.32	7.98	6.69	3	2.52	2	2	2
Step	2	2	2	2	2	2	2	2
Rastrigin	10.03	10.03	10.03	9.99	9.93	9.78	8.46	4.61
Ackley	13.8	13.77	13.75	13.63	13.57	2	2	2
Schwefel 2.26	5.85	5.85	5.85	5.85	5.84	5.79	5.86	5.81

For some functions, the step size has no effect on the G$_{avg}$ measure. These are the functions that do not have changes in gradient on a small scale, such as Table Legs, Straight, Half Spherical and Spherical. For the rest of the functions, there is a sufficiently small step size below which the G$_{avg}$ measure 'levels out' (as seen on Figure 4.16), indicating that the step size is small enough to capture the fine changes in gradient. The G$_{dev}$ and G$_{max}$ measures (Figures 4.17 and 4.18) show a similar trend of 'levelling

Table 4.9: Values of G$_{dev}$ for the one-dimensional benchmark functions illustrated in Figure 4.8 for decreasing step sizes.

Function	G$_{dev}$							
	$\frac{xRange}{2^{11}}$	$\frac{xRange}{2^{10}}$	$\frac{xRange}{2^{9}}$	$\frac{xRange}{2^{8}}$	$\frac{xRange}{2^{7}}$	$\frac{xRange}{2^{6}}$	$\frac{xRange}{2^{5}}$	$\frac{xRange}{2^{4}}$
Table Legs	63.98	45.23	31.97	22.58	15.94	11.22	7.87	5.47
Straight	0	0	0	0	0	0	0	0
Half Spherical	0.58	0.58	0.58	0.58	0.58	0.58	0.59	0.6
Spherical	1.15	1.16	1.16	1.16	1.16	1.16	1.17	1.18
Griewank	4.32	4.16	3.63	2.05	1.69	1.14	1.14	1.16
Step	16.4	11.51	8.02	5.5	3.63	2.14	1.42	1.27
Rastrigin	4.99	5	4.99	4.99	5.04	4.71	4.19	2.81
Ackley	8.94	8.75	7.88	6.94	3.22	2.99	2.99	2.96
Schwefel 2.26	3.65	3.65	3.65	3.66	3.66	3.65	4.08	3.3

Table 4.10: Values of G$_{max}$ for the one-dimensional benchmark functions illustrated in Figure 4.8 for decreasing step sizes.

Function	G$_{max}$							
	$\frac{xRange}{2^{11}}$	$\frac{xRange}{2^{10}}$	$\frac{xRange}{2^{9}}$	$\frac{xRange}{2^{8}}$	$\frac{xRange}{2^{7}}$	$\frac{xRange}{2^{6}}$	$\frac{xRange}{2^{5}}$	$\frac{xRange}{2^{4}}$
Table Legs	2048	1024	512	256	128	64	32	16
Straight	1	1	1	1	1	1	1	1
Half Spherical	2	2	2	2	1.99	1.98	1.97	1.94
Spherical	4	4	3.99	3.98	3.97	3.94	3.88	3.75
Griewank	16.74	15.98	13.7	7.07	6.66	3.85	3.79	3.68
Step	199.68	99.84	49.92	24.96	12.48	6.24	5.76	4.44
Rastrigin	18.1	18.09	18.09	18.03	17.78	17.56	14.44	8.37
Ackley	37.33	36.91	34.64	30.9	24.4	11.62	10.57	8.83
Schwefel 2.26	13.24	13.24	13.24	13.24	13.22	13.15	13.76	11.75

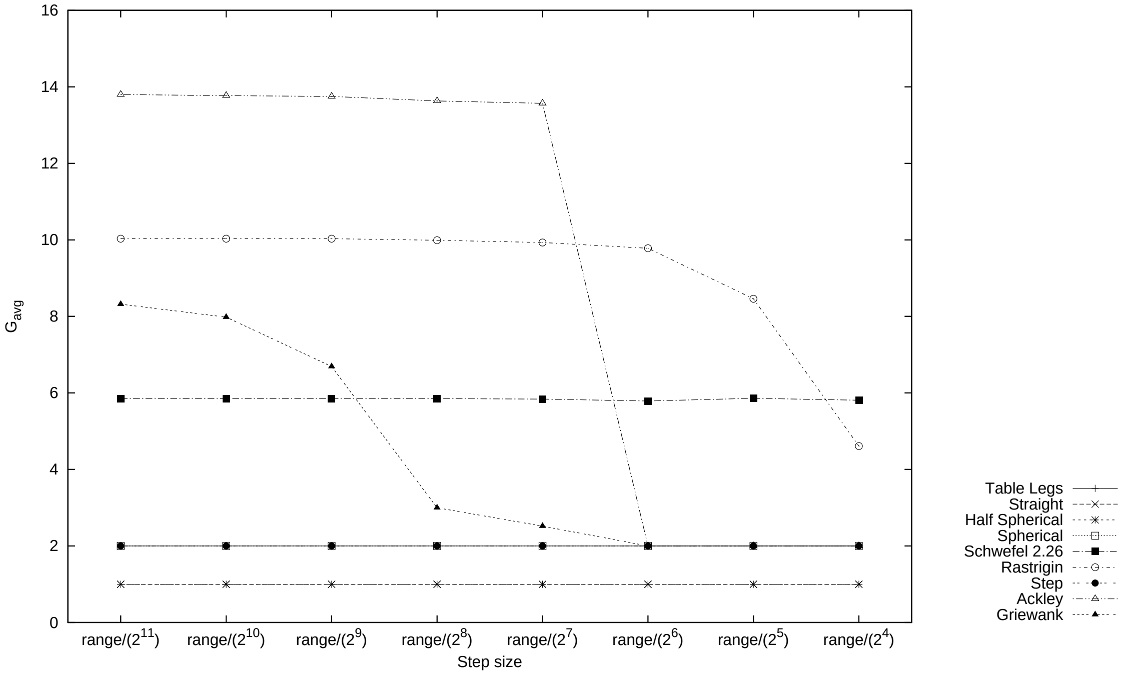


Figure 4.16: The effect of step size on the G$_{avg}$ measure for the one-dimensional functions defined in Table 4.3 and illustrated in Figure 4.8.

out' below a given step size, with the exception of the functions that have vertical slopes (Table Legs and Step). For these functions the maximum gradient continues to increase as the step size decreases, moving closer to the true gradient, which is infinite. Due to these high values, the graphs in Figures 4.17 and 4.18 are cut off at a maximum of 10 and 40, respectively, so that the differences between the values of the remaining functions can be seen.

The Table Legs function was constructed as an extreme case and the gradient measures therefore take on unusual values. The G$_{avg}$ measure is always exactly 2 and the G_{max} measure equals the number of steps taken in the walk. This is because every gradient calculation on the walk will be zero except for the gradients in the two cases where the function is vertical. At these intervals, the gradient will be equal to $\frac{1}{s/xRange}$. For a general step size of $\frac{xRange}{2^n}$, the gradient will therefore be 2^n, which is the number of steps taken on the walk. The average gradient for the Table Legs function will be

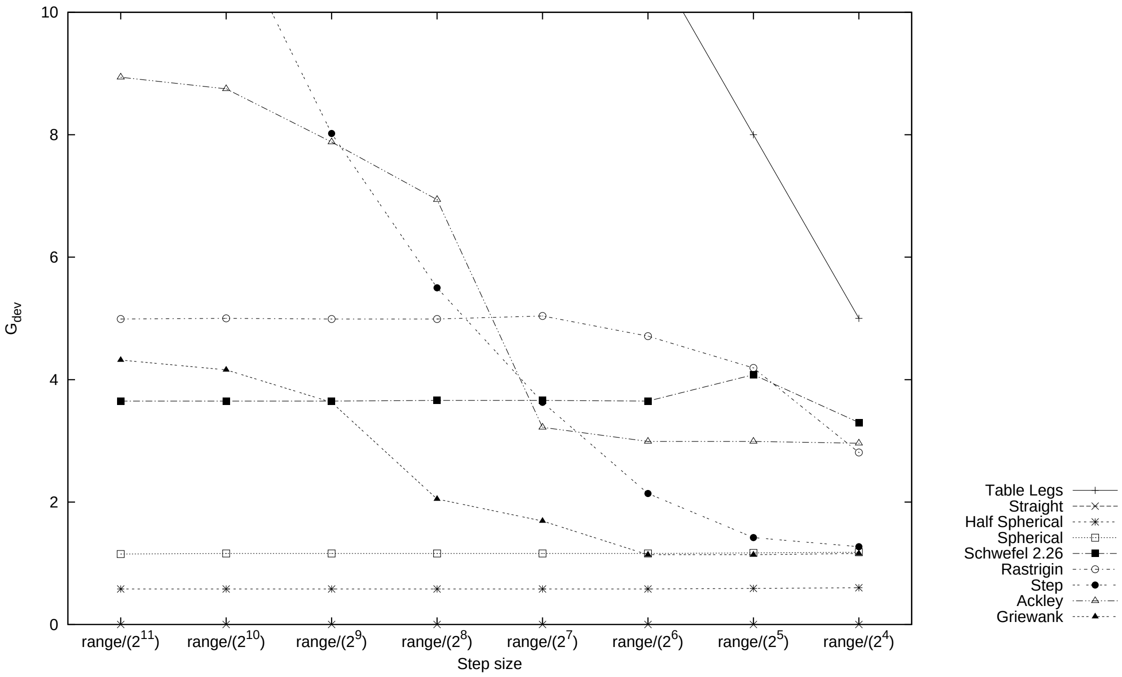


Figure 4.17: The effect of step size on the G$_{dev}$ measure for the one-dimensional functions defined in Table 4.3 and illustrated in Figure 4.8.

$(2^n * 2)/(numberSteps)$, which for a number of steps of 2^n, will result in a G$_{avg}$ measure of 2.

In viewing the effect of step size on the gradient measures, it is clear that the ideal step size for capturing detailed gradient estimations is problem dependant. For the Rastrigin function a step size of $\frac{xRange}{2^7}$ is small enough, but Griewank requires a much smaller size around $\frac{xRange}{2^{11}}$. For the suite of nine functions considered here a step size of $\frac{xRange}{2^{10}}$ is sufficient for distinguishing between the detailed gradient estimations of the problems. The G$_{dev}$ measure shows a very similar profile to the G_{max} measure as can be see in Figures 4.17 and 4.18, with the main difference being that the G_{max} quantities are higher. It would therefore seem that either of these measures would be sufficient for obtaining gradient information on a function, and that using both would not add significant information. Considering only the G$_{avg}$ and G$_{dev}$ measures at step size $\frac{xRange}{2^{11}}$, the following is observed: where functions have vertical gradients with neutral segments

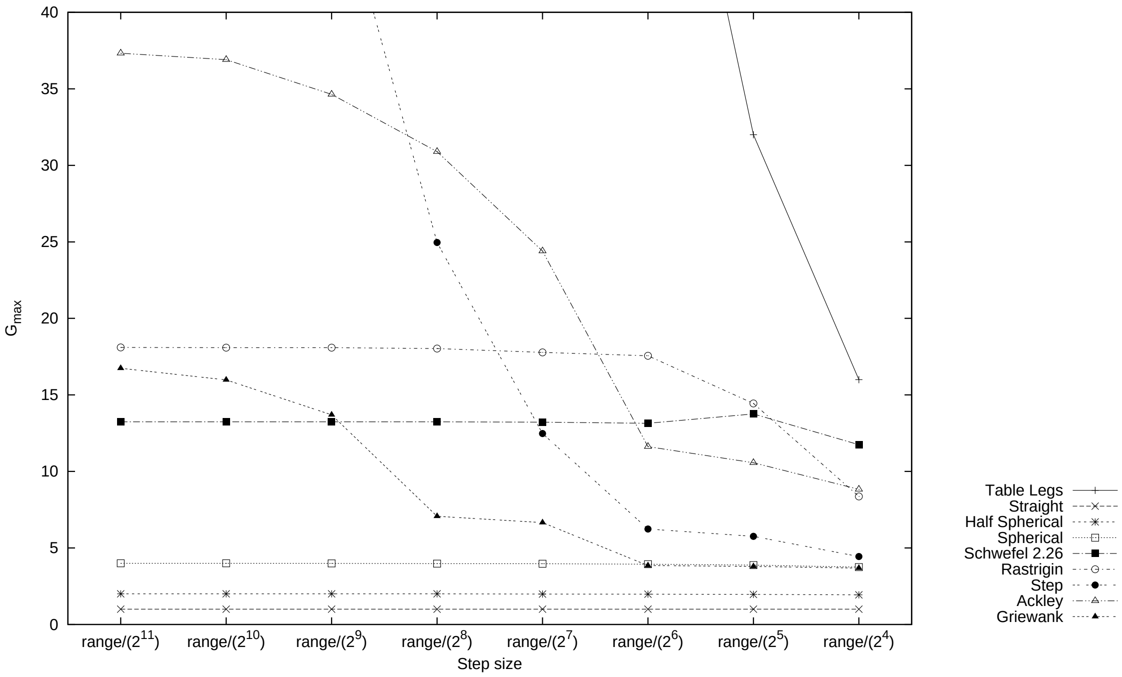


Figure 4.18: The effect of step size on the G_{max} measure for the one-dimensional functions defined in Table 4.3 and illustrated in Figure 4.8.

between them (Table Legs and Step), the G$_{dev}$ measure is greater than the G$_{avg}$ measure. For all other functions, the value of G$_{dev}$ is less than G$_{avg}$. The relationship between G$_{avg}$ and G$_{dev}$ therefore seems to be a significant factor in understanding gradients of functions.

The final proposed gradient measures are summarised in Table 4.11. The next section tests all of the proposed measures from this chapter on higher dimensional problems and investigates the link to PSO performance.

4.7 Linking to PSO performance on higher dimensional problems

The measures proposed in this chapter give results for one-dimensional functions that are consistent with a visual inspection of the plotted functions. This section investigates

Table 4.11: Proposed gradient estimation measures based on Manhattan random walks

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
G_{avg}	Average estimated gradient.	(1) Number of steps in the Manhattan progressive random walk, ns, (2) step size, s (a proportion of the range of the domain).	A positive real number, where a higher value indicates higher average gradients.	$ns + 1$ function evaluations and ns gradient estimations (Equation 4.12).
G_{dev}	Standard deviation from the average estimated gradient.	(1) Number of steps in the Manhattan progressive random walk, ns, (2) step size, s (a proportion of the range of the domain).	A positive real number, where a higher value indicates higher deviations from average gradients.	$ns + 1$ function evaluations and ns gradient estimations (Equation 4.12).

the proposed techniques on higher dimensional functions and also investigates the link to PSO performance in terms of a traditional gbest PSO algorithm.

4.7.1 Experimental setup

This section describes the experimental setup in terms of the benchmark functions used, the parameters for the performance landscape metrics and for the PSO algorithm.

Benchmark functions

A selection of the benchmark functions defined in Appendix A were used to test the fitness landscape measures proposed in this chapter. Ackley, Griewank, Quadric, Rana, Rastrigin, Rosenbrock, Salomon, Schwefel 2.26, Spherical and Step were used for dimension (D) 1 (not applicable for Rana and Rosenbrock), 2, 5, 15 and 30. The selected functions exhibit different landscape profiles. Quadric and Spherical are both smooth, unimodal functions with low gradients. Rosenbrock is a smooth, relatively flat function, widely believed to be unimodal, but has been shown to have a local (non-global) minimum for $4 \leq D \leq 30$ [140]. Ackley, Griewank, Rana, Rastrigin, Salomon, Schwefel 2.26 and Step are all rugged, although Griewank and Step are rugged on more of a micro scale than the others. The only multi-funnel landscapes are Schwefel 2.26 and Rana. Griewank, Rana, Rastrigin and Schwefel 2.26 have medium gradients, whereas Ackley and Salomon have very steep gradients. Step has vertical gradients, but these are interspersed by sections with zero gradients, so the average gradient is low.

Fitness landscape measures setup

The $\text{FEM}_{0.01}$ and $\text{FEM}_{0.1}$ were used to estimate micro and macro ruggedness of the benchmark problems and were based on D progressive random walks of 1000 steps. The DM measure was used to estimate the presence of funnels in the problem landscapes. DM values were based on uniform random samples of 1000 points and involved subtracting the dispersion of all solutions in the sample from the dispersion of the subset of 10% fittest solutions. Gradients were estimated based on D Manhattan progressive random walks of 1000 steps each with step size $\frac{(x^{max}-x^{min})*D}{1000}$. The G_{avg} and G_{dev} measures were calculated across all walks. Thirty independent runs of each fitness landscape measure algorithm were performed on each function/dimension combination. Mean values over the 30 runs were calculated and are reported in Table 4.12.

PSO performance

Each of the problem instances (function and dimension combinations) was solved using the traditional gbest PSO algorithm (described in Section 3.2). The values for QMetric, SSpeed and SRate (described in Section 3.3) were determined based on 30 independent runs of each algorithm on each problem instance. The parameter values used for the PSO algorithm were as follows: 50 particles, 0.72 inertia weight (w) and 1.496 for the cognitive and social acceleration constants (c_1 and c_2). Results are shown in the last three columns of Table 4.12.

4.7.2 Results

This section discusses the results presented in Table 4.12 and assesses the value of the fitness landscape metrics as predictors of PSO performance.

Fitness landscape metrics

The FEM ruggedness estimation values are in the range $[0, 1]$, where 1 indicates maximal ruggedness. Considering the values of $\text{FEM}_{0.01}$ in Table 4.12, Ackley and Salomon have the highest values, whereas Rana, Rastrigin and Schwefel 2.26 have the highest values for $\text{FEM}_{0.1}$, which is as expected. The dispersion metric (DM) is a measure of the presence of funnels. Negative values for DM indicate a simpler global topology, while larger values (positive values) are indicative of multi-funnels. All DM values in Table 4.12 are negative except for Rana and Schwefel 2.26 (above one dimension), which are the only two multi-funnelled benchmark functions. Considering the average gradient estimation values (G_{avg}), the following is observed:

- Spherical and Step functions have average gradients of approximately 2 in all dimensions, indicating that the basic shape of the function is the same even when the search space becomes much larger.
- Salomon has the highest values for G_{avg}, while Quadric and Rosenbrock have the lowest values indicating relatively flat landscapes.
- Griewank has fairly high average gradients in 1 dimension, but this decreases as the dimension increases. This indicates a simpler landscape in higher dimensions (explained in [81]).

Table 4.12: Landscape measures alongside PSO performance metrics for selected benchmark functions f defined in Appendix A, Table A.1 in different dimensions (D). All landscape measures are means over 30 independent runs and standard deviations are shown in parentheses.

f	D	Landscape measures					Performance metrics		
		FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	QMetric	SRate	SSpeed
f_{ack}	1	0.874 (±0.008)	0.769 (±0.012)	-0.296 (±0.003)	13.703 (±0.000)	8.870 (±0.000)	1.000	1.000	0.471
f_{ack}	2	0.858 (±0.008)	0.780 (±0.019)	-0.357 (±0.008)	33.562 (±15.963)	22.058 (±10.451)	1.000	1.000	0.634
f_{ack}	5	0.867 (±0.004)	0.832 (±0.017)	-0.333 (±0.017)	35.314 (±9.921)	20.681 (±5.808)	1.000	1.000	0.782
f_{ack}	15	0.870 (±0.003)	0.849 (±0.005)	-0.288 (±0.018)	3.505 (±0.573)	2.969 (±0.388)	0.933	0.933	0.855
f_{ack}	30	0.870 (±0.002)	0.846 (±0.007)	-0.270 (±0.018)	3.818 (±0.361)	3.094 (±0.267)	0.300	0.300	0.862
f_{grw}	1	0.788 (±0.012)	0.550 (±0.013)	-0.289 (±0.003)	7.962 (±0.000)	4.152 (±0.000)	1.000	1.000	0.721
f_{grw}	2	0.652 (±0.046)	0.606 (±0.019)	-0.358 (±0.011)	4.499 (±0.302)	3.209 (±0.215)	0.915	0.767	0.669
f_{grw}	5	0.478 (±0.032)	0.644 (±0.011)	-0.358 (±0.012)	2.148 (±0.056)	1.312 (±0.035)	0.560	0.067	0.211
f_{grw}	15	0.343 (±0.009)	0.644 (±0.007)	-0.338 (±0.013)	2.048 (±0.020)	1.206 (±0.012)	0.691	0.100	0.910
f_{grw}	30	0.286 (±0.006)	0.641 (±0.005)	-0.328 (±0.015)	2.009 (±0.021)	1.201 (±0.012)	0.902	0.367	0.921
f_{qad}	1	0.467 (±0.011)	0.543 (±0.010)	-0.297 (±0.003)	2.000 (±0.000)	1.155 (±0.000)	1.000	1.000	0.942
f_{qad}	2	0.519 (±0.065)	0.597 (±0.031)	-0.329 (±0.010)	1.373 (±0.228)	0.963 (±0.128)	1.000	1.000	0.893
f_{qad}	5	0.452 (±0.044)	0.567 (±0.021)	-0.213 (±0.019)	0.906 (±0.093)	0.863 (±0.051)	1.000	1.000	0.906
f_{qad}	15	0.355 (±0.020)	0.482 (±0.027)	-0.087 (±0.021)	0.726 (±0.028)	0.739 (±0.028)	1.000	1.000	0.869
f_{qad}	30	0.297 (±0.015)	0.379 (±0.025)	-0.058 (±0.021)	0.597 (±0.020)	0.609 (±0.013)	1.000	1.000	0.711
f_{ran}	2	0.470 (±0.077)	0.816 (±0.014)	0.051 (±0.014)	12.801 (±3.403)	19.217 (±5.663)	0.187	0.000	0.000

Continued on Next Page…

Table 4.12 – Continued

f	D	Landscape measures					Performance metrics		
		FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	QMetric	SRate	SSpeed
f_{ran}	5	0.692 (±0.014)	0.867 (±0.004)	0.069 (±0.017)	18.150 (±1.186)	19.740 (±1.684)	0.000	0.000	0.000
f_{ran}	15	0.730 (±0.005)	0.871 (±0.003)	0.044 (±0.021)	17.465 (±0.451)	16.779 (±0.430)	0.000	0.000	0.000
f_{ran}	30	0.739 (±0.003)	0.870 (±0.004)	0.032 (±0.021)	14.198 (±0.240)	12.613 (±0.238)	0.000	0.000	0.000
f_{ras}	1	0.542 (±0.009)	0.881 (±0.007)	-0.212 (±0.010)	10.030 (±0.000)	4.996 (±0.000)	1.000	1.000	0.814
f_{ras}	2	0.591 (±0.020)	0.871 (±0.005)	-0.224 (±0.015)	11.871 (±1.887)	5.925 (±0.943)	1.000	1.000	0.792
f_{ras}	5	0.602 (±0.014)	0.865 (±0.005)	-0.239 (±0.016)	14.190 (±0.951)	7.078 (±0.474)	0.533	0.533	0.731
f_{ras}	15	0.596 (±0.008)	0.864 (±0.003)	-0.232 (±0.014)	16.765 (±0.848)	8.395 (±0.423)	0.000	0.000	0.000
f_{ras}	30	0.591 (±0.005)	0.865 (±0.002)	-0.227 (±0.016)	16.894 (±0.364)	8.387 (±0.179)	0.000	0.000	0.000
f_{ros}	2	0.372 (±0.073)	0.507 (±0.027)	-0.220 (±0.015)	1.340 (±0.104)	1.700 (±0.203)	1.000	1.000	0.843
f_{ros}	5	0.467 (±0.022)	0.640 (±0.009)	-0.311 (±0.014)	1.157 (±0.058)	1.378 (±0.066)	0.931	0.067	0.321
f_{ros}	15	0.411 (±0.012)	0.687 (±0.005)	-0.280 (±0.014)	1.061 (±0.017)	1.199 (±0.019)	0.888	0.000	0.000
f_{ros}	30	0.350 (±0.008)	0.694 (±0.005)	-0.273 (±0.014)	1.030 (±0.011)	1.134 (±0.010)	0.482	0.000	0.000
f_{sal}	1	0.891 (±0.007)	0.802 (±0.011)	-0.277 (±0.005)	61.860 (±0.000)	34.228 (±0.000)	1.000	1.000	0.574
f_{sal}	2	0.888 (±0.005)	0.802 (±0.012)	-0.346 (±0.007)	64.260 (±6.982)	41.572 (±7.486)	1.000	1.000	0.592
f_{sal}	5	0.890 (±0.003)	0.807 (±0.007)	-0.350 (±0.016)	51.397 (±2.504)	32.287 (±2.013)	0.000	0.000	0.000
f_{sal}	15	0.889 (±0.002)	0.805 (±0.005)	-0.324 (±0.011)	24.737 (±0.832)	18.815 (±0.616)	0.000	0.000	0.000
f_{sal}	30	0.885 (±0.001)	0.802 (±0.004)	-0.318 (±0.013)	20.055 (±0.565)	14.220 (±0.390)	0.000	0.000	0.000
$f_{sch2.26}$	1	0.485 (±0.010)	0.822 (±0.008)	-0.003 (±0.030)	5.853 (±0.000)	3.649 (±0.000)	1.000	1.000	0.816
$f_{sch2.26}$	2	0.537 (±0.022)	0.846 (±0.007)	0.035 (±0.018)	7.640 (±0.848)	4.765 (±0.524)	0.967	0.967	0.819
$f_{sch2.26}$	5	0.564 (±0.019)	0.852 (±0.005)	0.038 (±0.017)	9.731 (±1.122)	6.079 (±0.701)	0.400	0.400	0.826

Continued on Next Page…

Table 4.12 – Continued

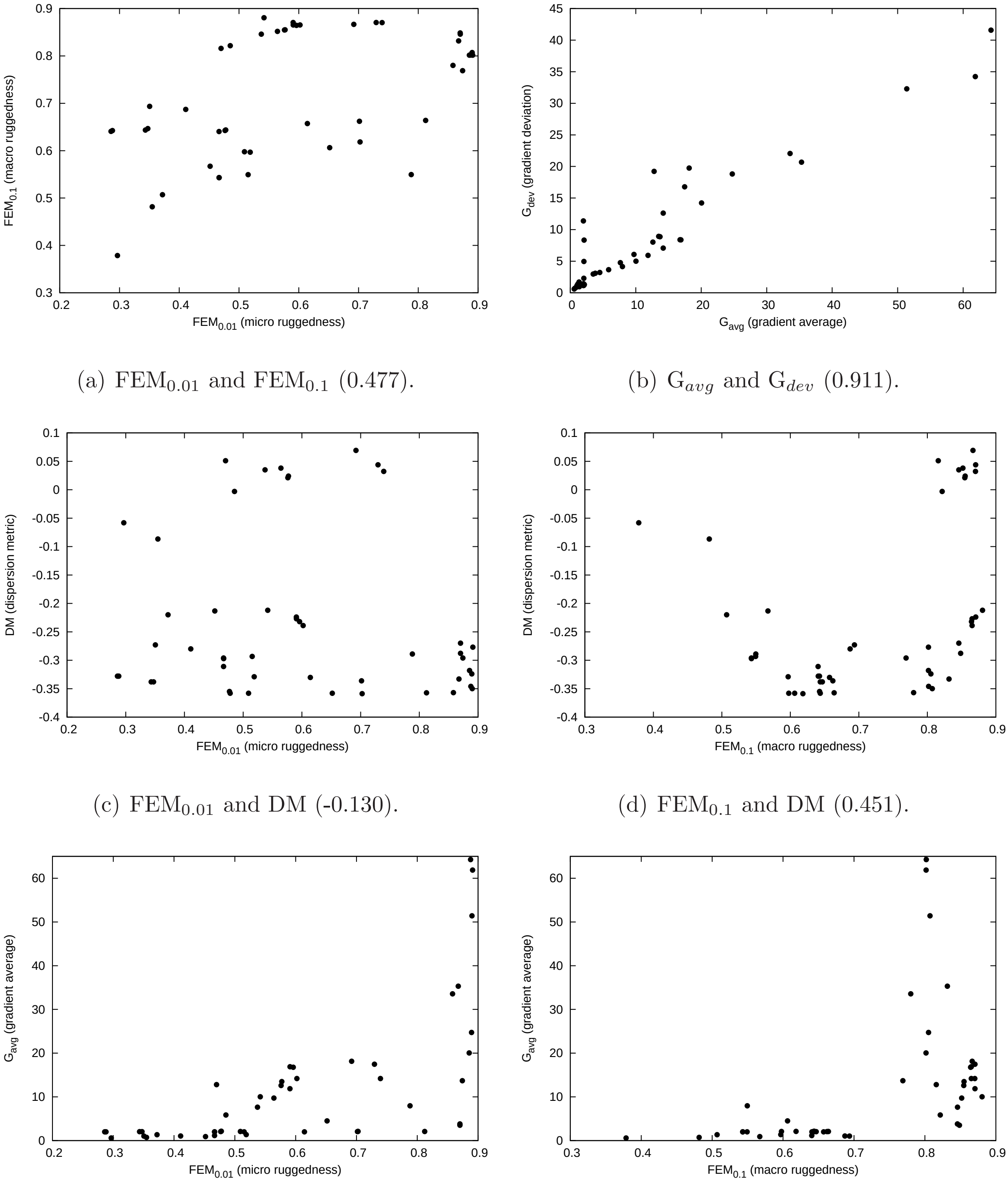
f	D	Landscape measures					Performance metrics		
		FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	QMetric	SRate	SSpeed
$f_{sch2.26}$	15	0.576 (±0.010)	0.855 (±0.002)	0.021 (±0.017)	12.611 (±1.048)	8.028 (±0.664)	0.000	0.000	0.000
$f_{sch2.26}$	30	0.577 (±0.007)	0.855 (±0.002)	0.024 (±0.022)	13.502 (±0.828)	8.927 (±0.553)	0.000	0.000	0.000
f_{sph}	1	0.467 (±0.012)	0.543 (±0.012)	-0.296 (±0.002)	2.000 (±0.000)	1.155 (±0.000)	1.000	1.000	0.845
f_{sph}	2	0.509 (±0.068)	0.598 (±0.022)	-0.358 (±0.010)	2.078 (±0.113)	1.199 (±0.065)	1.000	1.000	0.890
f_{sph}	5	0.477 (±0.036)	0.643 (±0.013)	-0.355 (±0.011)	2.059 (±0.048)	1.194 (±0.029)	1.000	1.000	0.916
f_{sph}	15	0.347 (±0.011)	0.647 (±0.008)	-0.338 (±0.010)	2.045 (±0.025)	1.204 (±0.015)	1.000	1.000	0.941
f_{sph}	30	0.288 (±0.006)	0.642 (±0.006)	-0.328 (±0.014)	2.004 (±0.016)	1.197 (±0.009)	1.000	1.000	0.940
f_{stp}	1	0.515 (±0.006)	0.549 (±0.012)	-0.293 (±0.003)	2.000 (±0.000)	11.375 (±0.000)	1.000	1.000	0.994
f_{stp}	2	0.703 (±0.025)	0.618 (±0.025)	-0.359 (±0.010)	2.110 (±0.140)	8.345 (±0.556)	1.000	1.000	0.988
f_{stp}	5	0.812 (±0.017)	0.664 (±0.012)	-0.357 (±0.015)	2.085 (±0.043)	4.955 (±0.102)	1.000	1.000	0.977
f_{stp}	15	0.702 (±0.009)	0.662 (±0.007)	-0.336 (±0.011)	2.073 (±0.018)	2.299 (±0.018)	1.000	1.000	0.972
f_{stp}	30	0.614 (±0.007)	0.657 (±0.006)	-0.330 (±0.012)	2.012 (±0.023)	1.441 (±0.017)	0.924	0.900	0.857

- Rastrigin and Schwefel 2.26 show an increase in G_{avg} in higher dimensions, but the average gradient measure for Ackley decreases in 15 and 30 dimensions. A possible explanation is that the fitness range of Ackley stays the same regardless of the dimension. In 30 dimensions, the fitness range is therefore the same as in 1 dimension and yet the search space has increased enormously. This results in a relative "flattening" of the function compared to the other functions where the fitness range grows with the increase in search space.

For most problems, there is a strong correlation between the G_{avg} and G_{dev} values. This is illustrated in Figure 4.19(b). One exception is in the case of the Step function in lower dimensions, where the vertical jumps in fitness result in high deviations, but low averages. Another exception is for the Rana function, which has a higher deviation value than average in dimensions 2 and 5, which is indicative of very steep gradients in places.

Figure 4.19 shows scatter diagrams of some of the landscape metrics visually illustrating the correlation (or lack of correlation) between different measures. There is only a moderate correlation between $\mathrm{FEM}_{0.01}$ and $\mathrm{FEM}_{0.1}$ (Figure 4.19(a)), with some problems having high micro ruggedness and medium macro ruggedness and some having high macro ruggedness and medium micro ruggedness. This diagram confirms that there is value in having two measures of ruggedness as the measures capture different information on problems. There is a much stronger correlation between the G_{avg} and G_{dev} measures (Figure 4.19(b)), with only a few problems having relatively higher G_{dev} values than G_{avg} values. More data is needed to confirm whether both measures are necessary to predict PSO performance.

Figures 4.19(c) and 4.19(d) show the relationship between the two ruggedness measures and dispersion metric. There is a moderate correlation between macro ruggedness and dispersion metrics, which is a measure of global structure. It is interesting to see that there seem to be three distinct clusters: in the top right is the cluster of problems with high macro ruggedness and positive DM values (indicating multi-funnels). In the bottom right are problems with high macro ruggedness and low DM values (simple global structure), and in the bottom middle is a group of problems with medium macro ruggedness and simple global structure.



(a) FEM$_{0.01}$ and FEM$_{0.1}$ (0.477).

(b) G$_{avg}$ and G$_{dev}$ (0.911).

(c) FEM$_{0.01}$ and DM (-0.130).

(d) FEM$_{0.1}$ and DM (0.451).

(e) FEM$_{0.01}$ and G$_{avg}$ (0.788).

(f) FEM$_{0.1}$ and G$_{avg}$ (0.727).

Figure 4.19: Scatter diagrams showing the correlation between some of the fitness landscape measures. Spearman's correlation coefficient values are given in parentheses in the sub-captions.

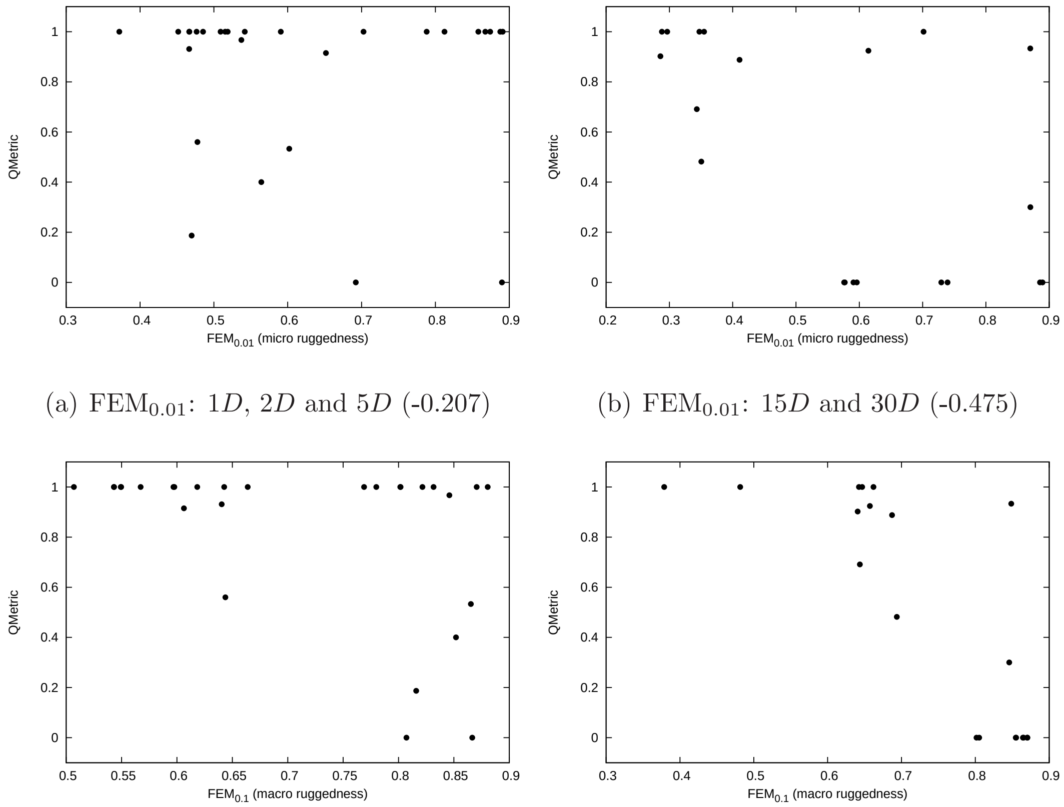
Figures 4.19(e) and 4.19(f) show the ruggedness measures against gradient average. In both cases there is a strong correlation. In general, problems with low ruggedness have low gradient averages. What the scatter diagrams show (for macro ruggedness in particular) is that the gradient average provides a way of distinguishing between problems that have a high ruggedness. The gradient measures therefore provide different information that could be useful in understanding PSO behaviour.

4.7.3 The link to PSO performance

This subsection investigates the link between the fitness landscape measures and PSO performance. Figure 4.20 shows scatter plots of the ruggedness measures against QMetric for the dataset in Table 4.12. Lower dimensional problems (1, 2 and 5 D) are grouped into a single plot and higher dimensional problems (15 and 30 D) are grouped into a second plot. The correlation with QMetric ranges from weak for FEM$_{0.01}$ in lower dimensions to moderate for FEM$_{0.01}$ in higher dimensions and FEM$_{0.1}$ in lower dimensions, to strong for FEM$_{0.1}$ in higher dimensions. The correlations are not straight-forward, but there are clusters of points in places. For example, in Figure 4.20(b), the instances with low micro ruggedness (below 0.5) all have reasonable QMetric values (above 0.4).

In Figure 4.20(d), all instances with QMetric values of 0 (indicating algorithm failure) have high macro ruggedness values (above 0.8). QMetric on its own only captures part of the picture of performance. An alternative approach is to allocate each instance into a performance class using a combination of QMetric, SRate and SSpeed values (as described in Section 3.3.4). Figure 4.21 plots these classes against FEM measures with each instance grouped according to dimension. The legend gives the class symbols in the order in which they should appear (from worst to best performance) in each dimension column if the measure is negatively correlated with performance. Although there are exceptions, the triangles (always solved and fast) mostly appear lower than the circles (not solved) in Figure 4.21, particularly in Figure 4.21(b).

Figure 4.22 shows the scatterplots of DM, G$_{avg}$ and G$_{dev}$ against QMetric. The plots of DM show distinct clusters. For example, in the top left corner of Figure 4.22(a) there is a group of problems that have low DM values (indicating simple global structure) and have high QMetric values (mostly with value 1). In the higher dimensional plot, there



(a) FEM$_{0.01}$: $1D$, $2D$ and $5D$ (-0.207)

(b) FEM$_{0.01}$: $15D$ and $30D$ (-0.475)

(c) FEM$_{0.1}$: $1D$, $2D$ and $5D$ (-0.531)

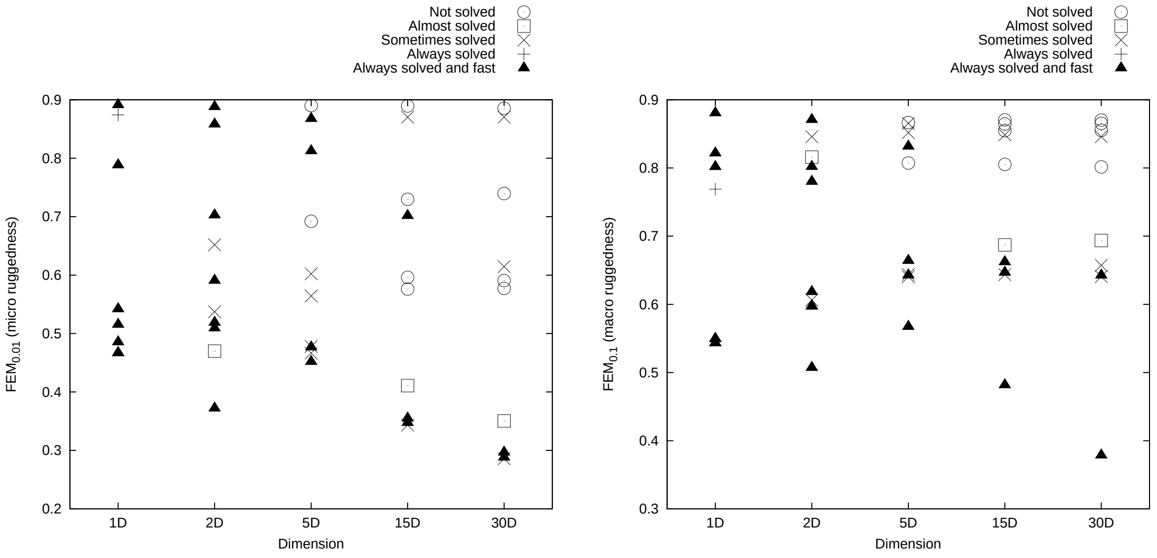
(d) FEM$_{0.1}$: $15D$ and $30D$ (-0.836)

Figure 4.20: PSO performance (measured as QMetric) plotted against FEM micro and macro ruggedness measures for functions with low dimension and higher dimensions (D).

is a small cluster of problems that have positive DM values (indicative of multi-funnels) and have a QMetric value of 0 (indicating algorithm failure). Both G$_{avg}$ and G$_{dev}$ have strong correlations with QMetric in higher dimensions. It is very clear from Figures 4.22(d) and 4.22(f) that the PSO failed to solve higher dimensional problems with high G$_{avg}$ or G$_{dev}$ values.

Figures 4.23 and 4.24 show the discretised performance classes against DM and gradient measures. The circles in general appear higher in the diagram than the triangles, particularly in the higher dimensions with the gradient measures.

These results show that there is some value in each of the fitness landscapes measures as part-predictors of PSO performance. Although all provide some information,



(a) $\text{FEM}_{0.01}$ (micro ruggedness).

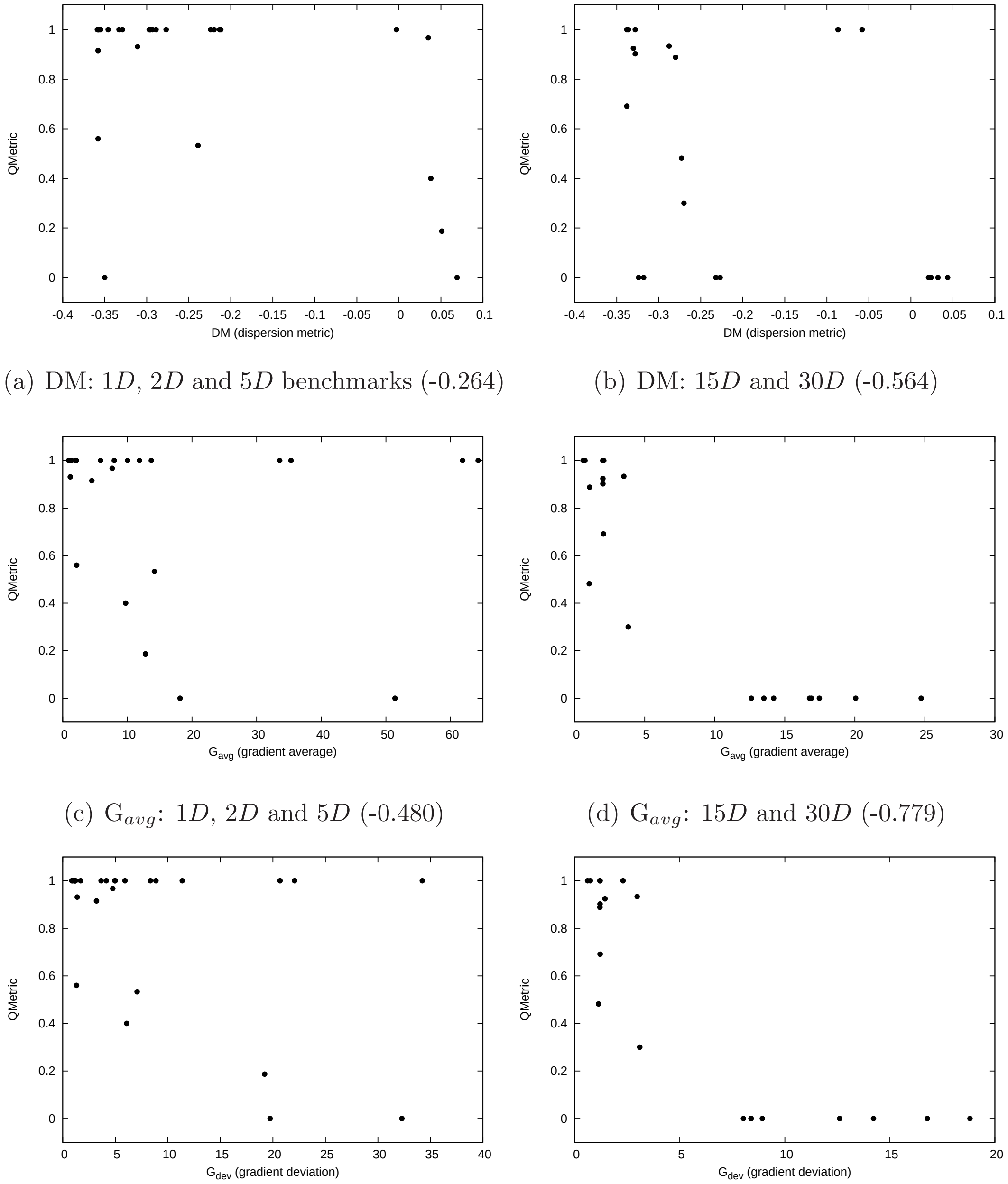
(b) $\text{FEM}_{0.1}$ (macro ruggedness).

Figure 4.21: Performance of a traditional PSO algorithm on benchmark problems plotted against ruggedness measures.

none provide sufficient information to be used as a single measure for predicting PSO performance.

4.8 Summary

This chapter proposed a number of fitness landscape metrics for continuous spaces: two ruggedness measures based on entropy, a dispersion measure for predicting the presence of funnels, and two fitness gradient estimation measures. These metrics were considered alongside performance measures of a traditional PSO algorithm on a range of benchmark functions. Results show that all measures show some correlation to PSO performance and can therefore be used as part-predictors of PSO performance. To properly predict PSO performance, more landscape features should be considered alongside these features of ruggedness, funnels and gradients. The following chapter investigates measures related to the searchability (evolvability) of a fitness landscape.



(a) DM: $1D$, $2D$ and $5D$ benchmarks (-0.264)

(b) DM: $15D$ and $30D$ (-0.564)

(c) G_{avg}: $1D$, $2D$ and $5D$ (-0.480)

(d) G_{avg}: $15D$ and $30D$ (-0.779)

(e) G_{dev}: $1D$, $2D$ and $5D$ (-0.395)

(f) G_{dev}: $15D$ and $30D$ (-0.790)

Figure 4.22: PSO performance (measured as QMetric) plotted against dispersion metric and gradient measures for functions with low dimension and higher dimensions (D).

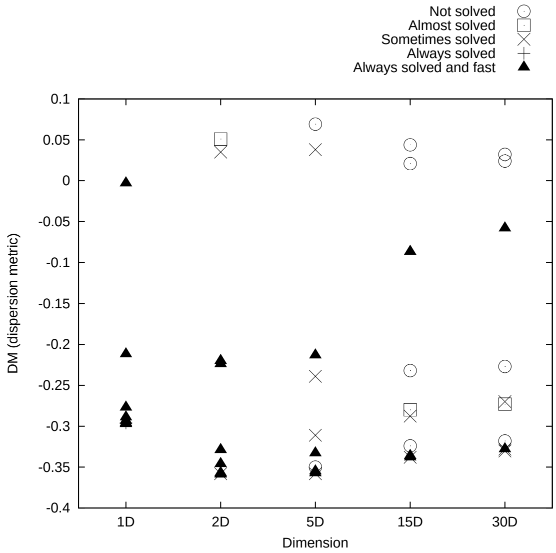
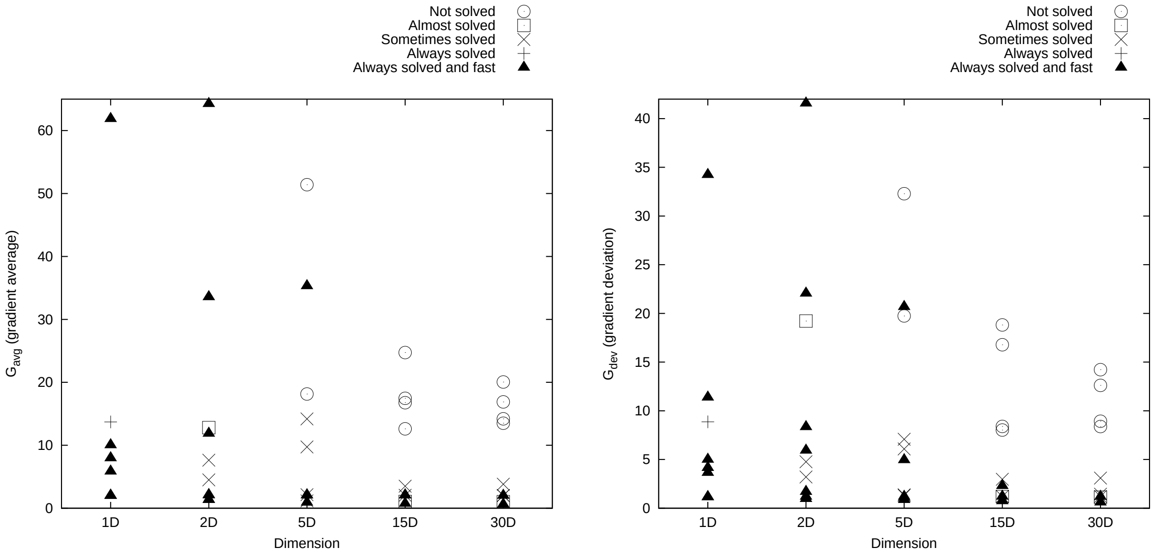


Figure 4.23: Performance of a traditional PSO algorithm on benchmark problems plotted against dispersion metric.



(a) G_{avg} (gradient average).

(b) G_{dev} (gradient deviation).

Figure 4.24: Performance of a traditional PSO algorithm on benchmark problems plotted against gradient measures.

Chapter 5

Searchability

Research from this chapter has been submitted as an article to Swarm Intelligence journal [86].

The notion of the evolvability of a fitness landscape with respect to a particular search algorithm was discussed in Section 2.3.11. The term *searchability* was introduced to broaden the scope of evolvability to encompass non-evolutionary based search techniques and is defined as the ability of a given search process to move to a place in the landscape of better fitness. This chapter further investigates techniques for analysing the searchability of fitness landscapes. Section 5.1 describes some existing techniques for quantifying or visualising evolvability. Two general techniques for measuring searchability are proposed in Section 5.2. These techniques are general in that they do not assume a particular search algorithm. Section 5.3 describes a technique for visualising searchability, called fitness clouds, adapted to be based on particle swarm updates. A number of measures based on fitness clouds are then proposed. Experimentation on the proposed techniques on one-dimensional benchmarks is performed in Section 5.4 and Section 5.5 investigates the link between searchability measures and PSO performance on higher dimensional benchmark problems.

5.1 Existing techniques for measuring evolvability

There are fitness landscape measures that were originally conceived as a way of quantifying problem difficulty. Two such examples are Jones and Forrest's fitness distance correlation [68] and Borenstein and Poli's information landscape hardness measure [15]. Both these techniques require knowledge of the global optima, and so cannot be used as predictive measures of algorithm performance on unknown problems when used in their original form. An alternative is to base the measure on a sample of the search space and to use the fittest point from the sample in the place of the global optimum. This implies a shift in focus away from measuring hardness to measuring searchability, since the aim is no longer to quantify how well or badly the problem guides search towards the optimum, but rather to quantify how well or badly the problem guides search towards a place of better fitness.

There are also techniques that were specifically designed to visualise or quantify evolvability. Three such techniques include fitness evolvability portraits [142], fitness clouds [173] and fitness-probability clouds [83]. These all produce visual plots as output (average evolvability metrics against fitness in the case of fitness evolvability portraits, a scatterplot showing the relationship between the fitness values of parents and offspring in the case of fitness clouds and a plot of fitness values against escape probability [94] in the case of fitness-probability clouds). Although visual plots are potentially useful for human analysis, numerical output is more useful for facilitating automated analysis of problem features for performance prediction. The negative slope coefficient [162, 164] is a numerical output measure that quantifies the evolvability of a fitness landscape and is based on Verel *et al.*'s fitness cloud [173]. Similarly, accumulated escape probability [83] is a numerical output measure that quantifies evolvability based on a fitness-probability cloud, but is restricted to problems with a discrete representation.

This section describes fitness distance correlation [68], information landscape hardness [15], fitness clouds [173] and negative slope coefficient [162, 164]. Sections 5.2 and 5.3 propose ways of adapting these techniques to be used to measure searchability of continuous problems in the context of particle swarm optimizers.

5.1.1 Fitness distance correlation

Fitness distance correlation (FDC) was introduced by Jones and Forrest [68] as a way of predicting the performance of a GA on problems with known global optima and measures the correlation between the fitness of solutions and the distance to the nearest optimum. The basic premise is that for a landscape to be easy to search, fitness values should decrease (increase) as distance to the optimum decreases in the case of minimisation (maximisation) problems. This phenomenon in a landscape would provide local search algorithms with information to guide search in the right direction.

Given a set of n points with associated fitness values $F = \{f_1, \ldots, f_n\}$ and distances of each point to the nearest optimum in search space $Dist = \{d_1, \ldots, d_n\}$, the FDC is calculated as the covariance of F and $Dist$ divided by the product of the standard deviation of F and standard deviation of $Dist$, or:

$$FDC = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i - \overline{f})(d_i - \overline{d})}{\sigma(F)\sigma(Dist)} \tag{5.1}$$

where $\overline{f}$, $\overline{d}$, $\sigma(F)$ and $\sigma(Dist)$ are the means of F and $Dist$ and the standard deviations of F and $Dist$, respectively.

The FDC measure takes on values from –1 (perfect anti-correlation) to +1 (perfect correlation), where low values are regarded as desirable for maximisation problems and high values desirable for minimisation problems.

In the original study by Jones and Forrest [68], Hamming distance was used as the measure of distance, but a number of subsequent studies have proposed alternatives, such as the use of crossover to determine distance [3] and distance metrics between trees for genetic programming problems [162]. Jones and Forrest [68] showed empirically that the FDC measure is a reliable indicator of GA performance on a wide range of problems.

A significant limitation of the FDC technique is that the optimal solution(s) must be known beforehand. It is also computationally intensive to compute ($O(n^2)$ [65]) and does not reliably predict the difficulty of optimising the problem [3, 65, 106, 107, 121, 129].

Section 5.2.1 proposes a modified FDC measure to quantify searchability with respect to local search, rather than problem hardness.

5.1.2 Information landscape measures

Borenstein and Poli [14, 15] introduced the concept of an *information landscape*: a matrix of all possible comparisons between solutions based on fitness values. Given a fitness function f of a maximisation problem and a set X of discrete solutions, an information matrix M is defined as having $|X| \times |X|$ entries $m_{i,j} = t(x_i, x_j)$, where

$$t(x_i, x_j) = \begin{cases} 1 & \text{if } f(x_i) > f(x_j) \\ 0.5 & \text{if } f(x_i) = f(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

For a minimisation problem, the definition changes to

$$t(x_i, x_j) = \begin{cases} 1 & \text{if } f(x_i) < f(x_j) \\ 0.5 & \text{if } f(x_i) = f(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

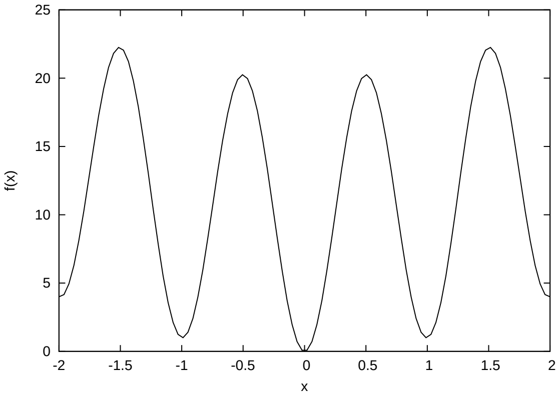
Figure 5.1 shows an example of the one-dimensional Rastrigin minimisation problem: the fitness landscape is plotted in (a), with a sample of points and fitness values in (b), and the associated information landscape in (c). Only a subset of the elements in the information matrix (the bolded cells in Figure 5.1(c)) are necessary to define the information landscape for the following reasons:

- The diagonal entries, being comparisons between the same solutions, are all 0.5, so can be ignored.
- The matrix has symmetries with respect to the diagonal (since $t(x_i, x_j) = 1 - t(x_j, x_i)$), so half of the matrix is unnecessary.
- The row and column of the optimal solution can be ignored, since it is known that this point is better than all others.

In this way, the information matrix can be reduced to a vector to store only the relevant entries in the matrix:

$$V = (v_1, v_2, \dots, v_n)$$

where the elements of V represent all the elements of the information matrix, excluding the unnecessary elements described above and $|V| = n = (|X| - 1)(|X| - 2)/2$.



(a) Fitness Landscape

Point	Fitness
-2	4
-1.5	22.25
-1	1
-0.5	20.25
0	0
0.5	20.25
1	1
1.5	22.25
2	4

(b) Sample of points with fitness values

	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
-2	0.5	**1**	**0**	**1**	0	**1**	**0**	**1**	**0.5**
-1.5	0	0.5	**0**	**0**	0	**0**	**0**	**0.5**	**0**
-1	1	1	0.5	**1**	0	**1**	**0.5**	**1**	**1**
-0.5	0	1	0	0.5	0	**0.5**	**0**	**1**	**0**
0	1	1	1	1	0.5	1	1	1	1
0.5	0	1	0	0.5	0	0.5	**0**	**1**	**0**
1	1	1	0.5	1	0	1	0.5	**1**	**1**
1.5	0	0.5	0	0	0	0	0	0.5	**0**
2	0.5	1	0	1	0	1	0	1	0.5

(c) Information landscape based on sample

Figure 5.1: Rastrigin function in 1 dimension with associated information landscape based on a sample of nine solutions, including the optimum at position 0. Only the bolded cells are necessary to define the information landscape.

Borenstein and Poli [14, 15] proposed a number of calculations that can be performed on information landscapes to estimate the quality and quantity of information available to search algorithms. These are based on a measure called the *distance* between two information landscapes V_1 and V_2, defined as

$$D(V_1, V_2) = \frac{1}{n}\sum_{i=1}^{n} |V_{1i} - V_{2i}| \tag{5.4}$$

A measure of GA hardness is then proposed based on the distance between the information landscape of a problem and the information landscape of an 'optimal' landscape (referred to as V_{max}). An optimal landscape is one which is known to be easy for a given search algorithm, or on which the algorithm will perform maximally. A modified version of this information landscape hardness measure to quantify searchability, based on a sample from a continuous search space, is proposed in Section 5.2.2.

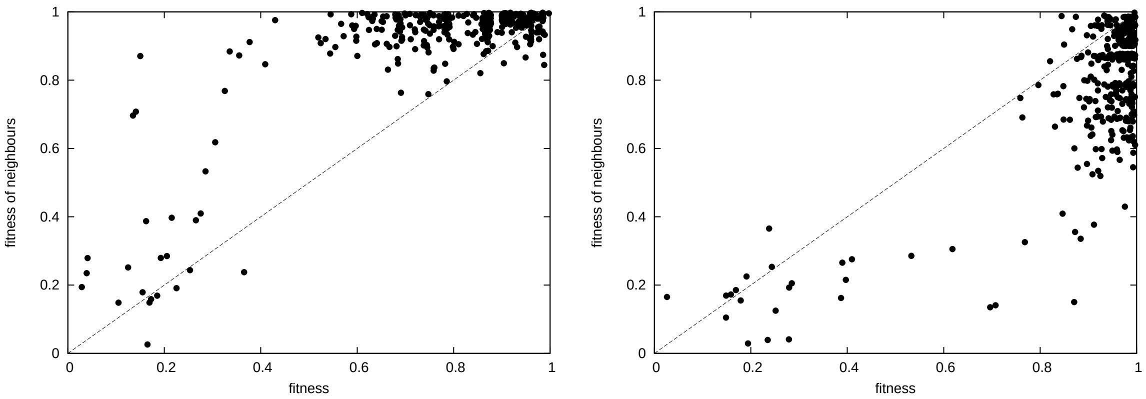
5.1.3 Fitness cloud

Verel *et al.* [173] introduced a technique, the fitness cloud, for visualizing evolvability of evolutionary search. The fitness cloud is a scatterplot showing the relationship between fitness values of parents and offspring. For each string x in the search space, a neighbour of x, called x', is determined based on some genetic search operator. The fitness cloud is then a scatterplot of all points $(f(x), f(x'))$ where f is the fitness function. The line $f(x) = f(x')$ in the scatterplot forms the division between points with good evolvability and points with bad evolvability. Points falling on the line are indicative of neutrality in a fitness landscape, where two neighbouring points have the same fitness. For example, the fitness cloud in Figure 5.2(a) gives a picture of better evolvability than the fitness cloud in Figure 5.2(b) since the majority of the points are above the $f(x) = f(x')$ line (dashed line on the graph). The shape of the fitness cloud scatterplot gives an indication of the evolvability of the given search operator on the given problem, and so provides some information on the difficulty of the problem.

Implementing a fitness cloud for a problem involves the following considerations:

- *Sampling methodology*: In the original study of fitness clouds [173] an exhaustive enumeration of the discrete search space was used as the basis for the fitness clouds. In most cases, it is not feasible to consider all possible solutions, so a methodology for sampling the space is required. A uniform random sample can be drawn, or some biased sampling methodology can be used, such as the Metropolis-Hastings technique [85] for discrete search spaces, that gives more weight to solutions with higher fitness, as proposed by Vanneschi *et al.* [164].

- *Definition of neighbourhood*: Neighbours of a sample of solutions could be defined



(a) Good evolvability

(b) Bad evolvability

Figure 5.2: Fitness cloud examples. Fitness values are normalised to range $[0, 1]$ where 0 is the worst and 1 is the best fitness.

using a heuristic-independent notion such as Hamming distance, or could be based on a particular search operator. If a definition of neighbourhood produces a number of possible neighbours, there is a further consideration of which neighbour to choose as the basis for the fitness cloud. For example, Vanneschi *et al.* [164] used fitness clouds in the context of genetic programming, where the neighbour was chosen using tournament selection on a sample generated using a subtree mutation operator. An alternative approach using fitness proportional selection was proposed by Poli and Vanneschi [118]. Lu *et al.* [83] showed that the neighbourhood sample size has a drastic influence on the fitness cloud generated and so argued that fitness clouds are an unreliable characterisation of evolvability.

A technique using particle swarm optimisation updates for determining neighbours as the basis of fitness clouds is proposed in Section 5.3.

5.1.4 Negative slope coefficient

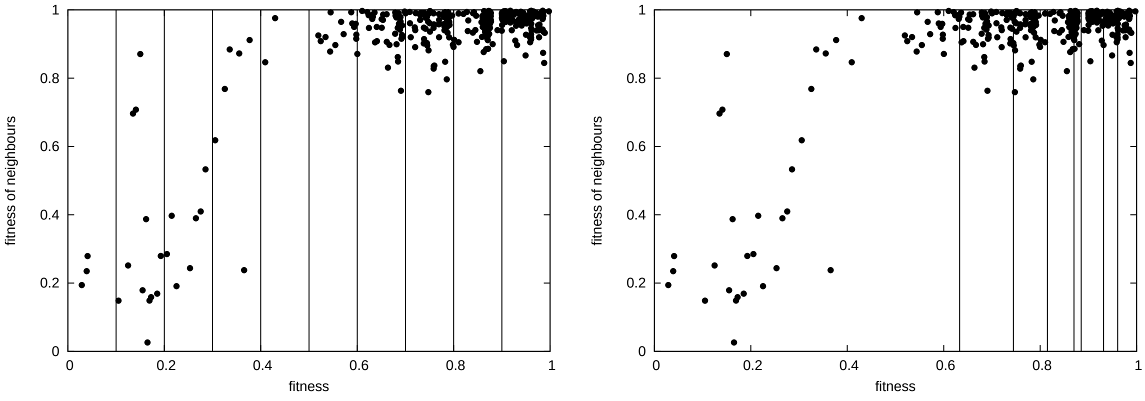
Vanneschi *et al.* [162, 164] proposed a measure of problem difficulty called the negative slope coefficient (NSC), which is based on the fitness cloud [173]. The NSC is a single value that quantifies the ability of genetic operators to produce offspring that are fitter than the parents. The NSC is calculated by partitioning the fitness cloud into discrete bins. Line segments are then defined between the centroids of adjacent bins. The negative slope coefficient is calculated as the sum of all negative slopes between segments. Vanneschi *et al.* [162, 164] hypothesised that the NSC measure could be used as a predictive difficulty measure for problems: if NSC=0 the problem is easy, but if NSC<0, the problem is difficult and smaller values indicate increased difficulty. The algorithm for calculating NSC is given in Algorithm 5.1. Vanneschi *et al.*'s proposed NSC measure was defined in the context of genetic programming, but a similar approach could be used in other algorithmic contexts, and this is investigated further for PSO in Section 5.3.2.

A significant aspect of computing the NSC measure involves deciding on the bin partitioning strategy (step 1 of Algorithm 5.1). In his thesis, Vanneschi [162] discussed the following strategies:

- *Bins of equal size*: This strategy involves an arbitrary decision on the number of bins and division of the range of fitness values into equal bin sizes. Figure 5.3(a) illustrates this approach where 10 bins of size 0.1 are used. A problem with this approach is that it can lead to bins containing too few points (bins with no points are even possible). The centroids of the bins (averages) would then lack statistical significance [162]. In a later study, Vanneschi *et al.* [166] showed experimentally that this approach resulted in misleading NSC values for some well-known genetic programming benchmark problems.

- *Bins containing equal numbers of points*: In this strategy, a decision is made on the minimum number of points in a bin. The bisection algorithm is used to recursively divide the bins into two bins containing the same number of points, until the threshold minimum number of points in a bin is reached. This approach is illustrated in Figure 5.3(b). A problem with this approach is that it can lead to bins of a very small size, sometimes resulting in the slopes between centroids of very small adjacent bins taking on very large values.

- *Size-driven bisection*: This approach divides the fitness cloud into bins taking both the size of the bins and the number of points they contain into account. The starting point is to partition the fitness cloud into two bins with equal numbers of



(a) Bins of equal size

(b) Bins containing equal numbers of points

Figure 5.3: Strategies for binning of fitness clouds.

points. After that, the bin with the larger size is partitioned in the same way. This process is continued until a bin contains fewer points than a threshold number of points or a bin becomes smaller than a minimum size. Vanneschi *et al.* showed that size-driven bisection resulted in NSC values that were reliable indicators of problem difficulty in the case of a number of genetic programming benchmark problems [166] and real world pharmaceutical applications [163]. Poli and Vanneschi [118] also provided a theoretical analysis of NSC (using fitness proportional selection of neighbours and size-driven bisection) that suggests that the measure can broadly discriminate between easy and hard genetic algorithm problems.

Although the usefulness of NSC with size-driven bisection has been demonstrated in the context of genetic programming, there are a number of drawbacks. One difficulty with the measure is the choice of parameters (the minimum number of points in a bin and the minimum size of a bin), which can dramatically influence the measure. In particular, it has been shown that the NSC measure tends to zero as the minimum number of points in a bin increases [168]. A further major drawback of the NSC measure is that no way has been found to normalise NSC values [162, 163, 166].

Section 5.3.2 proposes two NSC measures based on fitness clouds originating from PSO updates, but in Section 5.4 it is shown that the proposed measures are not reliable indicators of searchability.

Algorithm 5.1 Algorithm for calculating the NSC measure based on a fitness cloud.

```
1: Partition the horizontal axis of the fitness cloud (the $f(x)$ axis) into $m$ segments $I_1, I_2, \ldots, I_m$.
2: Partition the vertical axis of the fitness cloud (the $f(x')$ axis) into $m$ segments $J_1, J_2, \ldots, J_m$ so that each segment $J_i$ contains all the $f(x')$ values corresponding to the $f(x)$ values in $I_i$.
3: For each partition $I_i$, compute the average fitness value, $M_i$.
4: For each partition $J_i$, compute the average fitness value, $N_i$.
5: Define segments $S_1, S_2, \ldots, S_{m-1}$, such that $S_i$ is the segment from point $(M_i, N_i)$ to point $(M_{i+1}, N_{i+1})$.
6: For each segment $S_i$, calculate the slope $P_i$ using:
       $P_i = \frac{N_{i+1} - N_i}{M_{i+1} - M_i}$
7: Calculate the negative slope coefficient measure as:
       $NSC = \sum_{i=1}^{m-1} c_i$
   where
       $\forall i \in [1, m) : c_i = \begin{cases} P_i & \text{if } P_i < 0 \\ 0 & \text{otherwise} \end{cases}$
```

5.2 Proposed general searchability measures

The FDC measure [68] was originally proposed as a measure of problem difficulty based on the premise that search will be easier if there is high correlation between fitness values and the distance to the optimum in the case of minimisation problems and high anti-correlation between fitness values and distance to the optimum in the case of maximisation problems. If the measure instead quantifies the correlation between fitness values and the distance to the most fit value from a sample in place of the optimum, then the focus of the measure is changed to searchability, rather than problem difficulty.

In a similar way, Borenstein and Poli's information hardness measure [15], originally conceived as a measure of difficulty/deception, can be adapted to a measure of searchability if used with the fittest value from a sample instead of the optimal value.

5.2.1 Fitness distance correlation as a searchability measure

The following adapted FDC measure is proposed for continuous problems. Given a sample of n points, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, from the search space, with associated fitness values $F = \{f_1, \ldots, f_n\}$, the fittest point in the sample is determined and denoted $\mathbf{x}^*$. The Euclidean distances of every point $\mathbf{x}_i$ from $\mathbf{x}^*$ are calculated and denoted as $Dist^* = \{d_1^*, \ldots, d_n^*\}$. The fitness distance correlation searchability (FDC$_s$) measure is defined as the covariance of F and $Dist^*$ divided by the product of the standard deviation of F and standard deviation of $Dist^*$. Since these are samples, this can be estimated as

$$\mathrm{FDC}_s = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(f_i - \overline{f})(d_i^* - \overline{d^*})}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(f_i - \overline{f})^2}\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i^* - \overline{d^*})^2}} \tag{5.5}$$

which can be simplified to:

$$\mathrm{FDC}_s = \frac{\sum_{i=1}^{n}(f_i - \overline{f})(d_i^* - \overline{d^*})}{\sqrt{\sum_{i=1}^{n}(f_i - \overline{f})^2}\sqrt{\sum_{i=1}^{n}(d_i^* - \overline{d^*})^2}} \tag{5.6}$$

where $\overline{f}$ and $\overline{d^*}$ are the means of F and $Dist^*$, respectively.

5.2.2 Information landscape negative searchability measure

Borenstein and Poli's information landscape hardness measure [15] is based on the difference between the information landscape vector of a problem and a reference landscape vector (called V_{max} in the original study, but denoted using $\mathbf{v}_r$ in this study). Using a reference landscape to estimate hardness of a problem p with information landscape vector $\mathbf{v}_p$ requires the following:

- Calculation of the distance between $\mathbf{v}_r$ and $\mathbf{v}_p$ requires that the number of elements in the two vectors must be the same. This means that the number of points in solution space used to calculate the information landscape of the problem and the optimal landscape must be equal.

- The locations of solutions within the problem space of problem p and the reference problem must coincide.

- The position of the optimum must be the same for both information landscapes, otherwise the distance between the landscapes will not make sense as a measure of hardness.

Implementing the information landscape hardness measure for continuous problems required a reference landscape that could match any problem in terms of dimensionality and the domain of the solution space. In addition, the position of the global optimum of the reference landscape should be able to be set to coincide with the estimated optimum of the problem landscape. The well-known Spherical function in D dimensions $\left(f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2\right)$ can serve the purpose of such a reference landscape for the following reasons:

- The Spherical function is an 'optimal' landscape in that it presents no negative information for search: if any point $\mathbf{x}_i$ has a lower fitness value than another point $\mathbf{x}_j$, then $\mathbf{x}_i$ will be closer to the optimum than $\mathbf{x}_j$, and this is true whatever the domain.

- The Spherical function can be defined up to any dimension and is defined for all values of $\mathbf{x}$, so the domain can be set to match the domain of any real-encoded problem.

- The Spherical function can be shifted so that the optimum is positioned anywhere in the search space, so that it coincides with the estimated optimum of the problem landscape. Given a point $\mathbf{s} = (s_1, s_2, ..., s_D)$ in D-dimensional problem space, the Spherical function with the optimum point shifted to position $\mathbf{s}$ is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 - 2s_i x_i + {s_i}^2$$

Given the above, it is proposed that Borenstein and Poli's information landscape hardness measure [15] be adapted to an information landscape negative searchability measure (IL$_{ns}$) using the approach outlined in Algorithm 5.2. The measure is referred to as a

Algorithm 5.2 Algorithm for computing the IL_{ns} (information landscape negative searchability) measure for a minimisation problem.

```
Generate a sample of $n$ random points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from a uniform distribution of the search space of problem $p$ with dimension $D$.
Determine the position of the fittest solution in the sample, $\mathbf{x}^*$.
Construct vector $\mathbf{v}_p$ representing the information matrix of the problem using Equation 5.3.
Define reference function $f_r$ as $f_r(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 - 2x_i^* x_i + x_i^{*2}$.
Using the same sample of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and based on $f_r$, construct vector $\mathbf{v}_r$ representing the information matrix of the reference landscape.
Compute $\text{IL}_{ns}$ as the difference between $\mathbf{v}_p$ and $\mathbf{v}_r$ using Equation 5.4.
```

negative searchability measure because high values are indicative of bad information for search.

5.2.3 Summary of proposed general searchability measures

The proposed general searchability measures are summarised in Table 5.1. Both measures have linear time efficiency with respect to the size of the sample, but the IL_{ns} has polynomial memory requirements with respect to the sample size.

5.3 Proposed PSO searchability measures

Fitness clouds and the NSC measure originated in the context of discrete problems solved using evolutionary algorithms. This section investigates the notion of fitness clouds for continuous problems solved using PSOs. Single-valued measures based on fitness clouds are proposed, including a version of NSC based on PSO updates.

Table 5.1: Proposed general measures of searchability

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
FDC$_s$	Fitness distance correlation searchability measure.	size of sample, n.	$[-1, 1]$: For a minimisation problem, 1 indicates the highest measure of searchability (perfect correlation between fitness values and distance to the fittest solution).	n fitness evaluations and n distance calculations in solution space.
IL$_{ns}$	Information landscape negative searchability measure.	size of sample, n.	$[0, 1]$: A value of 0 indicates maximum searchability (no difference from the reference landscape vector $\mathbf{v}_r$).	$2n$ fitness evaluations for problem and reference landscape; $(n-1)(n-2)$ memory requirement for both information landscape vectors.

5.3.1 Determining neighbours for fitness clouds using PSO

Constructing a fitness cloud for a given problem requires a sample of solutions and neighbours of those solutions. In the original fitness cloud publication [173] two solutions are regarded as neighbours if there is "*a transformation related to a local search heuristic or an operator which allows it to pass*" from one solution to another. For PSO algorithms, the search operators are in the form of position update equations. The two most contrasting position update models in terms of exploration/exploitation would be the cognitive-only model (using only the personal best as a guide) and the social-only model (using only the global best as a guide). These two models are proposed for calculating neighbours to be used as the basis for generating fitness clouds.

Given a problem in multi-dimensional floating-point space, the position of each particle i at iteration t of the algorithm can be represented as $\mathbf{x}_i(t)$. For the PSO models used in this study, the swarm of particles at $t = 0$ are initialised with random positions and at each iteration of the algorithm, the positions of particles are updated as follows:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{5.7}$$

where $\mathbf{v}_i(t+1)$ is the velocity of particle i at time $(t+1)$. The two contrasting velocity update models used for determining neighbours are as follows:

1. Cognitive-only PSO update [73]:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 \cdot \mathbf{r}_1(t) \odot (\mathbf{y}_i(t) - \mathbf{x}_i(t)) \tag{5.8}$$

   where w is the inertia weight, c_1 is the cognitive acceleration constant, $\mathbf{r}_1(t) \sim U(0,1)^D$ where D is the dimension of the problem, $\odot$ denotes element-by-element vector multiplication, $\mathbf{y}_i(t)$ refers to particle i's personal best position, and $\mathbf{y}_i(0) \neq \mathbf{x}_i(0)$. The cognitive-only PSO update results in high exploration, since each particle is pulled in the direction of its own best position, resulting in essentially a swarm of individual local search optimizers.

2. Social-only PSO update [73]:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_2 \cdot \mathbf{r}_2(t) \odot (\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)) \tag{5.9}$$

   where c_2 is the social acceleration constant, $\mathbf{r}_2(t) \sim U(0,1)^D$, and $\hat{\mathbf{y}}(t)$ refers to the global best position at time step t, being the best solution from the personal best positions of all particles. The social-only update model results in faster exploitation than the cognitive-only update model, since all particles are pulled in the direction of the same global best particle at each time step.

The approach used for constructing the fitness cloud of a minimisation problem using PSO updates for determining neighbours is given in Algorithm 5.3. The basic idea is to perform two PSO updates on the initial swarm of solutions to determine the neighbours. The reason for two updates is that the initial velocity of all particles is zero. This means that the inertia term ($w \cdot \mathbf{v}_i(t)$ in Equations 5.8 and 5.9) will be zero for the first update and will only come into effect on the second iteration of the algorithm. Note that steps

Algorithm 5.3 Algorithm for constructing a fitness cloud of a minimisation problem using PSO updates to determine neighbours of a sample.

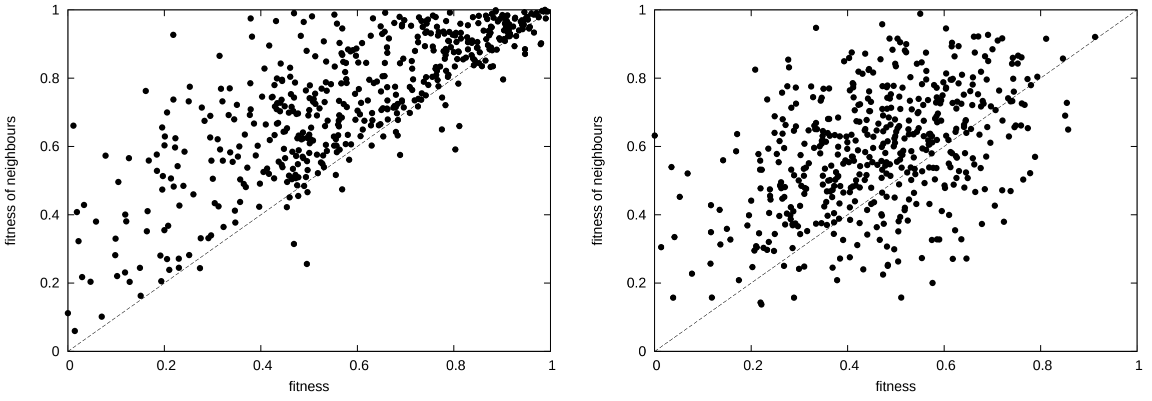
```
Generate a sample swarm of $n$ random solution vectors for iteration 0: $\mathbf{x}_1(0), \ldots, \mathbf{x}_n(0)$ from a uniform distribution of the search space of the problem.
Determine the fitness values of all solutions, $f(\mathbf{x}_1(0)), \ldots, f(\mathbf{x}_n(0))$.
for each solution, $\mathbf{x}_i(0)$, generate a personal best position, $\mathbf{y}_i(0)$ as follows:
    Generate a new position vector, $\mathbf{z}$, a small distance from $\mathbf{x}_i(0)$, but still in the domain of the problem, by adding Gaussian noise with a standard deviation equal to 10% of the range of the problem to each component of $\mathbf{x}_i(0)$.
    Determine the fitness value of the new position vector, $f(\mathbf{z})$.
    If $f(\mathbf{z}) < f(\mathbf{x}_i(0))$, then set $\mathbf{y}_i(0) = \mathbf{z}$.
    Else set $\mathbf{y}_i(0) = \mathbf{x}_i(0)$ and set $\mathbf{x}_i(0) = \mathbf{z}$.
end for
Determine the global best solution of iteration 0, $\hat{\mathbf{y}}(0)$, selected from the personal best positions $\mathbf{y}_i(0)$.
Set all initial velocities $\mathbf{v}_i(0)$ to zero.
For each $\mathbf{x}_i(0)$, determine the velocity update $\mathbf{v}_i(1)$ using the relevant PSO update equation and calculate the iteration 1 positions: $\mathbf{x}_1(1), \ldots, \mathbf{x}_n(1)$.
Determine the iteration 1 personal best solutions $\mathbf{y}_1(1), \ldots, \mathbf{y}_n(1)$ and the global best solution $\hat{\mathbf{y}}(1)$, only considering points in bounds as personal best and global best candidates.
For each $\mathbf{x}_i(1)$, determine the $2^{nd}$ iteration velocity update using the relevant PSO update equation and calculate $\mathbf{x}_i(2)$ to form the neighbours of the initial sample.
Determine the fitness values of all neighbours, $f(\mathbf{x}_1(2)), \ldots f(\mathbf{x}_n(2))$.
Normalise the fitness values of all points, $f(\mathbf{x}_1(0)), \ldots, f(\mathbf{x}_n(0))$, and the fitness values of all neighbours, $f(\mathbf{x}_1(2)), \ldots f(\mathbf{x}_n(2))$, to the range $[0, 1]$, where 0 is the worst fitness and 1 is the best fitness.
Considering only those points that stayed within the bounds of the domain of the problem in the $2^{nd}$ position update, generate the fitness cloud from the normalised fitness values.
```

3 to 8 of Algorithm 5.3 is to ensure that the personal best particle is not the same as the current particle in the first iteration. If this was not done, then both the first and second terms of Equation 5.8 for the cognitive-only model will be zero, resulting in no particles moving. The strategy used to prevent this is to generate a new random solution a small distance from the initial solution (by adding Gaussian noise) and to swap these points if the new solution is not better than the initial solution. In this way a personal best is generated for each individual and the particles are able to build up some velocity.

Steps 9 to 14 are self-explanatory. In step 15, the fitness values of all initial points and neighbours are normalised to the range $[0, 1]$, where 0 is the worst fitness and 1 is the best fitness (note that the best and worst fitnesses are as encountered during execution of the algorithm, so that the algorithm can be run on unknown problems). Normalising the fitness values in this way effectively converts the minimisation problem into a maximisation problem for the purposes of the fitness cloud visualisation and allows for comparisons between fitness clouds of different problems. If the fitness cloud scatterplot is drawn with the original fitness values of a minimisation problem, it will have to be interpreted in the opposite way to the original fitness cloud approach. Points below the diagonal would be regarded as having good searchability, rather than bad searchability. This ‘upside down’ fitness cloud causes problems later with the computation of the negative slope coefficient measure, since a negative slope would indicate good searchability, rather than bad searchability. To avoid this confusion and need to redefine terminology, the fitnesses are converted to behave as for a maximisation problem.

Step 16 of Algorithm 5.3 highlights an issue particular to continuous domain problems with boundary constraints. When optimizing such problems using PSO algorithms, there is a good chance that particles will leave the search space, even in the first iteration of the algorithm. This was proved theoretically to be the case even when initial velocities are set to zero [58], particularly in the case of high-dimensional search spaces. To avoid the fitness cloud containing fitness values corresponding to points outside the domain of the problem, particles that leave the search space in the second iteration of the algorithm are ignored. This means that the final fitness cloud could contain fewer fitness pairs than the initial number of points in the sample.

Figure 5.4 shows two sample fitness clouds produced as a result of running Algo-



(a) $2D$ Spherical

(b) $2D$ Rastrigin

Figure 5.4: Fitness clouds from sample runs of Algorithm 5.3 on two different $2D$ functions based on cognitive PSO updates.

rithm 5.3 using the PSO cognitive updates on two-dimensional functions Spherical and Rastrigin. Figure 5.4(a) clearly shows that for the Spherical function, in most cases, the fitness improved when particles moved in the direction of their personal best position for two iterations. In the case of the Rastrigin function, there were more particles that decreased in fitness, as is evident by the larger number of points below the diagonal in Figure 5.4(b).

5.3.2 Proposed measures based on fitness clouds

The first proposed single-valued measure based on a fitness cloud is simply the proportion of fitness improving points in the cloud. This is termed the fitness cloud index (FCI) and is calculated as follows: Given an initial sample of random solutions, and after determining the neighbours given some algorithm operator/update strategy, the set of valid points with neighbours are determined, where valid in the PSO context implies that the points remain within the bounds of the search space after the update operations. The FCI is defined as the proportion of fitness-improving elements in the valid set of points and neighbours. More precisely: given a minimisation problem with a fitness function f and a sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n points with associated neighbours $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$, determine the subset $S_v \subseteq S$ of n_v valid points, such that S_v consists of all $\mathbf{x}_i \in S$ where

$\mathbf{x}'_i$ is within the bounds of the search space. The FCI measure is then defined as:

$$FCI = \frac{\sum_{i=1}^{n_v} g(\mathbf{x}_i)}{n_v} \tag{5.10}$$

where

$$g(\mathbf{x}_i) = \begin{cases} 1 & \text{if } f(\mathbf{x}'_i) < f(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

Note that the FCI measure is based on a simple comparison of neighbouring fitness values, so does not require normalisation of the fitness values or conversion to a maximisation problem (step 15 of Algorithm 5.3). By definition of being a proportion, the FCI measure is normalised to the range $[0, 1]$, where 0 indicates the worst possible searchability and 1 indicates perfect searchability of the problem with respect to the given search operator. The two variations of FCI based on cognitive and social PSO updates are summarised in Table 5.2.

The NSC measure, described in Section 5.1.4 is also a single measure based on a fitness cloud, but involves a more intricate extraction of evolvability of different segments of the fitness cloud. The cloud is divided into vertical bins, representing bands of fitness values. A negative slope in a line segment from the centroid of one fitness band to another is indicative of negative evolvability in that segment. The NSC measure therefore quantifies the evolvability of portions of the fitness cloud, one at a time, and then adds the results, ignoring portions with positive evolvability (positive slopes). Given a fitness cloud based on PSO updates, Algorithm 5.1 can be used to compute the NSC measure. The notations NSC$_{cog}$ and NSC$_{soc}$ are used to denote the NSC measures derived using the cognitive and social PSO update strategies, respectively. Features of these measures are given in Table 5.2. Disadvantages of the NSC measures are the many parameters and the unbounded range of possible resulting values.

5.4 Experimentation on one-dimensional problems

This section performs experiments on the proposed measures of searchability summarised in Tables 5.1 and 5.2. Simple one-dimensional benchmarks are used to see if the proposed

Table 5.2: Proposed measures of searchability based on fitness clouds

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
FCI_{cog} FCI_{soc}	Fitness cloud index based on cognitive or social PSO updates.	(1) size of sample, n, (2) inertia weight, w, (3) acceleration constants, c_1 (for FCI_{cog}) or c_2 (for FCI_{soc}).	$[0, 1]$: indicating the proportion of fitness improving solutions after two PSO updates.	$3n$ fitness evaluations, $2n$ solution updates.
NSC_{cog} NSC_{soc}	Negative slope coefficient with neighbourhood defined using cognitive or social PSO updates.	(1) size of sample, n, (2) inertia weight, w, (3) acceleration constants, c_1 (for NSC_{cog}) or c_2 (for NSC_{soc}), (4) minimum number of points in a bin, (5) minimum size of a bin.	$(-\infty, 0]$: A value of 0 indicates maximum searchability (no negative slopes between centroids of bins in the fitness cloud). Smaller values indicate decreased searchability.	$3n$ fitness function evaluations, $2n$ solution updates.

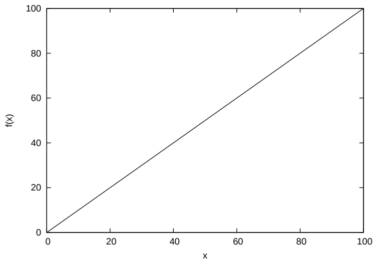
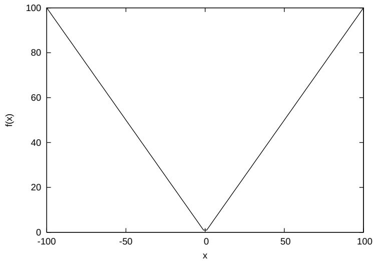
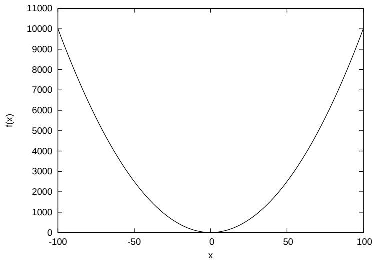
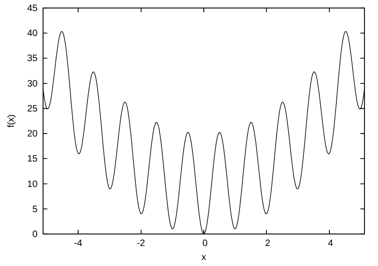
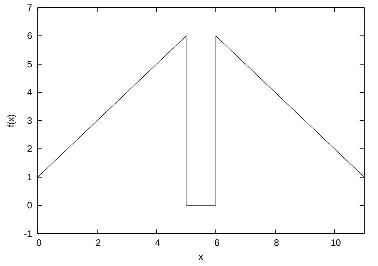
measures give expected results, based on a visual inspection of functions. Section 5.5 experiments with higher dimensional problems.

5.4.1 Benchmark functions

Five simple benchmark functions were selected with expected decreasing searchability. These functions are defined and illustrated in Table 5.3. The expected results of each of the proposed measures listed in Tables 5.1 and 5.2 on the benchmark functions are discussed in this section.

The FDC_s metric quantifies the correlation between fitness values and distance to the fittest solution of the sample. For both the Straight and Absolute Value benchmarks, this value should be at the maximum (+1) because the functions are linear. For the Spherical function the value should be close to 1, because there is a positive correlation,

Table 5.3: One-dimensional benchmark functions with different searchability characteristics.

Straight	$f(x) = x,$ where $x \in [0, 100]$	
Absolute Value	$f(x) = \|x\|,$ where $x \in [-100, 100]$	
Spherical	$f(x) = x^2,$ where $x \in [-100, 100]$	
Rastrigin	$f(x) = x^2 - 10\cos(2\pi x) + 10,$ where $x \in [-5.12, 5.12]$	
Hole-in-Mountain	$f(x) = x + 1,$ where $x \in [0, 5)$ $= 0,$ where $x \in [5, 6]$ $= -x+12,$ where $x \in (6, 11]$	

but not a perfect linear correlation between fitness and distance. Rastrigin should give a lower FDC$_s$ value than Spherical, but still an overall positive correlation due to the underlying spherical shape of Rastrigin. It is expected that Hole-in-Mountain give a negative value for FDC$_s$ due to the deceptive structure of the function.

The IL$_{ns}$ measure quantifies the difference in the information landscape between the benchmark and the shifted Spherical function. For the Straight, Absolute Value and Spherical benchmarks, the IL$_{ns}$ measure should be close to 0, as the information is identical to the shifted spherical function (recall that the information landscape captures information on whether there are fitness differences between points or not, not the magnitude of the difference). Since the optimum of the Spherical function is shifted to the position of the fittest solution from a sample of the benchmark (and not necessarily the optimum of the benchmark), there may be slight differences in the information landscapes, but these should be small. The IL$_{ns}$ measure should increase for Rastrigin and be closer to 1 for Hole-In-Mountain.

The FCI measures simply quantify the proportion of fitness improving elements in the fitness cloud. Both FCI measures should yield values of 1 for the Straight function, since if any particle is pulled in the direction of a better particle, regardless of whether it is a global or personal best guide, the fitness can only improve. The Absolute Value and Spherical functions should give good FCI values (close to 1). Some points may be below the diagonal in the fitness cloud, since the fitness of a particle can deteriorate if it moves in the direction of a better particle, but overshoots the global minimum and moves to a position higher than the original position. In the case of the Rastrigin function, due to the ruggedness, there are many opportunities for the fitness of a particle to deteriorate if pulled in the direction of a better particle, regardless of whether it is a global or personal best guide, so lower FCI values are expected. The Hole-in-Mountain function should result in low searchability for the social-only PSO model (lower FCI values) and relatively high searchability for the cognitive-only PSO model. For the social-only model, assuming one of the initial random points was positioned in the 'hole' (the global minimum plateau), all particles will be pulled towards the centre, resulting in an increase (deterioration) of fitness for many particles. Although this is the desired behaviour for a search algorithm (moving towards the global optimum), the measure predicts searchability, not optimality. In the case of the cognitive-only model, the simple linear slopes that define most of the Hole-in-Mountain function should give a similar, but slightly worse, searchability profile to the Absolute Value benchmark.

For the same reasons as described above, both NSC measures should result in 0 for the Straight function (perfect searchability), values close to 0 (small negative) for Absolute Value and Spherical, and smaller values (larger negative) for Rastrigin. The Hole-in-Mountain should have a smaller (larger negative) NSC$_{soc}$ value than the other functions, but a reasonably good NSC$_{soc}$ value.

5.4.2 Experimental setup

For each benchmark problem, 30 independent runs of the algorithms for computing each measure were performed. The calculations of all measures were based on sample sizes of 500 points (randomly sampled from a uniform distribution). For the PSO updates, the inertia weight (w), cognitive acceleration (c_1) and social acceleration (c_2) were set to the popular values of 0.7298, 1.496, and 1.496, respectively [32], a parameter choice that guarantees convergence [160]. For the NSC measures, size-driven bisection was used for partitioning the fitness clouds with the minimum number of points in a bin set to 30 and the minimum size of a bin set to 5% of the range of the problem.

5.4.3 Results and discussion

Table 5.4 lists the mean measures over 30 runs for each benchmark function with standard deviations shown below the means in parentheses. A study of the values reveals the following:

- The values for the FDC$_s$ and IL$_{ns}$ measures are in line with the expected values as discussed in Section 5.4.1. Relatively low standard deviations for these first two measures also indicate that the measures are fairly reliable.

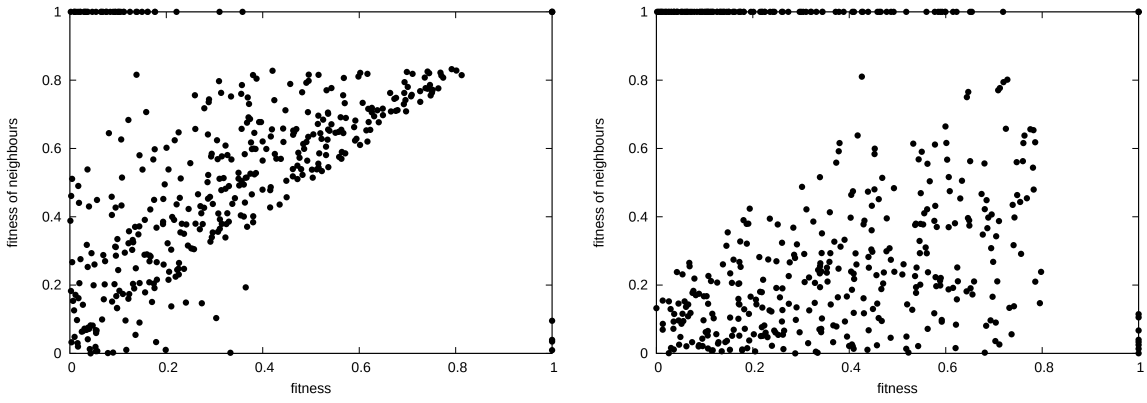
- The values for the FCI$_{cog}$ are in line with the expected values. Slightly lower FCI$_{soc}$ values for the Absolute Value and Spherical functions indicate that more particles overshot the minimum to higher fitness values, most probably due to the higher

Table 5.4: Values of the FDC$_s$, IL$_{ns}$, FCI and NSC measures for the one-dimensional problems shown in Table 5.3. Values are averages over 30 runs, each with 500 random initial points. Standard deviations are given below each value in parentheses.

Function	**FDC**$_s$	**IL**$_{ns}$	**FCI**$_{cog}$	**FCI**$_{soc}$	**NSC**$_{cog}$	**NSC**$_{soc}$
Straight	1 (±0)	0 (±0)	1 (±0)	1 (±0)	-2.272 (±2.382)	-0.235 (±0.312)
Absolute Value	1 (±0)	0.002 (±0.002)	0.962 (±0.008)	0.897 (±0.015)	-6.833 (±4.287)	-12.777 (±5.292)
Spherical	0.968 (±0.002)	0.002 (±0.001)	0.965 (±0.010)	0.897 (±0.016)	-11.587 (±6.004)	-19.832 (±5.998)
Rastrigin	0.709 (±0.018)	0.254 (±0.011)	0.781 (±0.023)	0.800 (±0.017)	-8.509 (±5.180)	-13.974 (±7.999)
Hole-in-Mountain	-0.401 (±0.052)	0.795 (±0.031)	0.918 (±0.016)	0.405 (±0.047)	-0.622 (±0.862)	-2.914 (±1.589)

velocities (based on larger distances to the global best compared to the personal best). As predicted, the Hole-in-Mountain function has high searchability for the cognitive-only PSO model (0.918) and low searchability for the social-only PSO model (0.405). To see this difference visually, the fitness clouds of sample runs on the Hole-in-Mountain function are plotted in Figure 5.5. It is clear from the plots that the cognitive PSO updates result in high levels of searchability, while the social PSO updates result in low levels of fitness improvement.

- All of the NSC values are negative, which is not in line with the expected values. For example, the Straight function should have perfect PSO searchability and yet the mean NSC values are negative, which is supposed to indicate areas of negative searchability. It is also unexpected that the Spherical function have the lowest NSC measures across all problems. The reasons for these unexpected results are investigated further in Section 5.4.4.



(a) Cognitive PSO updates

(b) Social PSO updates

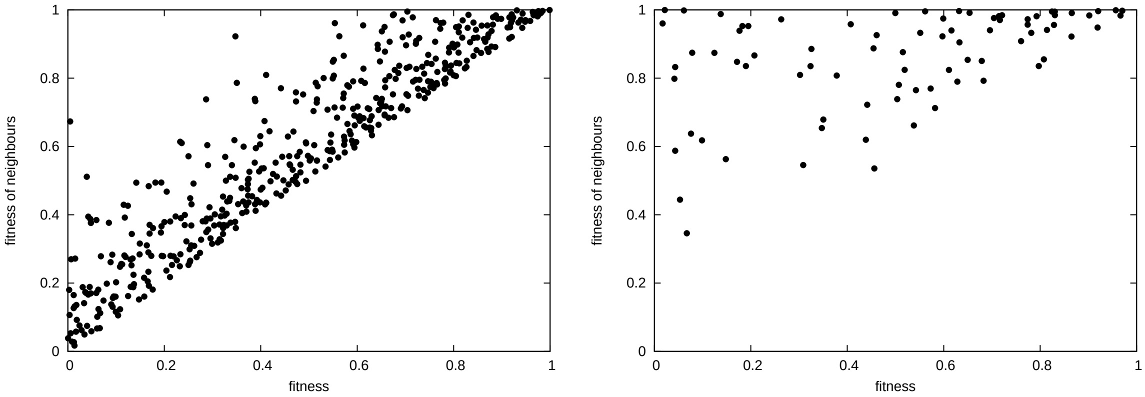
Figure 5.5: Fitness clouds from sample runs on Hole-in-mountain function for different PSO update strategies.

5.4.4 Investigation into NSC measure for PSO updates

The experiments on simple one-dimensional problems show that the NSC values did not give results as predicted. This section investigates possible reasons why.

Variable sizes of samples

A possible reason for the unexpected NSC values could be the variable sample size. As discussed in Section 5.3, many particles can leave the search space even during the first few iterations. Step 16 of Algorithm 5.3 for constructing a fitness cloud, simply ignored all points outside the search space after the second position update. To illustrate the impact of this, fitness clouds of sample runs of the algorithm on the Straight function are plotted in Figure 5.6. The scatterplots highlight the high variability in the number of points that remain within the boundaries of the search space. In the case of the social-only PSO only 84 out of the initial 500 points stayed within the bounds after two iterations and could be used in the fitness cloud. The reason more particles leave the search space in the case of the social-only model is that the velocities are based on the difference between the positions of each particle and the global best particle. Since these distances are in most cases large with respect to the search space in the first few



(a) Cognitive PSO (468)

(b) Social PSO (84)

Figure 5.6: Fitness clouds from sample runs on the Straight function for different PSO update strategies starting with 500 initial points. Number of points remaining in the search space are given in brackets.

iterations of the algorithm, the velocities will be high. This problem of variable numbers of fitness cloud pairs could be a reason for the unpredictable NSC values.

To investigate this further, a simplified example is used to illustrate how the NSC measure can be affected by a slight change in the number of points on which the calculation is based. Consider the one-dimensional Rastrigin benchmark function with domain $[-3, 3]$ as illustrated in Figure 5.7. In the figure, the fitness values have been normalised to the range $[0, 1]$, where 1 is the best fitness and 0 is the worst fitness. A sample of ten x positions are shown as black dots on the graph. The global best position is the particle with a normalised fitness of 0.9. Assuming a social-only PSO model for determining neighbours, all particles will be pulled in the direction of the the global best particle. Due to the acceleration coefficient and the random factor, this could result in small or large velocity updates. If the velocity update is small, a particle will be pulled a short distance in the direction of the global best particle, but if the velocity update is large a particle will be pulled a large distance in the direction of the global best particle, sometimes overshooting the global best particle's position. In some cases particles will move beyond the boundaries of the search space, in which case these particles are ignored in the calculation of the NSC measure. Assume that the ten particles move to new positions

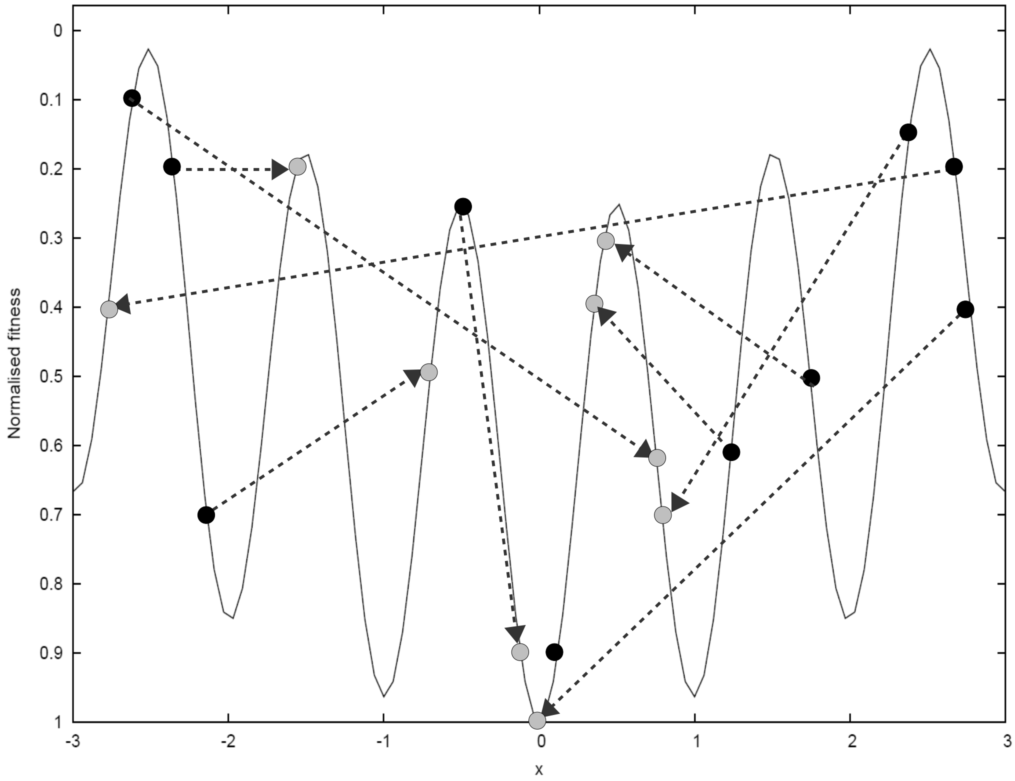


Figure 5.7: Rastrigin function with domain $[-3, 3]$ as an illustrative example. The initial random sample of solutions are indicated by black dots. New positions are indicated by dashed arrows and grey dots.

after the PSO updates, as illustrated with the dashed arrows and grey dots in Figure 5.7. The grey dots therefore represent the neighbours of the initial points to be used as the basis for the fitness cloud. Note that the global best particle from the initial sample in this case did not move.

The fitness values of the initial points with the neighbours will result in the $(f(x), f(x'))$ pairs as given in Table 5.5 (sorted in ascending order based on the fitness of the initial points $(f(x))$). Using size-driven bisection, the points are first divided into two bins with equal numbers of points. After that, the largest bin is repeatedly divided into two bins with equal numbers of points until some threshold minimum number of points or range. In this simplistic example, the threshold number of points is 2 and the range is 0.1. As

Table 5.5: Fitness pairs corresponding to the points and neighbours as illustrated in Figure 5.7 with resulting bins and midpoints.

$f(x)$	$f(x')$	1^{st} bin partitioning	2^{nd} bin partitioning	bin centroids
0.1	0.6	1 (range = 0.15)	1 (range = 0.15)	$(0.18, 0.56)$
0.15	0.7			
0.2	0.2			
0.2	0.4			
0.25	0.9			
0.4	1	2 (range = 0.5)	2 (range = 0.2)	$(0.5, 0.57)$
0.5	0.3			
0.6	0.4			
0.7	0.5		3 (range = 0.2)	$(0.8, 0.7)$
0.9	0.9			

shown in Table 5.5, this sample of points will result in three bins when the process of bin partitioning stops as the first largest bin (bin 2) cannot be partitioned further due to the minimum threshold of 2 points in a bin. The centroids of each of the three bins are then calculated. The NSC measure is calculated as the sum of all negative slopes between the centroids of adjacent bins. The slopes are illustrated in Figure 5.8(a), resulting in an NSC value of 0.

In a slightly different scenario, one of the points leaves the search space and its associated fitness pair $(0.2, 0.4)$ is not part of the fitness cloud. This results in completely different bin partitioning as shown in Table 5.6. The slopes associated with the new bin centroids are illustrated in Figure 5.8(b), resulting in an NSC value of -2.67.

This simple example shows how the same problem on two different runs can take on very different NSC values and could explain the unpredictable results of Table 5.4.

Fitness cloud modification: keeping particles in bounds

To avoid the situation where fitness clouds have variable numbers of points, a modification to Algorithm 5.3 is proposed, where particles leaving the search space are repaired

Table 5.6: Sorted fitness pairs corresponding to the points and neighbours as illustrated in Figure 5.7 with one fewer fitness pair $(0.2, 0.4)$, illustrating the effect on bin partitioning.

$f(x)$	$f(x')$	1^{st} bin partitioning	2^{nd} bin partitioning	3^{rd} bin partitioning	bin centroids
0.1	0.6	1 (range = 0.3)	1 (range = 0.3)	1 (range = 0.1)	$(0.15, 0.5)$
0.15	0.7				
0.2	0.2				
0.25	0.9			2 (range = 0.15)	$(0.325, 0.95)$
0.4	1				
0.5	0.3	2 (range = 0.4)	2 (range = 0.1)	3 (range = 0.1)	$(0.55, 0.35)$
0.6	0.4				
0.7	0.5		3 (range = 0.2)	4 (range = 0.2)	$(0.8, 0.7)$
0.9	0.9				

by setting them at the boundary. The modified algorithm for constructing a fitness cloud is given in Algorithm 5.4.

Given this change in the algorithm for constructing a fitness cloud, the definition of the fitness cloud index (given in Equation 5.11) also changes slightly as follows: Given a minimisation problem with a fitness function f and a sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n points with associated neighbours $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$, the FCI measure is defined as:

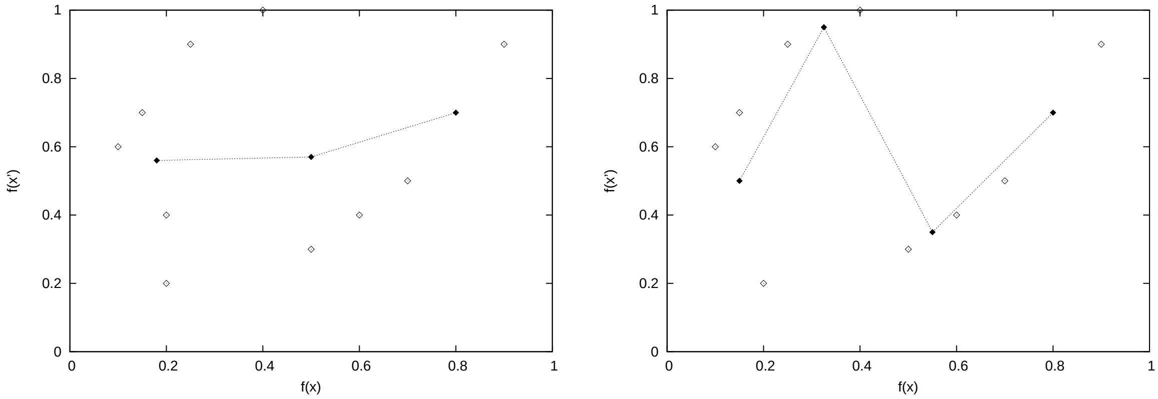
$$FCI = \frac{\sum_{i=1}^{n} g(\mathbf{x}_i)}{n} \tag{5.12}$$

where

$$g(\mathbf{x}_i) = \begin{cases} 1 & \text{if } f(\mathbf{x}'_i) < f(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

Revised experimentation on FCI and NSC with particles kept in search space

Based on the modified algorithm for constructing a fitness cloud, the experiments on the one-dimensional problems were re-run and the results are summarised in Table 5.7.



(a) Fitness cloud with 10 points (NSC = 0).

(b) Fitness cloud with 9 points (NSC = -2.67).

Figure 5.8: Fitness clouds of 10 and 9 points. Filled in diamonds give the centroids of bins. One less point $(0.2, 0.4)$ in (b) results in more bins introducing a negative slope between the 2^{nd} and 3^{rd} centroids.

Algorithm 5.4 Modified algorithm for constructing a fitness cloud of a minimisation problem using PSO updates to determine neighbours of a sample.

```
Steps 1 to 10 are the same as Algorithm 5.3.
⋮
11: For each $\mathbf{x}_i(0)$, determine the velocity update $\mathbf{v}_i(1)$ using the relevant PSO update equation and calculate the iteration 1 positions: $\mathbf{x}_1(1), \ldots, \mathbf{x}_n(1)$, repairing any positions outside the bounds of the search space on any dimension to be set on the boundary for that dimension.
12: Determine the iteration 1 personal best solutions $\mathbf{y}_1(1), \ldots, \mathbf{y}_n(1)$ and the global best solution $\hat{\mathbf{y}}(1)$, selected from the set of personal best positions $\mathbf{y}_i(1)$.
13: For each $\mathbf{x}_i(1)$, determine repaired positions $\mathbf{x}_i(2)$ based on calculated $\mathbf{v}_i(2)$ velocity updates (as in step 11).
14: Determine the fitness values of all iteration 2 positions, $f(\mathbf{x}_1(2)), \ldots f(\mathbf{x}_n(2))$.
15: Normalise the fitness values of all initial points, $f(\mathbf{x}_1(0)), \ldots, f(\mathbf{x}_n(0))$, and final neighbours, $f(\mathbf{x}_1(2)), \ldots f(\mathbf{x}_n(2))$, to the range $[0, 1]$, where 0 is the worst fitness and 1 is the best fitness, and generate the fitness cloud from the normalised fitness values.
```

Table 5.7: Values of the FCI and NSC measures for one-dimensional problems shown in Table 5.3, based on revised fitness clouds (with particles that leave the search space being repaired by setting them at the boundary). Values are averages over 30 runs, each with 500 random initial points. Standard deviations are given below each value in parentheses.

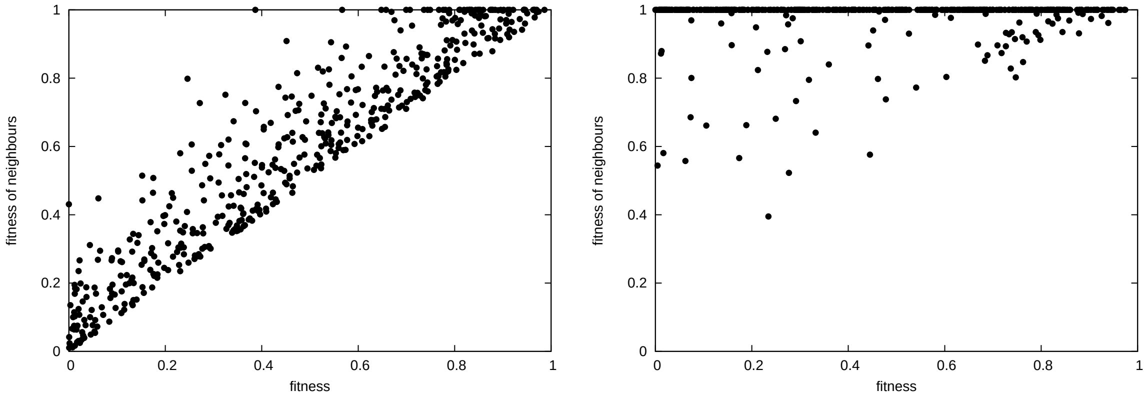
Function	**FCI**$_{cog}$	**FCI**$_{soc}$	**NSC**$_{cog}$	**NSC**$_{soc}$
Straight	1 (±0)	1 (±0)	-1.575 (±1.602)	-4.874 (±2.460)
Absolute Value	0.963 (±0.008)	0.878 (±0.014)	-2.210 (±2.190)	-4.634 (±4.345)
Spherical	0.963 (±0.008)	0.881 (±0.011)	-5.339 (±2.751)	-11.099 (±5.181)
Rastrigin	0.776 (±0.018)	0.799 (±0.017)	-1.553 (±1.813)	-2.828 (±2.186)
Hole-in-Mountain	0.929 (±0.011)	0.410 (±0.047)	-0.684 (±2.136)	-4.588 (±11.253)

Considering the results in Table 5.7, the following can be observed:

- The FCI values are very similar to the previous values (shown in Table 5.4). This indicates that the FCI values are not significantly affected if particles are allowed to leave the search space or not.

- In contrast, the NSC values are very different from previous values. For example, for the Straight function, the mean NSC$_{soc}$ value decreased from -0.235 to -4.874. To explain this, fitness clouds of sample runs on the Straight function are shown in Figure 5.9. Figure 5.9(a) shows that there are a number of points that are fairly high above the diagonal. This would cause the centroids of some segments to be higher than neighbouring segments, resulting in negative slopes. This is more evident in Figure 5.9(b) with many fitness cloud points with neighbouring fitness 1 (due to the high velocities in social PSO updates, many points end up at the boundary minimum point).

It is clear from these results that the NSC measure is not a meaningful measure of



(a) Cognitive PSO (NSC$_{cog}$ = -2.741)

(b) Social PSO (NSC$_{soc}$ = -4.712)

Figure 5.9: Fitness clouds from sample runs on the Straight function for different PSO update strategies, where particles are kept in bounds. Both clouds result in FCI values of 1, but negative values for NSC.

searchability for fitness clouds generated by PSO updates, even when the number of points in the fitness cloud is kept constant. For this reason the measure is not used further in this study.

5.5 Linking to PSO performance on higher dimensional problems

Results in the previous section show that the FDC$_s$, IL$_{ns}$ and FCI measures are meaningful indicators of different aspects of searchability for the simple one-dimensional functions given in Table 5.3. This section evaluates the measures on higher dimensional benchmark problems. The performance of a standard PSO algorithm on the same benchmarks is evaluated and the link between the searchability measures and actual algorithm performance is investigated.

5.5.1 Benchmark problems and expected results

A selection of the benchmark functions defined in Appendix A were used to test the fitness landscape measures proposed in this chapter. Ackley, Griewank, Quadric, Rana, Rastrigin, Rosenbrock, Salomon, Schwefel 2.26, Spherical and Step were used for dimensions 1 (not applicable for Rana and Rosenbrock), 2, 5, 15 and 30. These functions cover a range of characteristics. All functions are multimodal, except for Spherical, Quadric, and Rosenbrock for dimensions 1 to 3. (Note that although the Rosenbrock function is widely stated as unimodal, it has been shown to be multimodal for dimensions of 4 and higher [140].) Although Quadric (also known as Schwefel 1.2) is unimodal and is equivalent to Spherical in 1 dimension, it has been shown to have a weak fitness distance correlation (to the known optimum) in higher dimensions [104]. Functions Griewank and Step are rugged, but the underlying shapes match the Spherical function, so the fitness distance values should be similar to Spherical in higher dimensions. Likewise, the underlying shape of Salomon is the same as the Absolute Value function, so should also give relatively high fitness distance values. Rana and Schwefel 2.26 are multi-funnelled and so should give lower fitness distance values in higher dimensions. The IL_{ns} measure quantifies the difference between the information landscape of the problem and the information landscape of Spherical. Any problem that shares the same basic underlying shape as Spherical should therefore have a low IL_{ns} measure. All functions that have high (or low) FDC_s values, should therefore have low (or high) IL_{ns} values. For the FCI values, it is difficult to predict the effect of just two PSO updates on fitness values, particularly in higher dimensions.

5.5.2 Experimental setup

For the FDC_s calculations, uniform random samples of $500 \times D$ were used. This involves $500 \times D$ fitness calculations and $500 \times D$ Euclidean distance calculations. The IL_{ns} calculations were based on samples of 5000 points in all dimensions. The size of the sample was chosen to be constant due to the polynomially increasing memory requirements and was set as the same sample size as FDC_s in 10 dimensions. For the FCI measures, samples of size 500 were used. For the PSO updates, the inertia weight (w), cognitive acceleration (c_1) and social acceleration (c_2) were set to the values of 0.7298, 1.496, and 1.496, respectively and particles leaving the search space were repaired by setting them at the boundary (Algorithm 5.4).

For the performance metrics, each of the problem instances (function and dimension combinations) was solved using the traditional gbest PSO algorithm (described in Section 3.2). The values for QMetric, SSpeed and SRate (described in Section 3.3) were determined based on 30 independent runs of each algorithm on each problem instance. The parameter values used for the PSO algorithm were as follows: 50 particles, 0.7298 inertia weight (w) and 1.496 for the cognitive and social acceleration constants (c_1 and c_2).

5.5.3 Searchability measures results

The results are summarised in Table 5.8. For each problem / dimension combination, the four searchability measures, FDC$_s$, IL$_{ns}$, FCI$_{cog}$, and FCI$_{soc}$, are reported as mean values based on 30 runs of the algorithm and the standard deviations are given in parentheses.

The values of FDC$_s$ in Table 5.8 range from values close to 1 for Spherical in low dimensions, to as low as 0.005 for Rana in 30 dimensions. For most functions the value of FDC$_s$ decreases as the dimension increases. For example, Spherical reduces from 0.968 in $1D$ to 0.566 in $30D$. The fitness distance correlation coefficient of the Spherical function based on the true optimum should stay close to 1 for any dimension [104]. Estimating the optimum can, however, lead to lower FDC$_s$ values, if the estimated optimum is not close to the true optimum. As a simplified example, consider the Spherical function in $1D$ with an inadequate sample of three points: $x_0 = -2$, $x_1 = 1$, $x_2 = 2$, with associated fitness values $f_0 = 4$, $f_1 = 1$, $f_2 = 4$. The estimated minimum of this sample is therefore at x_1 and the associated distance values to x_1 are $d_0 = 3$, $d_1 = 0$, and $d_2 = 1$. The resulting FDC$_s$ value is 0.756, which is not a reflection of the perfect bowl shape of the Spherical function. In a similar way, a sample of 500×30 points in 30 dimensions is an inadequate sample and results in FDC$_s$ values that are lower than the true FDC values.

Table 5.8: Searchability landscape measures alongside PSO performance metrics for benchmark functions f defined in Appendix A, Table A.1 in different dimensions (D). All searchability measures are means over 30 independent runs and standard deviations are shown in parentheses.

f	D	Searchability measures				Performance metrics		
		FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	QMetric	SRate	SSpeed
f_{ack}	1	0.793 (±0.008)	0.152 (±0.003)	0.804 (±0.018)	0.876 (±0.012)	1.000	1.000	0.471
f_{ack}	2	0.774 (±0.010)	0.191 (±0.004)	0.770 (±0.014)	0.907 (±0.015)	1.000	1.000	0.634
f_{ack}	5	0.701 (±0.025)	0.243 (±0.012)	0.737 (±0.018)	0.935 (±0.012)	1.000	1.000	0.782
f_{ack}	15	0.506 (±0.025)	0.336 (±0.012)	0.727 (±0.019)	0.899 (±0.029)	0.933	0.933	0.855
f_{ack}	30	0.431 (±0.018)	0.367 (±0.009)	0.716 (±0.020)	0.853 (±0.044)	0.300	0.300	0.862
f_{grw}	1	0.967 (±0.002)	0.017 (±0.003)	0.929 (±0.011)	0.862 (±0.015)	1.000	1.000	0.721
f_{grw}	2	0.966 (±0.007)	0.037 (±0.015)	0.904 (±0.014)	0.906 (±0.014)	0.915	0.767	0.669
f_{grw}	5	0.903 (±0.034)	0.118 (±0.024)	0.843 (±0.015)	0.935 (±0.022)	0.560	0.067	0.211
f_{grw}	15	0.653 (±0.024)	0.276 (±0.018)	0.769 (±0.019)	0.840 (±0.062)	0.691	0.100	0.910
f_{grw}	30	0.567 (±0.027)	0.313 (±0.007)	0.703 (±0.018)	0.705 (±0.113)	0.902	0.367	0.921
f_{qdr}	1	0.968 (±0.002)	0.000 (±0.000)	0.963 (±0.008)	0.875 (±0.015)	1.000	1.000	0.942
f_{qdr}	2	0.648 (±0.012)	0.227 (±0.004)	0.906 (±0.012)	0.857 (±0.013)	1.000	1.000	0.893
f_{qdr}	5	0.342 (±0.030)	0.376 (±0.008)	0.876 (±0.013)	0.828 (±0.018)	1.000	1.000	0.906
f_{qdr}	15	0.115 (±0.013)	0.460 (±0.006)	0.861 (±0.015)	0.806 (±0.031)	1.000	1.000	0.869
f_{qdr}	30	0.071 (±0.010)	0.476 (±0.005)	0.859 (±0.016)	0.802 (±0.019)	1.000	1.000	0.711
f_{ran}	2	0.017 (±0.063)	0.484 (±0.017)	0.727 (±0.018)	0.610 (±0.278)	0.187	0.000	0.000

Continued on Next Page...

Table 5.8 – Continued

f	D	Searchability measures				Performance metrics		
		FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	QMetric	SRate	SSpeed
f_{ran}	5	0.012 (±0.035)	0.500 (±0.011)	0.701 (±0.022)	0.665 (±0.157)	0.000	0.000	0.000
f_{ran}	15	0.008 (±0.019)	0.499 (±0.007)	0.696 (±0.021)	0.645 (±0.102)	0.000	0.000	0.000
f_{ran}	30	0.005 (±0.011)	0.501 (±0.007)	0.698 (±0.017)	0.645 (±0.119)	0.000	0.000	0.000
f_{ras}	1	0.708 (±0.015)	0.252 (±0.003)	0.775 (±0.014)	0.795 (±0.016)	1.000	1.000	0.814
f_{ras}	2	0.641 (±0.062)	0.259 (±0.016)	0.743 (±0.020)	0.827 (±0.022)	1.000	1.000	0.792
f_{ras}	5	0.499 (±0.077)	0.317 (±0.024)	0.731 (±0.023)	0.807 (±0.060)	0.533	0.533	0.731
f_{ras}	15	0.393 (±0.042)	0.373 (±0.010)	0.714 (±0.020)	0.766 (±0.073)	0.000	0.000	0.000
f_{ras}	30	0.358 (±0.023)	0.381 (±0.009)	0.711 (±0.020)	0.675 (±0.107)	0.000	0.000	0.000
f_{ros}	2	0.546 (±0.022)	0.287 (±0.004)	0.894 (±0.012)	0.674 (±0.027)	1.000	1.000	0.843
f_{ros}	5	0.687 (±0.063)	0.249 (±0.023)	0.846 (±0.016)	0.806 (±0.055)	0.931	0.067	0.321
f_{ros}	15	0.555 (±0.081)	0.301 (±0.035)	0.756 (±0.018)	0.757 (±0.056)	0.888	0.000	0.000
f_{ros}	30	0.477 (±0.074)	0.335 (±0.022)	0.668 (±0.017)	0.615 (±0.101)	0.482	0.000	0.000
f_{sal}	1	0.971 (±0.002)	0.076 (±0.000)	0.824 (±0.017)	0.870 (±0.014)	1.000	1.000	0.574
f_{sal}	2	0.960 (±0.009)	0.083 (±0.004)	0.799 (±0.015)	0.900 (±0.014)	1.000	1.000	0.592
f_{sal}	5	0.872 (±0.048)	0.162 (±0.024)	0.770 (±0.024)	0.929 (±0.021)	0.000	0.000	0.000
f_{sal}	15	0.627 (±0.037)	0.285 (±0.016)	0.710 (±0.021)	0.840 (±0.061)	0.000	0.000	0.000
f_{sal}	30	0.534 (±0.017)	0.323 (±0.008)	0.655 (±0.022)	0.673 (±0.077)	0.000	0.000	0.000
$f_{sch2.26}$	1	0.317 (±0.032)	0.397 (±0.007)	0.821 (±0.013)	0.520 (±0.022)	1.000	1.000	0.816
$f_{sch2.26}$	2	0.300 (±0.057)	0.405 (±0.005)	0.783 (±0.017)	0.523 (±0.036)	0.967	0.967	0.819
$f_{sch2.26}$	5	0.171 (±0.113)	0.447 (±0.025)	0.771 (±0.018)	0.629 (±0.077)	0.400	0.400	0.826

Continued on Next Page…

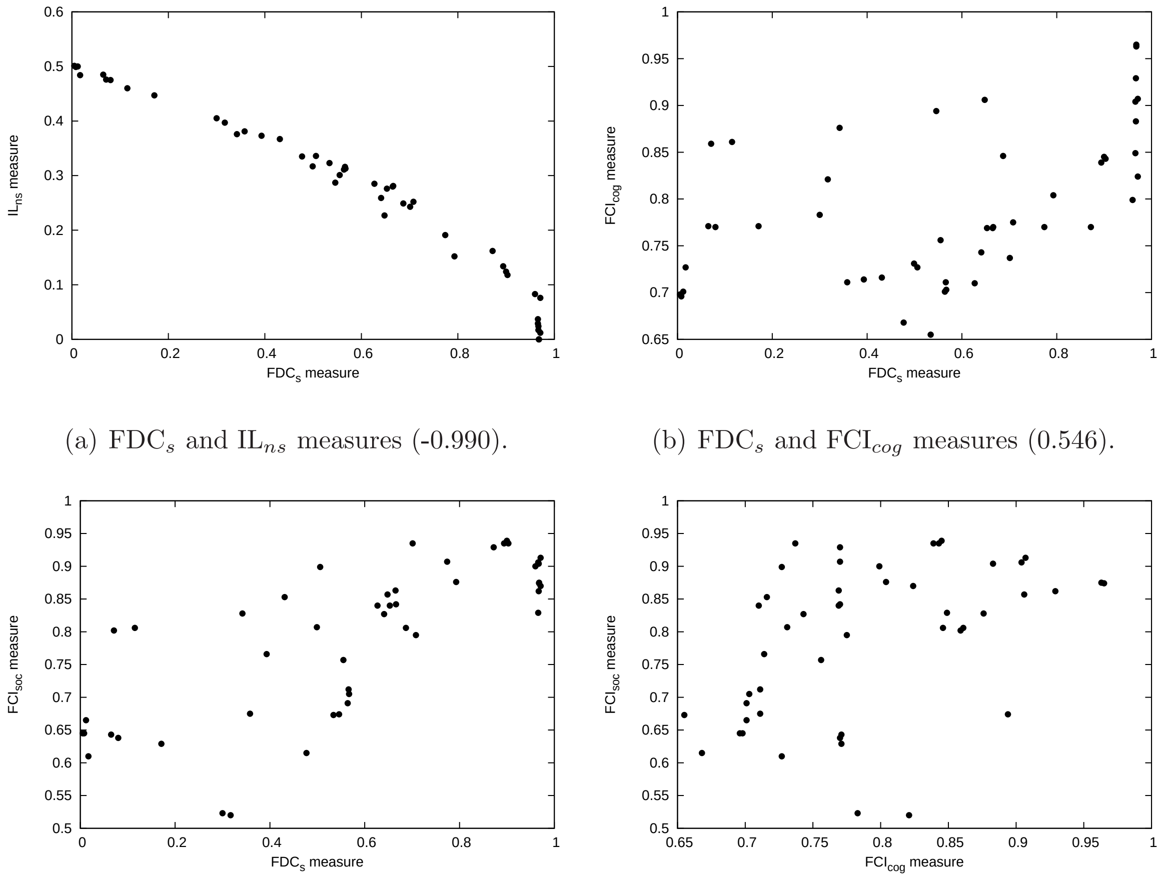
Table 5.8 – Continued

f	D	Searchability measures				Performance metrics		
		FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	QMetric	SRate	SSpeed
$f_{sch2.26}$	15	0.080 (±0.078)	0.475 (±0.017)	0.770 (±0.016)	0.638 (±0.061)	0.000	0.000	0.000
$f_{sch2.26}$	30	0.065 (±0.043)	0.485 (±0.016)	0.771 (±0.018)	0.643 (±0.070)	0.000	0.000	0.000
f_{sph}	1	0.968 (±0.001)	0.000 (±0.000)	0.965 (±0.008)	0.874 (±0.013)	1.000	1.000	0.845
f_{sph}	2	0.971 (±0.003)	0.012 (±0.005)	0.907 (±0.012)	0.913 (±0.013)	1.000	1.000	0.890
f_{sph}	5	0.900 (±0.031)	0.124 (±0.023)	0.845 (±0.015)	0.939 (±0.027)	1.000	1.000	0.916
f_{sph}	15	0.666 (±0.024)	0.281 (±0.009)	0.770 (±0.019)	0.842 (±0.053)	1.000	1.000	0.941
f_{sph}	30	0.566 (±0.016)	0.316 (±0.008)	0.711 (±0.024)	0.712 (±0.066)	1.000	1.000	0.940
f_{stp}	1	0.966 (±0.002)	0.029 (±0.004)	0.849 (±0.015)	0.829 (±0.019)	1.000	1.000	0.994
f_{stp}	2	0.967 (±0.005)	0.024 (±0.006)	0.883 (±0.014)	0.904 (±0.017)	1.000	1.000	0.988
f_{stp}	5	0.894 (±0.038)	0.134 (±0.019)	0.839 (±0.018)	0.935 (±0.024)	1.000	1.000	0.977
f_{stp}	15	0.665 (±0.045)	0.280 (±0.013)	0.769 (±0.017)	0.863 (±0.060)	1.000	1.000	0.972
f_{stp}	30	0.564 (±0.014)	0.311 (±0.010)	0.701 (±0.019)	0.691 (±0.093)	0.924	0.900	0.857

The question then is whether there is value in the FDC$_s$ measure if the value is far from the true FDC value. Investigating the values in Table 5.8, it would seem that the relative FDC$_s$ values within each dimension are consistent. For example, the $1D$ functions with high FDC$_s$ values (> 0.9) are Griewank, Quadric, Salomon, Spherical and Step. In $30D$, the group of functions with the highest FDC$_s$ values (> 0.5) are Griewank, Salomon, Spherical and Step. The only function that is in the first group and not in the second group is Quadric, which is equivalent to Spherical in $1D$, but has been shown to have a weak FDC in higher dimensions [104]. The functions with the lowest FDC$_s$ values in $30D$ are Schwefel 2.26 (0.065) and Rana (0.005), which are both multi-funnelled landscapes, so are expected to have the lowest values. It would seem, therefore, that there is value in the relative FDC$_s$ values in the different dimension groups.

In lieu of discussing the IL$_{ns}$ values in Table 5.8, Figure 5.10(a) shows a scatterplot of the FDC$_s$ values against the IL$_{ns}$ values. It is clear that there is a very strong negative correlation between the FDC$_s$ and IL$_{ns}$ values (Spearman's correlation coefficient of -0.990). This shows that although the two measures use very different approaches, they are capturing essentially the same information on the fitness landscape: high fitness distance correlation seems to imply an information landscape that is similar to the Spherical landscape. Alternatively, an information landscape that is very different from the Spherical landscape seems to imply low fitness distance correlation.

Figure 5.10(c) shows that there is also a strong correlation between the FDC$_s$ and FCI$_{soc}$ measures (Spearman's coefficient of 0.799), but that this is not as strong as with the IL$_{ns}$ measure. This means that there is a link between these measures: when the fitness and distance are highly correlated, PSO social updates will probably result in fitness improvement. Figures 5.10(b) and 5.10(d) show that there is only a moderate correlation between FDC$_s$ and FCI$_{cog}$ (Spearman's coefficient of 0.546) and between FCI$_{cog}$ and FCI$_{soc}$ (Spearman's coefficient of 0.454), reflecting that there is some limited overlap in the information captured by these measures, but that each measure captures something different.



(a) FDC$_s$ and IL$_{ns}$ measures (-0.990).

(b) FDC$_s$ and FCI$_{cog}$ measures (0.546).

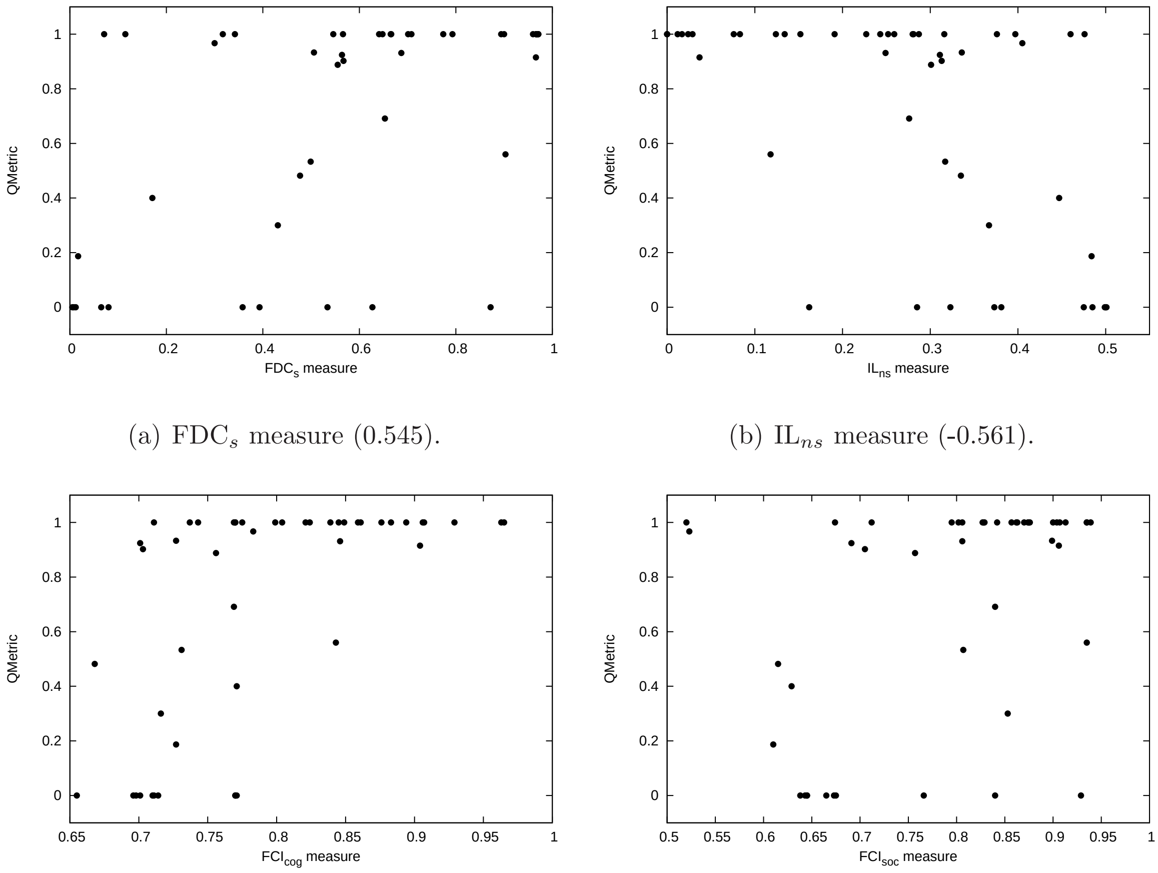
(c) FDC$_s$ and FCI$_{soc}$ measures (0.799).

(d) FCI$_{cog}$ and FCI$_{soc}$ measures (0.454).

Figure 5.10: Scatter diagrams showing the correlation between the different searchability measures. Spearman's correlation coefficient values are given in parentheses in the sub-captions.

5.5.4 PSO performance and searchability measures

The last three columns of Table 5.8 give the performance metrics, based on 30 runs of the PSO algorithm. For a feature metric to be useful it should show some correlation (or negative correlation) to performance. However, as discussed in Section 2.1, no one technique can serve as a predictor of hardness. Figures 5.11(a) to 5.11(d) show the scatter plots of the four searchability measures against the QMetric performance measure. All measures show a moderate correlation or negative correlation to QMetric. The measure that provides the strongest correlation is the FCI$_{cog}$ measure (Spearman's correlation



(a) FDC$_s$ measure (0.545).

(b) IL$_{ns}$ measure (-0.561).

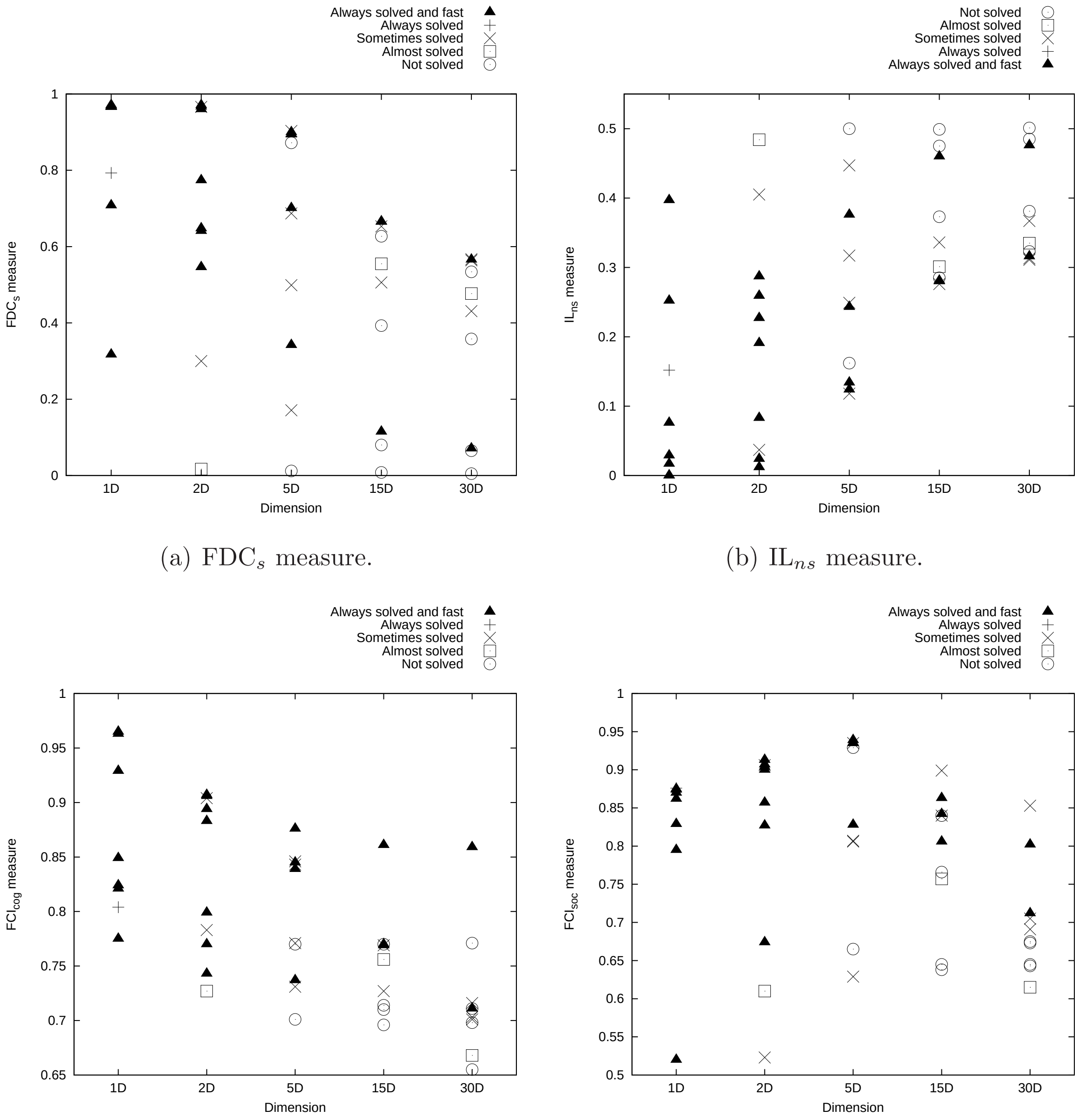
(c) FCI$_{cog}$ measure (0.662).

(d) FCI$_{soc}$ measure (0.466).

Figure 5.11: Scatter diagrams showing the correlation between the performance of a traditional PSO algorithm (as quantified by QMetric) and different searchability measures. Spearman's correlation coefficient values are given in parentheses in the sub-captions.

coefficient of 0.662). The scatterplots in Figure 5.11 have a large proportion of values at the top and at the bottom with a few points scattered in between. This is indicative of distinct groups of problems based on success or failure of the algorithm in solving the problem. QMetric on its own only captures part of the picture of performance, not considering, for example, how quickly a solution is found. An alternative approach to plotting a single performance measure is to allocate each problem instance solved by the algorithm into a performance class using a combination of QMetric, SRate and

SSpeed values (as described in Section 3.3.4). Figure 5.12 plots these classes against the searchability measures with each instance grouped according to dimension.



(a) FDC$_s$ measure.

(b) IL$_{ns}$ measure.

(c) FCI$_{cog}$ measure.

(d) FCI$_{soc}$ measure.

Figure 5.12: Performance of a traditional PSO algorithm on benchmark problems plotted against different searchability measures.

In Figure 5.12, if a given searchability measure is a good predictor of PSO performance, then the order of symbols in a dimension column should match the order of symbols in the legend. Note that for the IL_{ns} measure in Figure 5.12(b), the symbols in the legend are displayed in reverse order, because the measure is a negative searchability measure. In the one-dimensional case, all problems were solved in all cases and all except one were solved fast. The problem that took longer than the others to solve on average is the Ackley function. This could be because of the high level of ruggedness of that function. For the two-dimensional case there is one function that was almost solved (Rana) and two functions that were sometimes solved (Griewank and Schwefel 2.26). The measures that provide the best predictive value for the $2D$ case are the FDC_s and IL_{ns} measures. The order of all symbols in the plot, except for one cross (Griewank), match the legend. For the $5D$ problems, there are two problems that are not solved (Rana and Salomon), indicated by the two circles. For the Rana function, all the searchability measures are mostly in line with this failure (the highest circle in Figure 5.12(b) and the lowest circle in 5.12(a) and 5.12(c)). However, for the Salomon function, the searchability measures are not indicative of the failure. The steepness of the gradients for the Salomon function could be an alternative indicator of the failure in this case. For the higher dimensional problems (15 and 30D), all searchability measures provide some value as predictors of algorithm failure, although none predict all cases correctly. These examples illustrate that the four searchability measures provide some insight into the difficulty of problems for PSO algorithms, but that they do not provide the full picture of what makes a problem hard for a PSO.

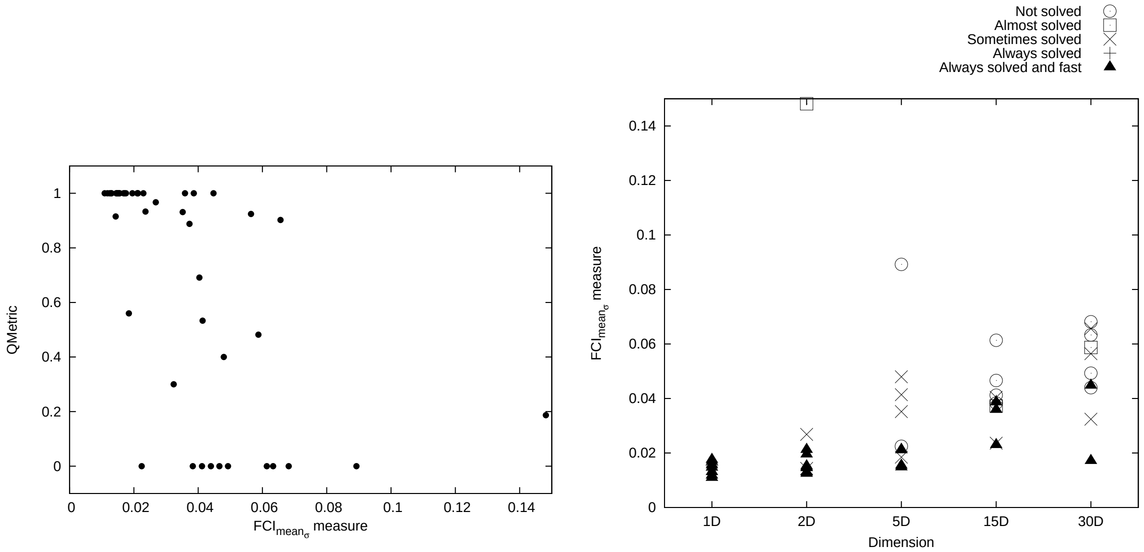
Unpredictability of FCI measures

Studying the data in Table 5.8 it can be seen that in some cases although the actual FCI value does not correlate with performance, the standard deviation of the FCI measures seems to capture some aspect of the difficulty of the problem. For example, considering Rastrigin in 30 dimensions and, just below it in the table, Rosenbrock in 2 dimensions: the algorithm fails completely on Rastrigin $30D$, but very successfully solves Rosenbrock 2D. These two functions have very similar FCI_{soc} values (0.675 and 0.674), but the standard deviation of the FCI_{soc} for Rastrigin 30D is almost four times the value of the standard deviation of the FCI$_{soc}$ for Rosenbrock 2D. Similarly, in the case of the Salomon function in 2 and 5 dimensions, the the FCI$_{cog}$ and FCI$_{soc}$ values are very similar, but the standard deviations are much larger for the 5D function, where the algorithm failed. It would seem that the unpredictability of the FCI measures could be a part-indicator of the difficulty of the problem. In other words, if multiple runs of two iterations of a PSO algorithm result in highly variable searchability profiles, then this could be indicative of a harder problem to solve for a PSO. A simple way of combining the standard deviations of the FCI values of both update strategies is to take the mean. A proposed additional measure is therefore the mean of the standard deviation of both FCI measures and is referred to as the fitness cloud mean standard deviation or FCI$_{\bar{\sigma}}$. The measure is summarised in Table 5.9. The parameters are the same as for FCI$_{cog}$ and FCI$_{soc}$, but the range of the result is different. Given a sample of points in the range [0,1] (as for the FCI measures), the minimum standard deviation is 0 and the maximum standard deviation is approximately 0.509. In terms of computational efficiency, the FCI$_{\bar{\sigma}}$ measure requires a big enough sample in order to calculate the mean and standard deviation and this value is set at 30.

Figure 5.13 shows the relationship between the proposed FCI$_{\bar{\sigma}}$ measure and PSO performance. The measure shows a strong negative correlation to QMetric (Spearman's correlation coefficient of -0.705). From Figure 5.13(b), the symbols in each dimension column matches the legend for most cases, indicating that instances of the same dimension with higher FCI$_{\bar{\sigma}}$ values in most cases perform worse than instances with lower FCI$_{\bar{\sigma}}$.

Table 5.9: Additional proposed measure of searchability based on fitness clouds

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
FCI$_{\bar{\sigma}}$	Fitness cloud index mean standard deviation.	(1) size of sample, n, (2) inertia weight, w, (3) acceleration constants, c_1 and c_2.	$[0, 0.509]$: indicating the level of deviation from the FCI mean for both update strategies.	30 × (3n fitness evaluations, 2n solution updates).



(a) FCI$_{\overline{\sigma}}$ against QMetric.

(b) Discretised performance per dimension

Figure 5.13: Link between the performance of a traditional PSO algorithm and the FCI$_{\overline{\sigma}}$ measure on benchmark problems. The Spearman's correlation coefficient between FCI$_{\overline{\sigma}}$ and QMetric is -0.705, indicating a strong negative correlation.

5.5.5 Discussion

The proposed FDC$_s$ and IL$_{ns}$ measures can be used to quantify the general searchability of an unknown optimisation problem. Results on a set of benchmark problems show that there is a very strong negative correlation between these measures, which indicates that the measures capture similar information. Both measures are based on initial random samples, require a single parameter (the size of the sample) and have linear time efficiency with respect to the size of the sample. The memory requirement of the IL$_{ns}$ measure, however, is polynomial with respect to the sample size, so it may not be a suitable measure for higher dimensional problems that require large sample sizes. For the benchmark problems considered, the FDC$_s$ and IL$_{ns}$ values were moderately correlated and negatively correlated with performance of a traditional gbest PSO algorithm (as measured by QMetric).

The FCI$_{cog}$ and FCI$_{soc}$ measures quantify the searchability of a problem with respect to cognitive and social updates in the initial stages of PSO search. For the benchmark problems considered, both FCI measures were moderately correlated with performance of a traditional gbest PSO algorithm (as measured by QMetric), with FCI$_{cog}$ showing a stronger correlation than FCI$_{soc}$. The measure FCI$_{\overline{\sigma}}$ quantifies the unpredictability of the FCI$_{cog}$ and FCI$_{soc}$ measures and showed a strong negative correlation to PSO performance.

It is a premise of this study that no single problem feature on its own can serve as a predictor of problem difficulty. Instead, a range of different features need to be considered together to attempt to predict algorithm performance on an unseen problem. In this scenario, the searchability measures proposed above all show potential value as part-predictors of PSO performance if used with other measures for features such as ruggedness, presence of funnels and gradients.

5.6 Summary

This chapter investigated a number of possible measures of searchability for continuous landscapes. Two general measures of searchability are proposed as adaptations of measures previously quantifying deception for local search (fitness distance correlation and information landscape hardness measure). In addition, a number of measures derived from fitness clouds based on PSO updates are proposed. All measures were evaluated on simple one dimensional functions to see if results were consistent with a visual inspection of the functions. Results of the negative slope coefficient based on PSO updates were not consistent with expected results and this measure was therefore abandoned. The remaining searchability measures were tested on higher dimensional problems and all measures showed some relation to the performance of a traditional gbest PSO algorithm. The proposed fitness landscape measures from this chapter and from Chapter 4 are summarised in Appendix B. The following chapter combines all the proposed fitness landscape metrics with performance of different PSO algorithms and investigates the possibility of predicting PSO performance based on these metrics.

Chapter 6

Predicting PSO Performance

This chapter investigates the link between problem characteristics, as measured by the feature metrics proposed in Chapters 4 and 5, and PSO algorithm performance. The chapter starts by introducing the topic and summarising recent related work on the general topic of linking optimisation problem characteristics to algorithm performance. Section 6.2 then describes how the training and testing datasets were generated to serve as the basis for the data mining exercise that follows. Section 6.3 presents the results of predicting failure for the different PSO variations and Section 6.4 presents the results of attempting to predict PSO performance.

6.1 Introduction

Predicting population-based stochastic algorithm behaviour on unknown problems is a difficult task. Some algorithms even behave in an unpredictable way on known problems. For example, in one early study of GAs, Mitchell *et al.* [98] attempted to understand the GA by developing a class of simplistic functions designed to match GA behaviour. These functions, called 'Royal Road' functions were based on the GA notion of building blocks [47] and could be varied in different ways to tune the difficulty of the problem for a GA. Despite the Royal Road functions being tuned into landscapes on which GAs should perform well, experiments in a later study showed that a hill-climbing algorithm significantly outperformed the GA on these functions [38]. Based on these unexpected results and further investigation into the behaviour of the GA, a new Royal Road function was designed on which an 'idealised' GA performed as expected [99]. This extended study over a number of years by Forrest, Holland and Mitchell [38, 98, 99] highlights the difficulty in understanding evolutionary and other population-based stochastic optimisation algorithms and the problems that they are trying to solve. If a carefully designed simplistic problem tailor-made to be solved by a GA results in unpredictable behaviour when solved by a GA, then it is surely even harder to predict behaviour of a given algorithm on a complex unknown problem?

Early work on the link between problem characteristics and algorithm performance was not successful. In 1997, Eiben and Bäck [33] attempted to find a link between the performance of an evolutionary algorithm (with differing recombination strategies) and the characteristics of functions. The function characteristics they considered were modality, separability, and (ir)regularity of the arrangement of local optima. Experimental results showed no relation between these characteristics and the performance of the algorithm. They concluded that *"This is actually part of a bigger problem. At the moment the evolutionary computation community does not have the appropriate ways to describe and distinguish objective functions, or more generally problem types . . . it seems that the presently maintained vocabulary does not capture those aspects of test functions that are significant for the behavior of evolutionary algorithms."*[33].

Recent work in the field is showing more promising results. Some contributions towards establishing links between problem characteristics and algorithm behaviour include:

- Smith-Miles [143] investigated the link between problem characteristics and algorithm performance for the quadratic assignment problem (QAP). Neural networks were successfully trained to predict the performance of three metaheuristics (tabu search, iterated local search and min-max ant system) using QAP specific features and fitness-distance metrics based on local search runs.

- Bischl *et al.* [12] used low-level problem features based on exploratory landscape analysis (proposed by Mersmann *et al.* [93]) of the BBOB test suite [52] and then successfully used one-sided support vector regression to predict which algorithms (from a portfolio of algorithms) would perform well.

- Muñoz *et al.* [101] developed a model using a neural network to predict performance (measured in terms of function evaluations to find a solution) of a CMA-ES algorithm [54]. The inputs to the model were a number of landscape features as well as algorithm parameters. In this way the model could be used to select the best predicted algorithm configuration of the problem.

The studies mentioned above have shown that links can be found between problem characteristics and algorithm performance and that problem characteristics can be used to predict or select the most appropriate algorithm from a set of possible algorithms. The prediction models used are, however, black boxes that do not help in understanding the link between problems and algorithms. A distinguishing feature of this study is that the aim is not only to predict algorithm failure or performance, but also to understand the algorithms better. By using decision tree induction, the kinds of problems that the algorithms struggle with are highlighted and the resulting models therefore provide insights into the algorithms themselves. Although this study is restricted to PSO algorithms, the same techniques can be applied to studying other optimisation algorithms.

Another distinguishing feature of this study is that the techniques used to characterise problems are based on general fitness landscape concepts, such as ruggedness and searchability and are therefore widely applicable to any real-valued optimisation problems. In addition, because the techniques are based on general landscape features (rather than low-level statistical features), most of the measures can be described in language that relates back to fitness landscapes. For example, the $\text{FEM}_{0.01}$ measure can be referred to as micro ruggedness. This means that the learning models that are generated can be re-expressed in fuzzy language that can be reasoned about in fitness landscape terms. This is demonstrated in Section 6.3.2.

6.2 Dataset generation

The 24 benchmark functions defined and illustrated in Appendix A were used as the basis for building a dataset for investigating the link between fitness landscape features and PSO performance. The functions were used as follows:

- Functions Ackley, Alpine, Griewank, Quadric, Quartic, Rastrigin, Salomon, Schwefel 2.22, Schwefel 2.26, Skew Rastrigin, Spherical, Step and Weierstrass were used to define six problem instances for 1, 2, 5, 10, 15 and 30 dimensions for each of the functions.
- Functions Bohachevsky, Levy, Pathological, Rana, Rosenbrock and Zakharov (not defined for 1 dimension) were used in 2, 5, 10, 15 and 30 dimensions.
- Function Michalewicz was used to define instances for 2, 5, 10 and 30 dimensions (the optimum value for other dimensions is not reported in literature).
- Functions Beale, Egg Holder, Goldstein-Price and Six-hump camel-back were used for only 2 dimensions.

This resulted in a total of 116 problem instances.

The fitness landscape features were determined for each problem instance using the ten proposed fitness landscape measures summarised in Appendix B, Table B. The following parameter values were used:

- For the FEM$_{0.01}$ and FEM$_{0.1}$ measures, D walks were performed each with 1000 steps. The total number of steps on all walks was therefore equal to $1000 \times D$.
- For the DM measure, the total size of the sample was set at 1000 points, with the sub-sample of best solutions set at 10% (100 points).
- For the G$_{avg}$ and G$_{dev}$ measures, D walks were performed each with 1000 steps. The total number of steps on all walks was therefore equal to $1000 \times D$. The step size was set at $\frac{(x^{max}-x^{min})*D}{1000}$, where x^{max} and x^{min} define the bounds of the search space. The G_{avg} and G_{dev} measures were calculated across all walks.
- For the FDC$_s$ measure, the size of the sample was set at $500 \times D$ and for the IL$_{ns}$ measure, the size of the sample was set at 5000.
- For the FCI$_{cog}$, FCI$_{soc}$ and FCI$_{\overline{\sigma}}$ measures, the size of the sample was set at 500. For the PSO updates, the inertia weight (w), cognitive acceleration (c_1) and social acceleration (c_2) were set to the values of 0.7298, 1.496, and 1.496, respectively.

For each fitness landscape measure, 30 independent runs of the relevant algorithm was performed and the mean value was calculated. The $\mathrm{FCI}_{\overline{\sigma}}$ measure is an exception, which is the mean of the standard deviations of the 30 runs of the FCI_{cog} and FCI_{soc} measures. In this way, each problem instance was characterised by the dimension and ten fitness landscape measures in the form of the following 11-valued characteristic vector:

$$(D, \mathrm{FEM}_{0.01}, \mathrm{FEM}_{0.1}, \mathrm{DM}, \mathrm{G}_{avg}, \mathrm{G}_{dev}, \mathrm{FDC}_s, \mathrm{IL}_{ns}, \mathrm{FCI}_{cog}, \mathrm{FCI}_{soc}, \mathrm{FCI}_{\overline{\sigma}})$$

To determine the actual difficulty of the problems, each of the 116 problem instances was solved using the seven PSO variations described in Section 3.2. The values for QMetric, SSpeed and SRate (described in Section 3.3) were determined based on 30 independent runs of each algorithm on each problem instance. For all of the PSO variations 50 particles were used and the inertia weight and acceleration constants were the same as for the FCI measures. For the local best PSO, a neighbourhood size of 2 was used. The performance of each algorithm on each problem instance was discretised into the five performance classes as described in Section 3.3.4, namely:

- S++: always solved and fast,
- S+: always solved,
- S: sometimes solved,
- S–: almost solved,
- F: not solved.

The full dataset was then divided into a training set consisting of 2/3 of the patterns (77 patterns) and a testing set of 1/3 of the patterns (39 patterns). To ensure a similar distribution of performance classes in the training and testing set, the dataset was divided as follows: the full dataset was sorted by all performance classes and every third pattern from the sorted list was selected for the testing dataset. The resulting training and testing datasets are listed in Appendix C in Tables C.1 and C.2 respectively (re-sorted by problem and dimension). In this way, the same function can appear in both the training and the testing set, but in different dimensions. For example, Ackley appears in dimensions 2, 5, 10 and 30 in the training set and in dimensions 1 and 15 in the testing dataset, but Pathological appears only in the training set.

6.3 Predicting algorithm failure

A simplified version of the performance prediction problem is to predict algorithm failure. For the purposes of this study, failure is regarded as the F performance class, meaning that the algorithm was not able to find the solution to within the fixed accuracy level of the problem in any of the runs and was not able to almost find the solution, either (as measured by QMetric). In this simplification of the problem, the classes S++, S+, S and S– are regarded together as a single class, indicating some level of success, denoted as S* (in the regular expression sense, where S* matches any of the success classes).

6.3.1 Decision tree induction

The C4.5 decision tree induction algorithm [122] was used to derive classification models for algorithm failure. Decision trees were built using the WEKA tool [51], in particular the WEKA version 3.6.9 J48 implementation of the C4.5 algorithm. The parameter values used were a confidence threshold of 25% (the default) and a minimum number of instances per leaf node of 5 (default in WEKA is 2). The confidence threshold affects the level of pruning, where smaller values incur more pruning. Increasing the minimum number of instances per leaf node in general has the effect of reducing the tree size and possibly producing a more general model. The results are shown in Tables 6.1 to 6.2.

In each case, the entire training set (77 patterns) was used to generate the tree after which the model was tested for accuracy using the testing set (39 patterns). For each tree the resulting training and testing accuracies are reported below the tree. In the visualisations of the tree models, the splitting values of real attributes are rounded off to three decimal places. The total number of instances that reach each leaf node is indicated in parentheses below the node. The number of instances that are incorrectly classified by the node, if any, are indicated after a slash in the parentheses. For example, the tree predicting failure for traditional gbest PSO in Table 6.1 can be interpreted as follows:

- The model predicts that problems with a dimension of 1, 2 or 5 will be of class S*. Thirty-seven of the training patterns fall into this category and all of these are in

Table 6.1: PSO failure prediction models for four PSO variations.

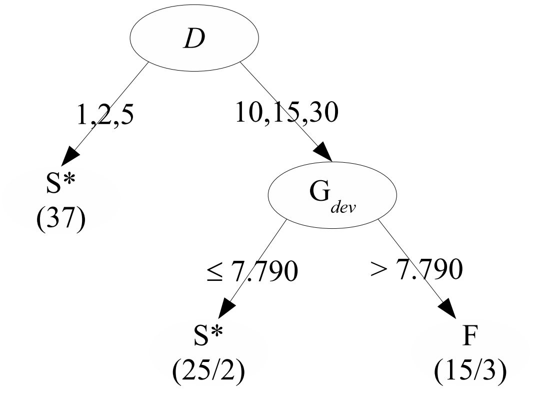
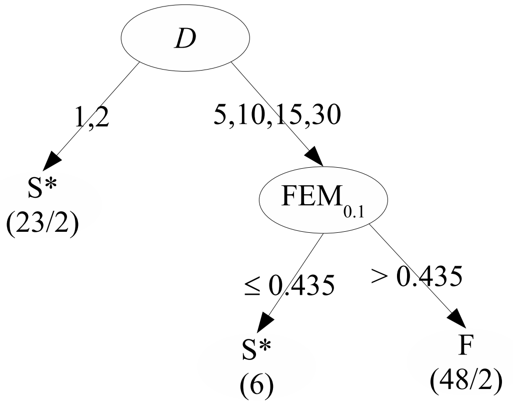
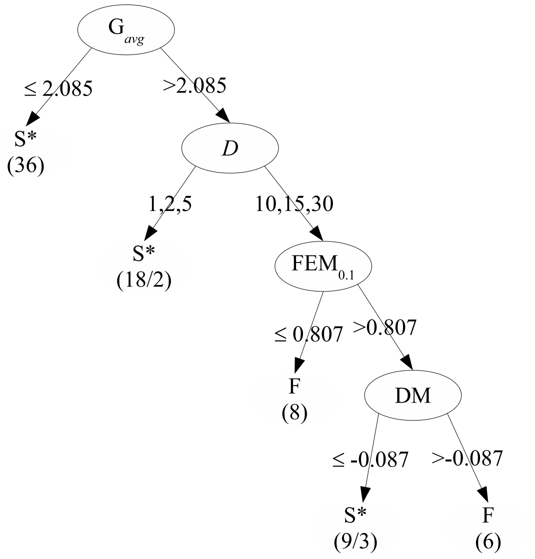
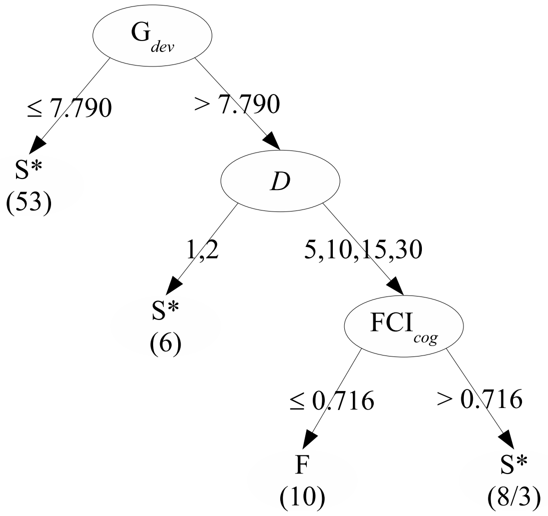
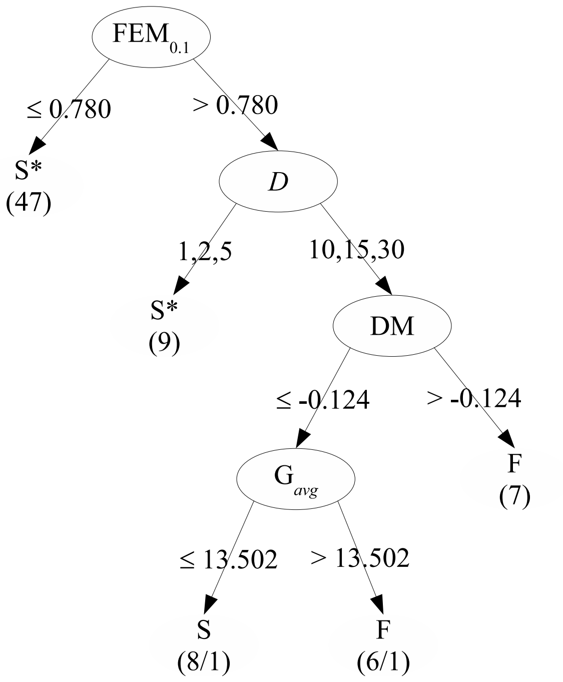
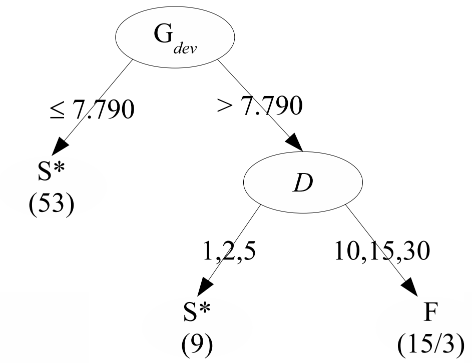
Traditional gbest PSO	**Cognitive PSO**
 D 1,2,5 10,15,30 S* (37) G dev ≤ 7.790 > 7.790 S* (25/2) F (15/3) Training accuracy: 93.5% Testing accuracy: 92.3%	 D 1,2 5,10,15,30 S* (23/2) FEM 0.1 ≤ 0.435 > 0.435 S* (6) F (48/2) Training accuracy: 94.8% Testing accuracy: 92.3%
Social PSO	**Local best PSO**
 G avg ≤ 2.085 >2.085 S* (36) D 1,2,5 10,15,30 S* (18/2) FEM 0.1 ≤ 0.807 >0.807 F (8) DM ≤ -0.087 >-0.087 S* (9/3) F (6) Training accuracy: 93.5% Testing accuracy: 76.9%	 G dev ≤ 7.790 > 7.790 S* (53) D 1,2 5,10,15,30 S* (6) FCI cog ≤ 0.716 > 0.716 F (10) S* (8/3) Training accuracy: 96.1% Testing accuracy: 92.3%

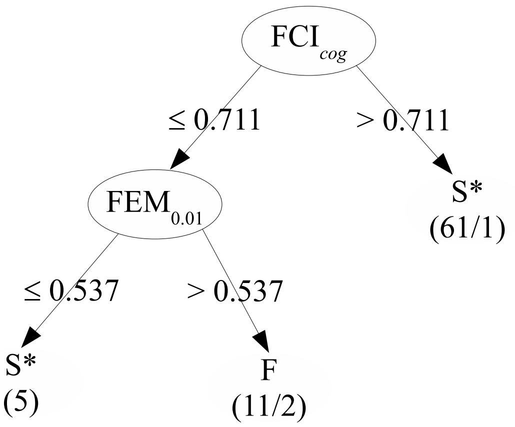
Table 6.2: PSO failure prediction models for asynchronous gbest PSO, barebones PSO and modified barebones PSO.

<table>
<tr><th>Asynchronous gbest PSO</th><th>Barebones PSO</th></tr>
<tr><td>


Training accuracy: 97.4%

Testing accuracy: 92.3%
</td><td>


Training accuracy: 96.1%

Testing accuracy: 87.2%
</td></tr>
<tr><th colspan="2">Modified barebones PSO</th></tr>
<tr><td colspan="2">


Training accuracy: 96.1%

Testing accuracy: 94.9%
</td></tr>
</table>

the class S*.

- The model predicts that problems with a dimension of 10, 15 or 30 and a G_{dev} value $\leq$ 7.790 will be of class S*. Twenty-five of the training patterns fall into this category and two of these are not of the class S* (incorrectly classified by the model).

- The model predicts that problems with a dimension of 10, 15 or 30 and a G_{dev} value $>$ 7.790 will be of class F. Fifteen of the training patterns fall into this category and three of these are not of the class F (incorrectly classified by the model).

Given the total number of training patterns of 77 and a total of five incorrectly classified patterns by this model, this leaves 72 training patterns correctly classified, resulting in a training accuracy of 93.5%. The accuracy of the testing data was 92.3% indicating that 36 out of the 39 testing patterns were correctly classified by the model. Before investigating the particular failure prediction models for each algorithm, a general observation can be made: The training and testing accuracies achieved by the models in Tables 6.1 to 6.2 show that it is possible to predict PSO failure based on fitness landscape features with a fairly high degree of accuracy. In all cases, models were constructed with training accuracy levels above 90%. For most algorithms, the testing accuracy was also above 90%, except for the social PSO and barebones PSO algorithms, with accuracies of 76.9% and 87.9%, respectively. Further work would need to confirm whether these results can be validated with other functions not used in this study. Section 7.2.1 elaborates on this point.

Another general observation is that different fitness landscape metrics feature in the failure prediction tree models of the different algorithms. This supports the idea that a single feature cannot be used to predict problem difficulty. Of particular interest are the attributes appearing higher up in the trees, as these features were chosen by the C4.5 algorithm as contributing the most to the information regarding the final class of S* or F. For example, G_{dev} (deviation from the average gradient) featured as the most significant factor in predicting failure for local best PSO, whereas $FEM_{0.1}$ (macro ruggedness) was the most significant in predicting failure for asynchronous global best PSO. It is worth noting the prominence of the gradient measures (G_{dev} and G_{avg}) in many of the models, indicating that the PSO algorithm is strongly influenced by steep gradients.

When viewing the models, it is important to bear in mind that many of the landscape measures are related. For example, the data in Section 4.7.2 showed that there was a strong correlation between the FEM ruggedness measures and the gradient measures. It is therefore possible that in the absence of one measure (say G_{dev}), another related measure (such as G_{avg} or one of the FEM measures) would instead feature in the decision tree models. This is an area of further work discussed in Section 7.2.

6.3.2 Fuzzy rule extraction

To understand specific decision trees, insight into the data for each feature metric is required. For example, considering the traditional gbest PSO failure prediction model, what does it mean if a G_{dev} value is greater than 7.790? The G_{dev} measure estimates the deviation from the average gradient based on random walks. A large value would therefore indicate very steep gradients in places and shallow gradients elsewhere, differing from the average gradient. Without an understanding of the range of G_{dev} values, it is, however, still not clear what it means if G_{dev} is greater than 7.790, because it is not known whether 7.790 is a high value or not.

Table 6.3 lists summary statistics on the fitness landscape values from the full dataset. These values are helpful for putting meaning to the splitting values in the decision trees. Considering the summary statistics for G_{dev}, it can be seen that the distribution of values is skewed to the left. Half of the instances have G_{dev} values below 3.207, despite an overall range of values from 0 to 82.369. The splitting value in the traditional gbest PSO decision tree, 7.790, is closest to the upper quartile, so a value greater than 7.790 could be regarded as covering the range of mostly high values for G_{dev}. The traditional gbest PSO failure prediction model can therefore be interpreted in fuzzy terms as: failure is predicted when the dimension is high (10,15 or 30) and there are high deviations in the gradient. Using similar reasoning, Table 6.4 provides fuzzy rules for each of the failure prediction decision trees. For each fitness landscape measure, the range of values between the minimum and the lower quartile are regarded as ‘low’, values between the lower quartile and median as ‘fairly low’, values between the median and upper quartile as ‘fairly high’ and between the upper quartile and the maximum as ‘high’.

Table 6.3: Summary statistics of the fitness landscape measure values from the full dataset.

	Minimum	Lower quartile	Median	Upper quartile	Maximum
$\text{FEM}_{0.01}$	0.237	0.390	0.533	0.738	0.891
$\text{FEM}_{0.1}$	0.288	0.596	0.682	0.824	0.881
DM	-0.359	-0.330	-0.287	-0.168	0.119
G_{avg}	0.000	2.000	2.901	11.377	64.260
G_{dev}	0.000	1.365	3.207	8.714	82.369
FDC_{s}	0.002	0.315	0.567	0.766	1.000
IL_{ns}	0.000	0.210	0.307	0.377	0.506
FCI_{cog}	0.614	0.727	0.774	0.854	0.965
FCI_{soc}	0.509	0.713	0.831	0.901	0.993
$\text{FCI}_{\overline{\sigma}}$	0.008	0.017	0.028	0.047	0.148

In some cases, although the trees differ slightly, the resulting failure classification rules are the same. For example, in the case of both the traditional gbest PSO and the barebones PSO models, the rules are logically equivalent.

6.3.3 Discussion

This section discusses the derived rules in Table 6.4. The question that is addressed is whether the rules make sense in terms of the algorithm models.

In the case of the traditional gbest PSO, steep gradients can result in deception, which can lead to algorithm failure. This is was explained in Section 4.1 and illustrated in Figure 4.1, where a landscape with steep gradients has narrower basins of attraction and a global attractor can pull particles away from the global optimum. The fuzzy rule for traditional gbest PSO, however, predicts that failure only occurs when the dimension is high and there are high deviations in the gradient. The reason that steep gradients do not lead to failure in lower dimensions could be because of the size of the search space. In lower dimensions, there is a higher chance of one of the 50 particles of the swarm being initialised in the global optimum basin, or discovering that basin during the search process. In higher dimensions, with the larger search space, this chance is

Table 6.4: Fuzzy rules for predicting algorithm failure derived from each of the decision trees in tables 6.1 and 6.2

PSO algorithm	**The algorithm is predicted to fail . . .**
Traditional gbest PSO	. . . if the dimension is high (10, 15, 30) and there are high deviations in the gradient.
Cognitive PSO	. . . if the dimension is medium to high (5, 10, 15, 30) and the macro ruggedness is not very low.
Social PSO	. . . if the average gradient is not low and the dimension is high (10, 15, 30) and (the macro ruggedness is not high or (the macro ruggedness is high and the dispersion metric is very high)).
Local best PSO	. . . if there are high deviations in the gradient and the dimension is medium to high (5,10,15,30) and FCI$_{cog}$ is low (many cognitive updates result in a deterioration in fitness).
Asynchronous gbest PSO	. . . macro ruggedness is high and the dimension is high (10, 15, 30) and (dispersion metric is high or (the dispersion metric is not high and the average gradient is high)).
Barebones PSO	. . . if there are high deviations in the gradient and the dimension is high (10, 15, 30).
Modified barebones PSO	. . . if FCI$_{cog}$ is low (many cognitive updates result in a deterioration in fitness) and micro ruggedness is fairly high to high.

reduced, assuming that the swarm size remains the same.

The fuzzy rule for cognitive PSO failure is similar to the one for gbest PSO, but instead focuses on macro ruggedness. This makes sense, because cognitive PSO does away with the global attractor and is essentially a population of local search optimisers (hill climbers). Even a low level of ruggedness could result in each particle being trapped in its own local optimum. Recall that cognitive PSO has a previous velocity term, which could help particles to escape small local optima. This could be a reason why macro ruggedness, rather than micro ruggedness, features as the main determiner of failure for cognitive PSO.

For the social PSO failure prediction, the first part of the rule is ‘if the average gradient is not low’. For the same reasons that traditional gbest PSO struggles with steep gradients, it makes sense that social PSO struggles even more with high gradients, because the model does not have the exploration component that comes with the personal best attractor. The remainder of the fuzzy rule, however, cannot be explained logically. Part of the rule states that the algorithm will fail ‘if the macro ruggedness is not high’, which does not make sense. This illogical portion of the tree could be a reason for the low testing accuracy of the model, compared with the other models.

In the case of the local best PSO fuzzy rule, the gradient deviation and dimension feature as the top two factors, as is the case for traditional gbest PSO. The last part of the rule, however, brings in FCI$_{cog}$, which is a measure of the proportion of cognitive updates that improve fitness. Local best PSO is similar to traditional gbest PSO in that it incorporates both a neighbourhood attractor and a local attractor. The difference is that the neighbourhood of lbest PSO is much smaller than for gbest PSO, so it is affected by a smaller number of particles with each update. It therefore makes sense that if many cognitive updates (which are local in nature) result in a deterioration in fitness, that lbest PSO updates could also struggle to improve fitness and therefore progress towards an area of improved fitness.

For the asynchronous gbest PSO the combinations that are predicted to result in failure are either:

- high dimensional functions with high macro ruggedness and a high dispersion metric (indicating the possibility of multiple funnels), or
- high dimensional functions with high macro ruggedness and high average gradients.

This means that for high dimensional problems, high macro ruggedness on its own is not a problem, but when combined with multiple funnels or high average gradients, it is predicted to be a problem for the algorithm. It is not understood why these particular combinations are significant for asynchronous gbest PSO, but it is interesting that the features that are important in other algorithms are combined for this algorithm.

The fuzzy rule for barebones PSO failure is equivalent to the rule for traditional gbest PSO. Both algorithms have a balance between exploration (in the form of the gbest attractor) and exploitation (in the form of the pbest attractor), so it makes sense that they are affected by the same fitness features.

For the modified barebones PSO, the combination of a low FCI_{cog} value and fairly high micro ruggedness is predicted to result in failure. The modified barebones PSO algorithm puts more emphasis on the personal best position in a particle's position update than the barebones PSO algorithm. Both the FCI_{cog} and micro ruggedness can be seen as measures more focussed on the measurement of local neighbourhood, since the FCI_{cog} quantifies the fitness-improving ability of cognitive updates and $\text{FEM}_{0.01}$ measures ruggedness on a small scale. It therefore makes sense that these measures feature in the decision tree for modified barebones PSO.

6.3.4 Summary

The data mining results for predicting PSO failure based on fitness landscape metrics are very promising. Most of the resulting decision trees not only show value as predictors of failure on unknown problems, but also provide insight into the algorithms themselves. These insights are captured in fuzzy rules derived from the decision trees. For example, one general observation is that the traditional gbest PSO and barebones PSO have a high probability of failing on high dimensional problems that have steep gradients. Estimating the gradient deviation metric is a computationally inexpensive task and could therefore serve as a valuable pre-optimisation probe into an unknown problem.

6.4 Predicting algorithm performance

This section investigates the possibility of predicting algorithm performance based on fitness landscape measures using decision tree induction, where performance is in terms of the five performance classes (S++, S+, S, S–, F) described in Section 3.3. As with the failure prediction, the training set was used to generate a decision tree for each algorithm, which was then tested for accuracy using the testing dataset. The default parameters for the J48 algorithm resulted in large trees, so parameters were selected to reduce the size of the trees and the occurrence of overfitting (15% confidence threshold and 5 minimum number of instances per leaf node). The results are shown in Figures 6.5 to 6.11. For each algorithm, the decision tree model is given with the associated training and testing accuracies. The confusion matrix of the testing data is also given, which gives the actual classes in the final column and the predicted classes in the first row. For example, for the traditional gbest PSO performance prediction model in Figure 6.5, the confusion matrix shows that there were 22 instances in the S++ class (19 + 3); 19 of these were correctly predicted as S++ and 3 were incorrectly predicted as being in the S class.

Considering the training and testing accuracies of the models in Figures 6.5 to 6.11, there is only one model, the cognitive PSO model, that achieved good accuracy on both the training and testing data. This model is essentially an extension of the cognitive PSO failure prediction model from Figure 6.1, where the left-most S* leaf node is split to distinguish between the success classes S++, S and S–. The confusion matrix shows that only 4 instances in the testing dataset were incorrectly classified: 1 instance of class S was predicted as S–, 1 instance of class S– was predicted as F and 2 instances of class F were predicted as S–. The entire tree is expressed using the following fuzzy rules:

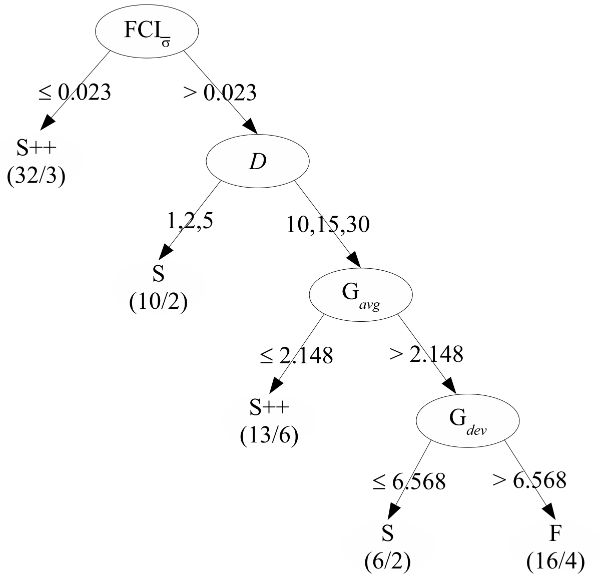
- If the dimension is 1, then the problem is always solved and fast.
- If the dimension is 2 and the micro ruggedness is fairly low, then the problem is sometimes solved.
- If the dimension is 2 and the micro ruggedness is fairly high, then the problem is almost solved.
- If the dimension is medium to high (5, 10, 15, 30) and the macro ruggedness is very low, then the problem is almost solved.
- If the dimension is medium to high (5, 10, 15, 30) and the macro ruggedness is not very low, then the algorithm fails to solve the problem.

The final rule above corresponds to the rule for cognitive PSO failure in Table 6.4.

The only other performance prediction model that achieved reasonable testing accuracy results is the modified barebones PSO model shown in Table 6.11. The tree is expressed in the following fuzzy rules:

Table 6.5: Global best PSO performance prediction model

Traditional gbest PSO performance prediction



Training accuracy: 77.9%

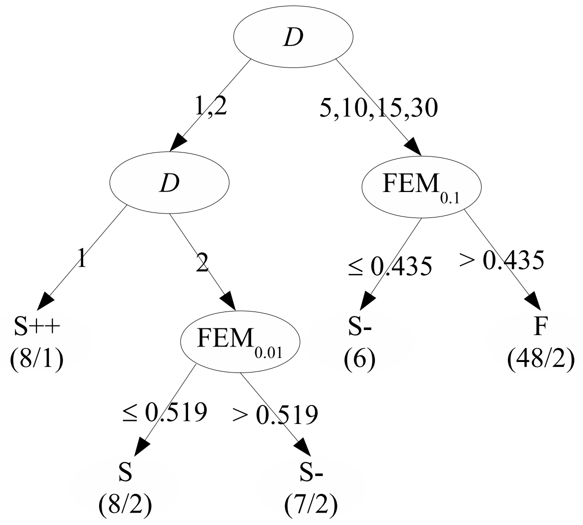
Testing accuracy: 66.7%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
19	0	3	0	0	S++
1	0	0	0	0	S+
4	0	3	0	1	S
2	0	0	0	0	S−
1	0	1	0	4	F

Table 6.6: Cognitive PSO performance prediction model

Cognitive PSO performance prediction



Training accuracy: 90.9%

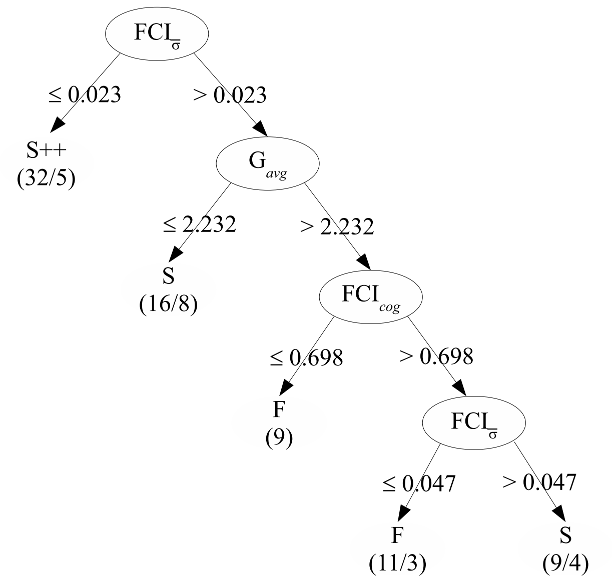
Testing accuracy: 89.7%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
5	0	0	0	0	S++
0	0	0	0	0	S+
0	0	3	1	0	S
0	0	0	6	1	S−
0	0	0	2	21	F

Table 6.7: Social PSO performance prediction model

Social PSO performance prediction



Training accuracy: 74%

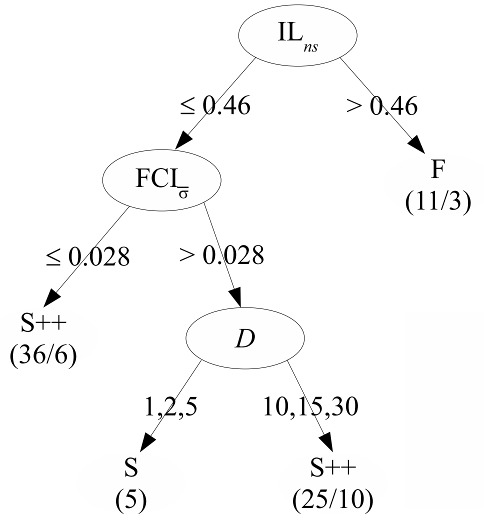
Testing accuracy: 46.2%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
11	0	4	0	1	S++
1	0	0	0	0	S+
6	0	3	0	2	S
0	0	2	0	0	S−
2	0	3	0	4	F

Table 6.8: Local best PSO performance prediction model

Local best PSO performance prediction



Training accuracy: 75.3%

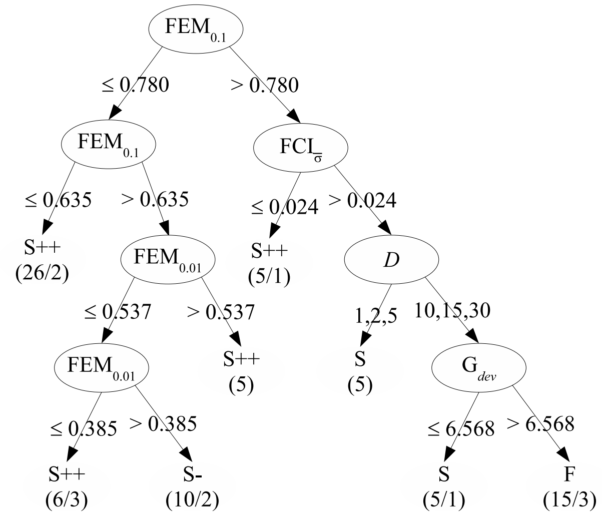
Testing accuracy: 64.1%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
20	0	3	0	0	S++
3	0	0	0	0	S+
2	0	1	0	0	S
2	0	0	0	1	S−
3	0	0	0	4	F

Table 6.9: Asynchronous global best PSO performance prediction model

Asynchronous gbest PSO performance prediction



Training accuracy: 84.4%

Testing accuracy: 61.5%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
17	0	3	3	0	S++
1	0	0	0	0	S+
4	0	2	0	0	S
2	0	0	0	0	S–
1	0	1	0	5	F

- If FCI$_{cog}$ is low (many cognitive updates result in a deterioration in fitness) and micro ruggedness is fairly low, then the problem is sometimes solved.

- If FCI$_{cog}$ is low and micro ruggedness is fairly high, then the algorithm fails to solve the problem. (This rule corresponds with the rule for modified barebones PSO failure in Table 6.4.

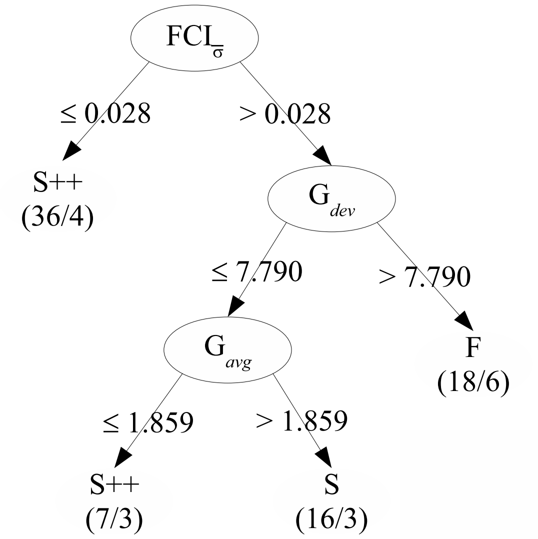
- If FCI$_{cog}$ is not low and IL$_{ns}$ is not very high (the information landscape is not very different from the information landscape of Spherical), then the problem is always solved and fast.

- If FCI$_{cog}$ is not low and IL$_{ns}$ is very high (the information landscape of the problem is very different from the information landscape of Spherical), then the problem is sometimes solved.

From the diagonal of the confusion matrix, 24 of the testing instances were correctly predicted as S++, 1 was correctly predicted as S, and 3 were correctly predicted as F. There were 11 incorrectly predicted instances.

Table 6.10: Barebones PSO performance prediction model

Barebones PSO performance prediction



Training accuracy: 79.2%

Testing accuracy: 59%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
17	0	3	3	0	S++
1	0	0	0	0	S+
4	0	2	0	0	S
2	0	0	0	0	S–
1	0	1	0	5	F

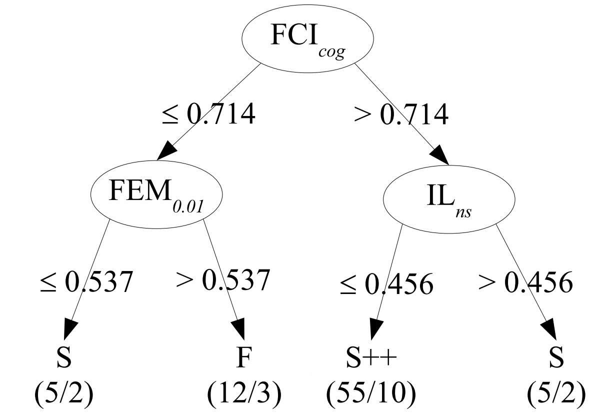
The detail of the other performance prediction models are not discussed further due to the low testing accuracy rates and the questionable validity of the models. The low accuracy rates are probably due to insufficient data to cover all performance classes with under-representation of certain performance classes in the training data. For example, for the traditional gbest PSO, there are only two instances falling into performance class S+ and six instances falling into performance class S–. This is insufficient data for building models and for generalising well on unseen data. This issue is discussed further in Section 7.2

6.5 Summary

This chapter investigated the feasibility of developing prediction models for PSO performance. Using 24 benchmark functions with different dimensions, a dataset of 116 problem instances was generated. For each problem instance the algorithms for the ten

Table 6.11: Modified barebones PSO performance prediction model

Modified barebones PSO performance prediction



Training accuracy: 77.9%

Testing accuracy: 71.8%

Confusion matrix (testing data)

Predicted classes					
S++	S+	S	S −	F	
24	0	1	0	1	S++
0	0	0	0	0	S+
4	0	1	0	2	S
0	0	2	0	0	S–
1	0	0	0	3	F

proposed fitness landscape measures were run. The fitness landscape metrics in addition to the dimension of the problem resulted in 11 features characterising each problem instance. The 116 problem instances were then solved using the seven PSO variations to result in seven actual performance classes for each instance. The dataset was divided into a training dataset of 77 problem instances (2/3) and a testing dataset of 39 problem instances (1/3).

The C4.5 decision tree induction algorithm was used to firstly produce prediction models for algorithm failure or success (where success was defined as some level of success, including finding a solution fairly close to the actual solution). Models with over 90% accuracy for both the training and the testing data were achieved for the traditional gbest PSO, cognitive PSO, local best PSO, asynchronous PSO and modified barebones PSO. The two models that had lower testing accuracies were the barebones PSO model that achieved a training accuracy of 96.1% and a testing accuracy of 87.2% and the social PSO model with a training accuracy of 93.5% and a lower testing accuracy of 76.9%. In addition to the value of the models as predictors of PSO failure, the models also provide insight into the algorithms themselves. Each decision tree was re-expressed as fuzzy rules for predicting algorithm failure. For example, for the traditional gbest PSO algorithm and the barebones PSO, the most important factors affecting algorithm failure is the combination of problems with high deviations in the gradient and high dimensions. While for the cognitive PSO, problems with medium to high dimensions that also have medium to high macro ruggedness are predicted to fail. Although the accuracy of these rules would need to be evaluated on 'unseen' problems (functions that were not part of the study), these initial results provide valuable insights into the PSO algorithm. One general observation that is evident from the models is that PSO performance is strongly influenced by gradients. Understanding the kinds of problems that are difficult for an algorithm makes it possible for each algorithm to be used on the problems that are appropriate to the algorithm. In theory this means that every algorithm should have its own set of problems on which to perform well, which essentially ties in with the no-free-lunch theorems for search/optimisation [180, 181].

Secondly, the C4.5 algorithm was used to produce models for predicting five different performance classes (one outright failure and four degrees of success) for each PSO algorithm. The performance prediction models were less successful than the failure prediction models. A possible reason for this is that the dataset was not big enough and the success classes were not adequately represented in the training and testing datasets. The one model that achieved good training and testing accuracy levels was for the cognitive PSO algorithm (90.9% accuracy for training data and 89.7% for the testing data). Future work is needed to ascertain whether a larger dataset with better representation of all success classes will be able to produce better prediction models for the other algorithms.

Chapter 7

Conclusions

This chapter summarises the main findings of the research. The objectives are revisited and discussed and a number of areas for future work are outlined.

7.1 Summary of conclusions

The first objective of this research was to conduct a survey of existing techniques for *a priori* characterisation of optimisation problems. To link the survey to fitness landscapes, each technique was categorised in terms of the fitness landscape feature measured (called the 'Focus' of the technique in the survey). For some techniques, the focus was not obvious. For example, fitness distance correlation (Technique 7) is described by the authors as a difficulty measure. Although the description of fitness distance correlation is fairly straight-forward, it is not obvious what this translates to in fitness landscape terms (with 'difficulty' not sufficient for capturing the focus). By tying each technique to a fitness landscape feature, it became possible to identify gaps in terms of features that do not have associated techniques for measuring them. In addition to the focus, the 'Search Independence', 'Assumptions' and 'Result' descriptors all proved valuable in terms of highlighting ways in which the existing techniques could be adapted for practical use in different contexts.

The second objective was to develop a characteriser for continuous optimisation problems. Various techniques were investigated and the following fitness landscape metrics were proposed for characterising problems:

- The first entropic measures of micro and macro ruggedness: The adaptation of an existing technique [170, 171, 172] for discrete landscapes required a random walk in continuous spaces. A new algorithm was developed, called a progressive random walk algorithm, that resulted in better coverage of the search space than a simple random walk algorithm. The proposed walk used a neighbourhood structure based on hypercubes rather than the usual hyperspheres. The implication of using a hypercube-based neighbourhood is that computationally expensive Euclidean distances are avoided. In addition, a single measure of ruggedness was proposed based on the maximum point of the entropy graph over different ε values (specifying the level beyond which fitness values are regarded as different). Based on experiments with different random walk step sizes, two ruggedness measures were proposed with step sizes of 1% (micro ruggedness) and 10% (macro ruggedness) of the domain of the problem.

- A dispersion metric for approximating the global landscape structure, or presence of funnels: The only adaptation from the original metric [84] was to normalise the distances for comparison between problems with different domains.

- Gradient average and deviation measures: These were new measures based on a random walk with equal-sized steps in multi-dimensional space, called a Manhattan progressive random walk.

- A fitness distance correlation searchability measure: The existing fitness distance correlation measure [68] was used with the fittest solution from a sample instead of the known optimum.

- An information landscape negative searchability measure: An existing measure of hardness based on information landscapes [14, 15] was adapted for continuous spaces using a shifted Spherical function as the reference landscape (a known easy problem).

- A fitness cloud index based on cognitive or social PSO updates: The existing fitness cloud technique [173] was implemented using PSO cognitive and social updates for determining neighbours. The fitness cloud index single measure was proposed as the proportion of fitness-improving elements in the cloud.

- The fitness cloud index mean standard deviation: The final proposed measure quantifies the average unpredictability of fitness-improving social and cognitive updates.

All of the measures above meet the required properties as outlined in Section 1.4.2. Each measure was shown to correlate in some way to PSO algorithm performance (possibly only in some dimensions) and therefore show potential value as a part-predictor of PSO performance. All measures can be used on unknown problems, only requiring a fitness function and domain for each variable. The output of each measure is a single numerical value and, finally, the computational work required to execute the algorithm for each measure is significantly less than the computational work required to solve the problem using multiple algorithms.

The third objective of this research was to develop prediction models for PSO performance, with a first sub-objective of predicting PSO failure. Using the proposed measures in addition to the dimension of the problem on a range of benchmark functions and dimensions, decision trees were induced for predicting failure for seven different PSO algorithms. For most PSO models, very high levels of accuracy were achieved for predicting failure. The decision trees are not only valuable as predictors of failure, but also provide valuable insight into the algorithms themselves in terms of the kinds of problems that the algorithm would potentially struggle to solve. The objective of predicting different degrees of algorithm performance was less successful. The different performance classes were not adequately represented in the dataset, providing insufficient patterns for inducing accurate decision trees.

7.2 Future work

There are a number of areas of future work that have emerged as a result of this study. These are briefly discussed below.

7.2.1 Validating failure prediction models with other functions

The PSO failure prediction models presented in this study were the result of a preliminary investigation into the feasibility of using fitness landscape measures to predict algorithm performance and were based on a limited number of benchmark functions. Although one set of problem instances was used to induce the decision trees and another set was used to test the models, these two sets used the same set of functions to produce instances of different dimensions. Further work should validate whether the models are accurate for other functions not used in this study.

7.2.2 Predicting PSO performance with a larger dataset

As discussed in Section 7.1, the PSO performance prediction models did not achieve adequate levels of accuracy. A more extensive dataset of additional problems with better representation of the different performance classes is required to properly test whether reasonable PSO performance prediction can be achieved. Although there are many functions defined in literature, not all are suitable. Suitable functions would have known optima, pose different levels of difficulty for PSO algorithms, and ideally be defined for multiple dimensions. Although knowledge of the optima is not needed to generate the fitness landscape measures, it is needed to generate performance data for the decision tree induction.

7.2.3 Further analysis of proposed fitness landscape measures

There are a number of ways in which the proposed fitness landscape measures in this thesis can be studied further. For example, it was shown in Chapter 5 that there was a very strong negative correlation between the FDC_s and IL_{ns} measures. If these two measures are so closely linked, then one could be redundant, but further investigation is needed to confirm this. Another issue not addressed in this thesis is the minimum computational workload required for each measure to provide just enough information to sufficiently characterise a problem. For example, in this study, the ruggedness measures were based on D walks with 1000 steps (a sample size of $1000 \times D$). Future work could analyse the effect of a range of sample sizes on the micro and macro ruggedness measures.

In this way the estimated minimum workload for adequately characterising problems can be determined. Future work could also include investigating the value of the measures as predictors of performance for other real-parameter optimisation algorithms, such as differential evolution or CMA-ES.

7.2.4 Measuring other landscape features

There are a number of landscape features that have not been investigated in this thesis. The level of variable interdependency (or lack of separability) in a problem affects the fitness landscape. One of the landscape features that can arise due to dependency between variables are ridges [100]. Finding ways of estimating variable interdependency in a function or ridges in a landscape is an area of future work. Other important landscape features that have not been investigated include neutrality and symmetry. Future work could include finding computationally inexpensive ways of approximating these features for continuous problems.

7.2.5 Further adaptations of existing fitness landscape analysis techniques

In Table 2.2, Way forward 4 described the possibility of adapting existing fitness landscape analysis techniques by generalising the notion of neighbourhood. For example, fitness clouds (Technique 16 in the survey of Table 2.1) is based on a sample of points and neighbours from the application of a search operator. This thesis investigated the use of PSO updates as the basis for fitness clouds and found that the resulting measures were useful for predicting PSO failure. An additional avenue worth investigating is the use of fitness clouds where neighbours are determined using a generic notion of neighbourhood, such as Euclidean distance. The result would be a fitness cloud index measure that is not specific to PSOs.

7.2.6 Automating algorithm selection

There is potential for taking this work further into the automated selection of algorithms for optimisation, where the fitness landscape analysis techniques are part of a larger system. Possible frameworks for such a system include:

- Population-based algorithm portfolios (or PAPs) [116] where multiple algorithms are run in parallel with communication between them. An essential part of PAP is selecting a set of algorithms to make up the portfolio. Failure prediction models such as those developed in this work could form part of a measure of risk associated with selecting a particular algorithm.

- An extension of the algorithm selection framework by Muñoz *et al.* [102] where on-line fitness landscape analysis is executed in parallel with an optimisation loop, using a portfolio of algorithms. Of particular relevance are the FCI fitness landscape measures as these are based on the effect of search operators on fitness. As an algorithm starts searching, information can be extracted to predict the likelihood of later failure.

7.2.7 Applying techniques to real-world problems

Finally, a rich area of future work is in applying the proposed fitness landscape analysis techniques to probe and better understand real-world problems. Real-world optimisation problems do not suffer from the limitations of artificial benchmark functions and could potentially raise different challenges not encountered in this thesis.

Bibliography

[1] C. W. Ahn and R. S. Ramakrishna. On the Scalability of Real-Coded Bayesian Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 12(3):307–322, June 2008.

[2] L. Altenberg. The Evolution of Evolvability in Genetic Programming. In K. Kinnear, editor, *Advances in Genetic Programming*, pages 47–74. MIT Press, Cambridge, 1994.

[3] L. Altenberg. Fitness Distance Correlation Analysis: An Instructive Counterexample. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, 1997.

[4] E. Angel and V. Zissimopoulos. On the Hardness of the Quadratic Assignment Problem with Metaheuristics. *Journal of Heuristics*, 8:399–414, July 2002.

[5] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, Bristol, UK, 1997.

[6] L. Barnett. Ruggedness and neutrality – the NKp family of fitness landscapes. In *Proceedings of the Sixth International Conference on Artificial Life*, pages 18–27, 1998.

[7] W. Beaudoin, S. Verel, P. Collard, and C. Escazut. Deceptiveness and Neutrality: The ND Family of Fitness Landscapes. In *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pages 507–514, 2006.

[8] M. Bélaidouni and J.-K. Hao. An Analysis of the Configuration Space of the Maximal Constraint Satisfaction Problem. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 49–58. Springer-Verlag, 2000.

[9] M. Bélaidouni and J.-K. Hao. Landscapes of the Maximal Constraint Satisfaction Problem. In *Artificial Evolution*, volume 1929 of *Lecture Notes in Computer Science*, pages 244–255. Springer-Verlag, 2000.

[10] H.-G. Beyer. Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. *Computer Methods in Applied Mechanics and Engineering*, 186:239–267, June 2000.

[11] G. Bilchev and I. C. Parmee. The ant colony metaphor for searching continuous design spaces. In *Selected Papers from AISB Workshop on Evolutionary Computing*, pages 25–39, 1995.

[12] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the Fourteenth International Genetic and Evolutionary Computation Conference*, pages 313–320, 2012.

[13] Y. Borenstein and R. Poli. Fitness Distributions and GA Hardness. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, pages 11–20. Springer Berlin/Heidelberg, 2004.

[14] Y. Borenstein and R. Poli. Information landscapes. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 1515–1522, 2005.

[15] Y. Borenstein and R. Poli. Information landscapes and problem hardness. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 1425–1431, 2005.

[16] Y. Borenstein and R. Poli. Kolmogorov complexity, Optimization and Hardness. In *IEEE Congress on Evolutionary Computation*, pages 112–119, July 2006.

[17] Y. Borenstein and R. Poli. Decomposition of fitness functions in random heuristic search. In *Proceedings of the 9th International Conference on Foundations of Genetic Algorithms*, volume 4436 of *Lecture Notes in Computer Science*, pages 123–137. Springer-Verlag, Berlin, Heidelberg, 2007.

[18] P. Caamaño, F. Bellas, J. A. Becerra, and R. J. Duro. Application domain study of evolutionary algorithms in optimization problems. In *Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference*, pages 495–502, 2008.

[19] P. Caamaño, A. Prieto, J. A. Becerra, F. Bellas, and R. J. Duro. Real-valued multimodal fitness landscape characterization for evolution. In *Proceedings of the 17th International Conference on Neural Information Processing: Theory and Algorithms - Volume Part I*, pages 567–574, 2010.

[20] P. Caamaño, J. A. Becerra, F. Bellas, and R. J. Duro. Are evolutionary algorithm competitions characterizing landscapes appropriately. In *Proceedings of the 13th annual Genetic and Evolutionary Computation Conference, Companion*, pages 695–702, 2011.

[21] A. Carlisle and G. Dozier. An Off-The-Shelf PSO. In *Proceedings of the Workshop on Particle Swarm Optimization*, pages 1–6, 2001.

[22] V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[23] T. Chen, K. Tang, G. Chen, and X. Yao. Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):1–22, February 2010.

[24] T. Chen, K. Tang, G. Chen, and X. Yao. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436:54–70, June 2012.

[25] J. C. Culberson. On the Futility of Blind Search. Technical report, Department of Computing Science, University of Alberta, Edmonton, Canada, July 1996.

[26] A. Czarn, C. MacNish, K. Vijayan, and B. Turlach. The detrimentality of crossover. In *Proceedings of the 20th Australian Joint Conference on Advances in Artificial Intelligence*, pages 632–636, 2007.

[27] Y. Davidor. Epistasis Variance: A Viewpoint on GA-Hardness. In G. J. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 23–35. Morgan Kaufmann, 1991.

[28] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[29] K. Deb and D. E. Goldberg. Sufficient Conditions for Deceptive and Easy Binary Functions. *Annals of Mathematics and Artificial Intelligence*, 10(4):385–408, December 1994.

[30] M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian).* PhD thesis, Politecnico di Milano, Italy, 1992.

[31] R. Eberhart and J. Kennedy. A New Optimizer using Particle Swarm Theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, 1995.

[32] R. Eberhart and Y. Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 84–88, July 2000.

[33] A. E. Eiben and T. Bäck. An Empirical Investigation of Multi-Parent Recombination Operators in Evolution Strategies. *Evolutionay Computation*, 5(3):347–365, 1997.

[34] A. E. Eiben and M. Jelasity. A Critical Note on Experimental Research Methodology In EC. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, pages 582–587, 2002.

[35] A. P. Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 191–202. Springer-Verlag, Berlin, Heidelberg, 2010.

[36] C. Fonlupt, D. Robilliard, and P. Preux. A Bit-Wise Epistasis Measure for Binary Search Spaces. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 47–56, London, UK, 1998. Springer-Verlag.

[37] W. Fontana, P. F. Stadler, E. G. Bornberg-Bauer, T. Griesmacher, I. L. Hofacker, M. Tacker, P. Tarazona, E. D. Weinberger, and P. Schuster. RNA Folding and Combinatory Landscapes. *Physical Review E*, 47:2083–2099, 1993.

[38] S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms II*. Morgan Kaufmann, San Mateo, CA, 1993.

[39] M. Gallagher. Fitness distance correlation of neural network error surfaces: A scalable, continuous optimization problem. In *Proceedings of the 12th European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Computer Science*, pages 157–166. Springer-Verlag, 2001.

[40] A. H. Gandomi and A. H. Alavi. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831 – 4845, 2012.

[41] J. Garnier and L. Kallel. How to Detect all Maxima of a Function. In *Theoretical aspects of evolutionary computing*, pages 343–370. Springer-Verlag, London, UK, 2001.

[42] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, 76(2):60–68, February 2001.

[43] F. Glover. Tabu Search – Part I. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[44] F. Glover. Tabu Search – Part II. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[45] D. E. Goldberg. Simple Genetic Algorithms and the Minimal Deceptive Problem. In L.Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 6, pages 74–88. Pitman, London, 1987.

[46] D. E. Goldberg. Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. *Complex Systems*, 3:153–171, 1989.

[47] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[48] J. J. Grefenstette. Deception considered harmful. In D. L. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 75–91. Morgan Kaufmann, San Mateo, CA, 1993.

[49] P. D. Grünwald and P. M. B. Vitányi. Algorithmic information theory. In P. Adriaans and J. van Benthem, editors, *Philosophy of Information*, pages 281–317. Elsevier, 2008.

[50] H. Guo and W. H. Hsu. GA-Hardness Revisited. In *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, volume 2724 of *Lecture Notes in Computer Science*, pages 1584–1585. Springer-Verlag, Berlin, 2003.

[51] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, November 2009.

[52] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009.

[53] N. Hansen and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 282–291. Springer Berlin / Heidelberg, 2004.

[54] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[55] J. He, C. Reeves, C. Witt, and X. Yao. A Note on Problem Difficulty Measures in Black-Box Optimization: Classification, Realizations and Predictability. *Evolutionary Computation*, 15(4):435–443, 2007.

[56] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, March 2004.

[57] R. B. Heckendorn, D. Whitley, and S. Rana. Nonlinearity, Hyperplane Ranking and the Simple Genetic Algorithm. In *Foundations of Genetic Algorithms 4*, pages 181–202. Morgan Kaufmann, 1997.

[58] S. Helwig and R. Wanka. Theoretical Analysis of Initial Particle Swarm Behavior. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, pages 889–898, 2008.

[59] F. Herrera, M. Lozano, and D. Molina. Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. *European Journal of Operational Research*, 169(2):450 – 476, 2006.

[60] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, August 1998.

[61] W. Hordijk. A Measure of Landscapes. *Evolutionary Computation*, 4(4):335–360, 1996.

[62] W. Hordijk and P. F. Stadler. Amplitude Spectra of Fitness Landscapes. *Advances in Complex Systems*, 1(1):39–66, 1998.

[63] J. Horn and D. E. Goldberg. Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 243–269. Morgan Kaufmann, San Francisco, CA, 1995.

[64] E. Izquierdo-Torres. Evolving Dynamical Systems: Nearly Neutral Regions in Continuous Fitness Landscapes. Master's thesis, University of Sussex, 2004.

[65] T. Jansen. On Classifications of Fitness Functions. In *Theoretical aspects of evolutionary computing*, pages 371–385. Springer-Verlag, London, UK, 2001.

[66] M. Jelasity, B. Tóth, and T. Vinkó. Characterizations of Trajectory Structure of Fitness Landscapes Based on Pairwise Transition Probabilities of Solutions. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 623–630, 1999.

[67] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. Phd thesis, The University of New Mexico, 1995.

[68] T. Jones and S. Forrest. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, 1995.

[69] L. Kallel, B. Naudts, and C. R. Reeves. Properties of Fitness Functions and Search Landscapes. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 175–206. Springer, Berlin, 2001.

[70] S. Kauffman. Adaptation on rugged fitness landscapes. In E.Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison-Wesley, Reading, 1989.

[71] S. Kauffman and S. Levin. Towards a General Theory of Adaptive Walks on Rugged Landscapes. *Journal of Theoretical Biology*, 128(1):11–45, September 1987.

[72] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, USA, 1993.

[73] J. Kennedy. The Particle Swarm: Social Adaptation of Knowledge. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 303–308, April 1997.

[74] J. Kennedy. Bare bones particle swarms. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 80–87, 2003.

[75] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1942–1948, 1995.

[76] K. E. Kinnear. Fitness Landscapes and Difficulty in Genetic Programming. In *Proceedings of the First IEEE Conference on Evolutionary Computing*, pages 142–147, 1994.

[77] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[78] C.-Y. Lee and X. Yao. Evolutionary programming using mutations based on the levy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13, February 2004.

[79] B. Levitan and S. Kauffman. Adaptive walks with noisy fitness measurements. *Molecular Diversity*, 1(1):53–68, 1995.

[80] M. Lipsitch. Adaptation on rugged landscapes generated by iterated local interactions of neighboring genes. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 128–135, 1991.

[81] M. Locatelli. A Note on the Griewank Test Function. *Journal of Global Optimization*, 25:169–174, February 2003.

[82] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers, 2002.

[83] G. Lu, J. Li, and X. Yao. Fitness-Probability Cloud and a Measure of Problem Hardness for Evolutionary Algorithms. In *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 108–117, Berlin, Heidelberg, 2011. Springer-Verlag.

[84] M. Lunacek and D. Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pages 477–484, 2006.

[85] N. Madras. *Lectures on Monte Carlo Methods*. Fields Institute monographs. American Mathematical Society, 2002.

[86] K. M. Malan and A. P. Engelbrecht. Fitness landscape evolvability metrics for part-prediction of pso performance. *Swarm Intelligence*, Under Review, Submitted September 2013.

[87] K. M. Malan and A. P. Engelbrecht. Quantifying Ruggedness of Continuous Landscapes using Entropy. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1440–1447, 2009.

[88] K. M. Malan and A. P. Engelbrecht. Ruggedness, Funnels and Gradients in Fitness Landscapes and the Effect on PSO Performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 963–970, 2013.

[89] K. M. Malan and A. P. Engelbrecht. Steep Gradients as a Predictor of PSO Failure. In *Proceedings of the Fifteenth International Conference on Genetic and Evolutionary Computation Conference, Companion*, pages 9–10, 2013.

[90] K. M. Malan and A. P. Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[91] K. M. Malan and A. P. Engelbrecht. Fitness Landscape Analysis for Metaheuristic Performance Prediction. In H. Richter and A. P. Engelbrecht, editors, *Recent advances in the theory and application of fitness landscapes*, volume 6 of *Emergence, Complexity and Computation*, pages 103–132. Springer, 2014.

[92] B. Manderick, M. K. de Weger, and P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150, 1991.

[93] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 829–836, 2011.

[94] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12:303–325, September 2004.

[95] P. Merz and B. Freisleben. Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, November 2000.

[96] S. K. Mishra. Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program. Technical report, Social Science Research Network (SSRN), August 2006.

[97] S. K. Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. Technical Report 2718, University Library of Munich, Germany, August 2006.

[98] M. Mitchell, S. Forrest, and J. H. Holland. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254, 1992.

[99] M. Mitchell, J. H. Holland, and S. Forrest. When Will a Genetic Algorithm Outperform Hill Climbing? In *Advances in Neural Information Processing Systems 6*, pages 51–58, 1994.

[100] R. Morgan and M. Gallagher. Using Landscape Topology to Compare Continuous Metaheuristics: A Framework and Case Study on EDAs and Ridge Structure. *Evolutionary Computation*, 20(2):277–299, June 2012.

[101] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization problems. In *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature – Part I*, pages 226–235, 2012.

[102] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. The Algorithm Selection Problem on the Continuous Optimization Domain. In C. Moewes and A. Nürnberger, editors, *Computational Intelligence in Intelligent Data Analysis*, volume 445 of *Studies in Computational Intelligence*, pages 75–89. Springer Berlin Heidelberg, 2013.

[103] C. L. Müller, B. Baumgartner, and I. F. Sbalzarini. Particle Swarm CMA Evolution Strategy for the Optimization of Multi-Funnel Landscapes. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pages 2685–2692, 2009.

[104] C. L. Müller and I. F. Sbalzarini. Global characterization of the CEC 2005 fitness landscapes using fitness-distance analysis. In *Proceedings of the 2011 International Conference on Applications of Evolutionary Computation – Part I*, pages 294–303, 2011.

[105] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. Landscape characterization of numerical optimization problems using biased scattered data. In *IEEE Congress on Evolutionary Computation*, pages 1–8, June 2012.

[106] B. Naudts and L. Kallel. Some Facts About So Called GA-Hardness Measures. Technical Report 379, CMAP, Ecole Polytechnique, France, 1998.

[107] B. Naudts and L. Kallel. A Comparison of Predictive Measures of Problem Difficulty in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1, April 2000.

[108] B. Naudts and J. Naudts. The Effect of Spin-Flip Symmetry on the Performance of the Simple GA. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 67–76, 1998.

[109] B. Naudts, D. Suys, and A. Verschoren. Epistasis as a Basic Concept in Formal Landscape Analysis. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 65–72, 1997.

[110] G. Ochoa. Consensus Sequence Plots and Error Thresholds: Tools for Visualising the Structure of Fitness Landscapes. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 129–138. Springer-Verlag, London, UK, 2000.

[111] G. Ochoa, R. Qu, and E. K. Burke. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. In *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference*, pages 341–348, 2009.

[112] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos. A Study of NK Landscapes' Basins and Local Optima Networks. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 555 – 562, July 2008.

[113] P. S. Oliveto, J. He, and X. Yao. Analysis of the (1 + 1)-EA for Finding Approximate Solutions to Vertex Cover Problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1006–1029, October 2009.

[114] A. Owen and I. Harvey. Adapting Particle Swarm Optimisation for Fitness Landscapes with Neutrality. In *Proceedings of IEEE Swarm Intelligence Symposium*, pages 258 –265, April 2007.

[115] M. Pelikan. NK landscapes, problem difficulty, and hybrid evolutionary algorithms. In *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference*, pages 665–672, 2010.

[116] F. Peng, K. Tang, G. Chen, and X. Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800, October 2010.

[117] S. Picek and M. Golub. The new negative slope coefficient measure. In *Proceedings of the 10th WSEAS International Conference on Evolutionary Computing*, pages 96–101, 2009.

[118] R. Poli and L. Vanneschi. Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference*, pages 1335–1342, 2007.

[119] K. V. Price, R. M. Storn, and J. A. Lampinen. Appendix A.1: Unconstrained Uni-Modal Test Functions. In *Differential Evolution A Practical Approach to Global Optimization*, Natural Computing Series, pages 514–533. Springer-Verlag, Berlin, Germany, 2005.

[120] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[121] R. J. Quick, V. J. Rayward-Smith, and G. D. Smith. Fitness Distance Correlation and Ridge Functions. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 77–86, 1998.

[122] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[123] N. J. Radcliffe and P. D. Surry. Fitness Variance of Formae and Performance Prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms III*, pages 51–72, 1995.

[124] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605–1614, May 2007.

[125] S. Rana. *Examining the role of local optima and schema processing in genetic search*. PhD thesis, Colorado State University, USA, 1999. Adviser: Whitley, Darrell.

[126] S. Rana, L. D. Whitley, and R. Cogswell. Searching in the Presence of Noise. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 198–207, 1996.

[127] W. M. Rand. *Controlled observations of the genetic algorithm in a changing environment: case studies using the shaky ladder hyperplane-defined functions*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 2005. Chair-Holland, John H. and Chair-Riolo, Rick L.

[128] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog, 1973.

[129] C. R. Reeves. Predictive Measures for Problem Difficulty. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 736–743, 1999.

[130] C. R. Reeves and C. C. Wright. Epistasis in Genetic Algorithms: An Experimental Design Perspective. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 217–224, 1995.

[131] C. M. Reidys and P. F. Stadler. Neutrality in fitness landscapes. *Applied Mathematics and Computation*, 117(2-3):321–350, 2001.

[132] J. R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65 – 118, 1976.

[133] J. Riley and V. Ciesielski. Analysis of the difficulty of learning goal-scoring behaviour for robot soccer. In *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pages 1569–1576, 2006.

[134] T. P. Riopka. Adapting to complexity during search in combinatorial landscapes. In *Proceedings of the 2003 International Conference on Applications of Evolutionary Computing*, pages 311–321, 2003.

[135] S. Roman. *Coding and Information Theory.* Springer-Verlag, 175 Fifth Avenue, New York, USA, 1992.

[136] H. Rosé, W. Ebeling, and T. Asselmeyer. The Density of States - a Measure of the Difficulty of Optimisation Problems. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 208–217. Springer-Verlag, London, UK, 1996.

[137] R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *Biosystems*, 39(3):263–278, 1996.

[138] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, Technical University of Berlin, 1975.

[139] F. Schweitzer, W. Ebeling, H. Rosé, and O. Weiss. Optimization of road networks using evolutionary strategies. *Evolutionary Computation*, 5(4):419–438, 1997.

[140] Y.-W. Shang and Y.-H. Qiu. A Note on the Extended Rosenbrock Function. *Evolutionary Computation*, 14:119–126, March 2006.

[141] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 69–73, 1998.

[142] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34, 2002.

[143] K. Smith-Miles. Towards insightful algorithm selection for optimisation using meta-learning concepts. In *Proceedings of the IEEE Joint Conference on Neural Networks*, pages 4118–4124, 2008.

[144] K. A. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys*, 41(1):6:1–6:25, 2008.

[145] P. F. Stadler and R. Happel. Correlation structure of the landscape of the graph-bipartitioning problem. *Journal of Physics A: Mathematical and General*, 25(11):3103–3110, 1992.

[146] P. F. Stadler. Towards a Theory of Landscapes. In R. Lopéz-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks*, volume 461, pages 77–163. Springer Verlag, Berlin, New York, 1995.

[147] P. F. Stadler. Landscapes and Their Correlation Functions. *Journal of Mathematical Chemistry*, 20:1–45, 1996.

[148] P. F. Stadler. Fitness Landscapes. In A. V. Michael Lässig, editor, *Biological Evolution and Statistical Physics*, volume 585 of *Lecture Notes in Physics*, pages 183–204. Springer-Verlag, Berlin, Heidelberg, 2002.

[149] P. Stadler and W.Schnabl. The landscape of the travelling salesman problem. *Physics Letters A*, 161(4):337–344, 1992.

[150] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore, May 2005.

[151] A. M. Sutton, D. Whitley, M. Lunacek, and A. Howe. PSO and multi-funnel landscapes: how cooperation might limit exploration. In *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pages 75–82, 2006.

[152] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[153] J. Tavares, F. B. Pereira, and E. Costa. The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2307–2314, July 2006.

[154] M. H. Tayarani and M. R. Akbarzadeh-Totonchi. Magnetic optimization algorithms a new synthesis. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2659–2664, 2008.

[155] I. Trelea. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. *Information Processing Letters*, 85:317–325, 2004.

[156] L. Trujillo, Y. Martínez, E. Galván-López, and P. Legrand. A comparison of predictive measures of problem difficulty for classification with Genetic Programming. In *ERA 2012*, Tijuana, Mexico, November 2012.

[157] L. Trujillo, Y. Martínez, E. Galván López, and P. Legrand. A comparative study of an evolvability indicator and a predictor of expected performance for genetic programming. In *Proceedings of the Fourteenth International Genetic and Evolutionary Computation Conference, Companion*, pages 1489–1490, 2012.

[158] P. D. Turney. Increasing Evolvability Considered as a Large-Scale Trend in Evolution. In *Proceedings of 1999 Genetic and Evolutionary Computation Conference, Workshop Program*, pages 43–46, 1999.

[159] F. van den Bergh and A. Engelbrecht. A Study of Particle Swarm Optimization Particle Trajectories. *Information Sciences*, 176(8):937–971, 2006.

[160] F. Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[161] C. Van Hoyweghen and B. Naudts. Symmetry in the search space. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1072 – 1078, 2000.

[162] L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. Phd thesis, Faculty of Sciences, University of Lausanne, 2004.

[163] L. Vanneschi. Investigating Problem Hardness of Real Life Applications. In R. L. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation, chapter 7, pages 107–125. Springer, Ann Arbor, 17-19May 2007.

[164] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Verel. Fitness Clouds and Problem Hardness in Genetic Programming. In K. Deb, editor, *Proceedings of Genetic and Evolutionary Computation Conference*, volume 3103 of *Lecture Notes in Computer Science*, pages 690–701. Springer Berlin Heidelberg, 2004.

[165] L. Vanneschi, Y. Pirola, and P. Collard. A quantitative study of neutrality in GP boolean landscapes. In *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pages 895–902, 2006.

[166] L. Vanneschi, M. Tomassini, P. Collard, and S. Verel. Negative Slope Coefficient: A Measure to Characterize Genetic Programming Fitness Landscapes. In *Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 178–189. Springer Berlin / Heidelberg, 2006.

[167] L. Vanneschi, M. Tomassini, P. Collard, S. Verel, Y. Pirola, and G. Mauri. A Comprehensive View of Fitness Landscapes with Neutrality and Fitness Clouds. In *Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 241–250. Springer Berlin / Heidelberg, 2007.

[168] L. Vanneschi, S. Verel, M. Tomassini, and P. Collard. NK Landscapes Difficulty and Negative Slope Coefficient: How Sampling Influences the Results. In *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, volume 5484 of *Lecture Notes in Computer Science*, pages 645–654. Springer-Verlag, Berlin, Heidelberg, 2009.

[169] V. K. Vassilev. An Information Measure of Landscapes. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 49–56, 1997.

[170] V. K. Vassilev. *Fitness Landscapes and Search in the Evolutionary Design of Digital Circuits*. Phd thesis, Napier University, 2000.

[171] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information Characteristics and the Structure of Landscapes. *Evolutionary Computation*, 8(1):31–60, 2000.

[172] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application. In *Advances in Evolutionary Computing: Theory and Applications*, pages 3–44. Springer-Verlag New York, Inc., 2003.

[173] S. Verel, P. Collard, and M. Clergue. Where are bottlenecks in NK fitness landscapes? In *Proceedings of the 2003 Congress on Evolutionary Computation*, volume 1, pages 273–280, 2003.

[174] S. Verel, P. Collard, M. Tomassini, and L. Vanneschi. Neutral Fitness Landscape in the Cellular Automata Majority Problem. In *Cellular Automata*, volume 4173 of *Lecture Notes in Computer Science*, pages 258–267. Springer Berlin / Heidelberg, 2006.

[175] S. Vérel, G. Ochoa, and M. Tomassini. Local Optima Networks of NK Landscapes With Neutrality. *IEEE Transactions on Evolutionary Computation*, 15(6):783–797, 2011.

[176] H. Waeselynck, P. Thévenod-Fosse, and O. Abdellatif-Kaddour. Simulated annealing applied to test generation: landscape characterization and stopping criteria. *Empirical Software Engineering*, 12(1):35–63, 2007.

[177] E. Weinberger. Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference. *Biological Cybernetics*, 63(5):325–336, September 1990.

[178] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias. Evaluating Evolutionary Algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.

[179] L. D. Whitley, K. E. Mathias, and L. D. Pyeatt. Hyperplane Ranking in Simple Genetic Algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 231–238, 1995.

[180] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, February 1995.

[181] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[182] S. Wright. The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in evolution. In *Proceedings of the Sixth International Congress on Genetics*, pages 356–366, 1932.

[183] S. Wright. Surfaces of Selective Value Revisited. *The Americal Naturalist*, 131(1):115–123, January 1988.

[184] B. Xin, J. Chen, and F. Pan. Problem difficulty analysis for particle swarm optimization: deception and modality. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 623–630, 2009.

[185] X.-S. Yang. Firefly algorithms for multimodal optimization. In O. Watanabe and T. Zeugmann, editors, *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, volume 5792 of *Lecture Notes in Computer Science*, pages 169–178. Springer, Berlin, Heidelberg, 2009.

[186] X.-S. Yang and S. Deb. Cuckoo Search via Lévy Flights. In *World Congress on Nature & Biologically Inspired Computing*, pages 210–214, December 2009.

[187] X. Yao, Y. Liu, and G. Lin. Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.

Appendix A

Benchmark Functions

This appendix defines 24 real-valued benchmark functions used in this study. The function definitions are listed alphabetically in Table A.1. Each function includes a reference to an existing source where the function is defined. The mathematical formula is provided with the domain and the known global optimum. Graphical plots of each function are provided in Figures A.1 to A.3.

Table A.1: Benchmark Functions (D is the dimension of the problem)

Function	Definition, domain and global optimum (f^*)
Ackley [187]	$f_{ack}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e,$ $x_i \in [-32, 32], \qquad f^*_{ack} = f_{ack}(0,\ldots,0) = 0.$
Alpine [124]	$f_{alp}(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert, \qquad x_i \in [-10, 10], \qquad f^*_{alp}(0,\ldots,0) = 0.$
Beale [96]	$f_{bea}(x_1, x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2,$ $x_i \in [-4.5, 4.5], \qquad f^*_{bea}(3, 0.5) = 0.$
Bohachevsky [53]	$f_{boh}(\mathbf{x}) = \sum_{i=1}^{D-1}\left(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7\right), \qquad D \geq 2,$ $x_i \in [-15, 15], \qquad f^*_{boh} = f_{boh}(0,\ldots,0) = 0.$
Egg Holder [97]	$f_{egg}(x_1, x_2) = -(x_2 + 47)\sin\left(\sqrt{\lvert x_2 + x_1/2 + 47\rvert}\right) + \sin\left(\sqrt{\lvert x_1 - (x_2 + 47)\rvert}\right)(-x_1),$ $x_i \in [-512, 512], \qquad f^*_{egg} = f_{egg}(512, 404.23181) \approx -959.640662720823.$
Goldstein-Price [187]	$f_{gp}(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$ $x_i \in [-2, 2], \qquad f^*_{gp}(0, -1) = 3.$
Griewank [187]	$f_{grw}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$ $x_i \in [-600, 600], \qquad f^*_{grw} = f_{grw}(0,\ldots,0) = 0.$

Continued on Next Page...

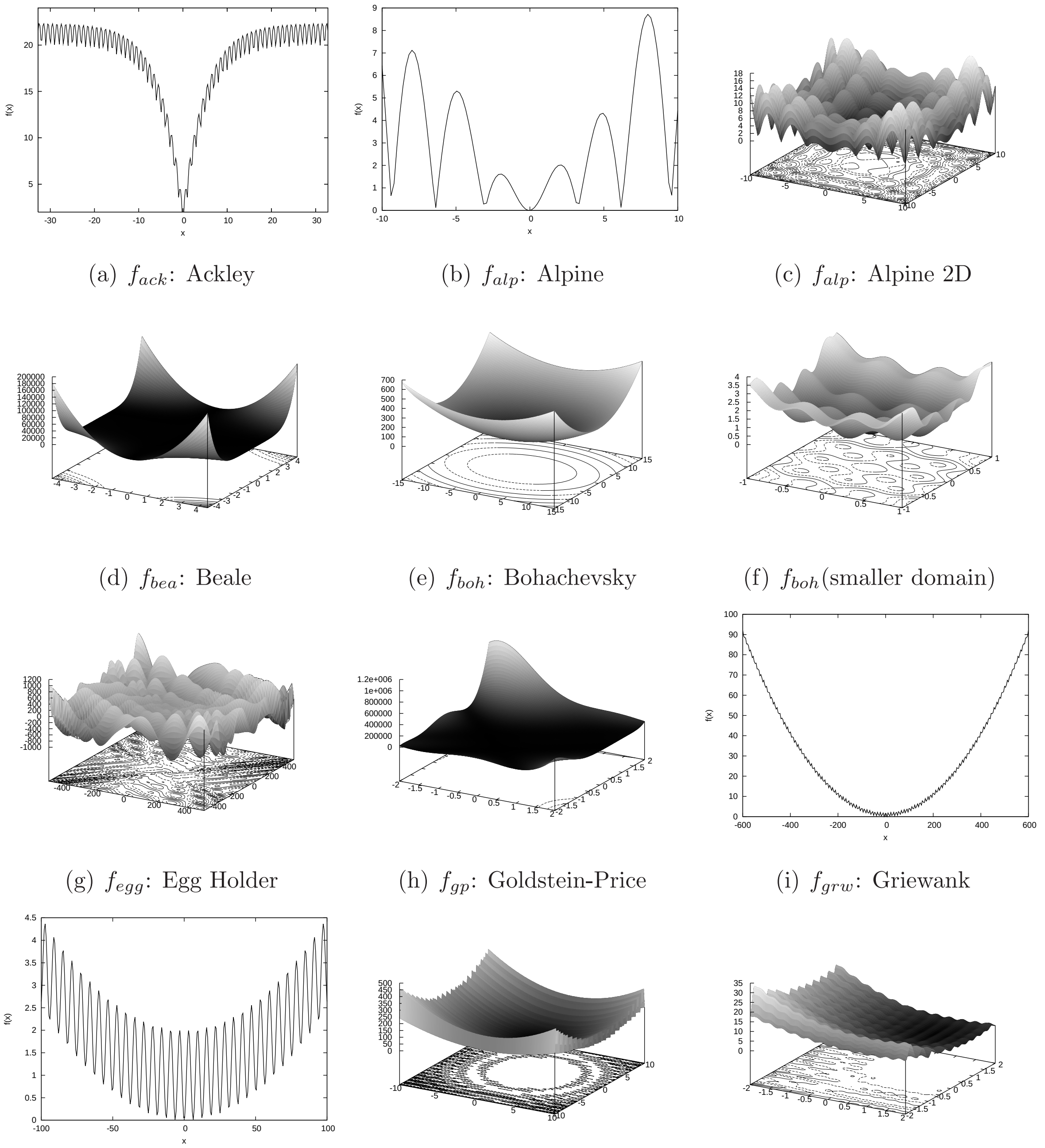
Table A.1 – Continued

Function	**Definition, domain and global optimum** (f^*)
Levy 13 [97]	$f_{lvy}(\mathbf{x}) = \sum_{i=1}^{D-1} \left(\sin^2(3\pi x_i) + (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_{i+1} - 1)^2 \left[1 + \sin^2(2\pi x_{i+1})\right]\right),$ $D \geq 2, \qquad x_i \in [-10, 10], \qquad f^*_{lvy} = f_{lvy}(1, \ldots, 1) = 0.$
Michalewicz [96]	$f_{mic}(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i) \left(\sin(i x_i^2/\pi)\right)^{2p}, \quad x_i \in [0, \pi],$ for $\quad p = 10, \quad f^*_{mic} \approx -1.8013 \ (D = 2), \quad f^*_{mic} \approx -4.6877 \ (D = 5),$ $f^*_{mic} \approx -96602 \ (D = 10), \quad f^*_{mic} = -29.6309 \ (D = 30).$
Pathological [124]	$f_{pth}(\mathbf{x}) = \sum_{i=1}^{D-1} \left(0.5 + \frac{\sin^2 \sqrt{(100x_i^2 + x_{i+1}^2)} - 0.5}{1 + 0.001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2}\right), \quad D \geq 2,$ $x_i \in [-100, 100], \qquad f^*_{pth} = f_{pth}(0, \ldots, 0) = 0.$
Quadric (Schwefel 1.2)[187]	$f_{qdr}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2, \qquad x_i \in [-100, 100], \qquad f^*_{qdr} = f_{qdr}(0, \ldots, 0) = 0.$
Quartic [187]	$f_{qrt}(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4, \qquad x_i \in [-1.28, 1.28], \qquad f^*_{qrt} = f_{qrt}(0, \ldots, 0) = 0.$
Rana (expanded) [119]	$f_{ran}(\mathbf{x}) = \sum_{i=1}^{D} x_i \sin(\alpha) \cos(\beta) + \left(x_{(i+1) \bmod D} + 1\right) \cos(\alpha) \sin(\beta), \quad D \geq 2,$ where $\alpha = \sqrt{\lvert x_{i+1} + 1 - x_i \rvert} \qquad$ and $\quad \beta = \sqrt{\lvert x_i + x_{i+1} + 1 \rvert},$ $x_i \in [-512, 512], \qquad f^*_{ran} = f_{ran}(-512, \ldots, -512).$
Rastrigin [187]	$f_{ras}(\mathbf{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right), \qquad x_i \in [-5.12, 5.12], \qquad f^*_{ras} = f_{ras}(0, \ldots, 0) = 0.$
Rosenbrock [187] (generalized)	$f_{ros}(\mathbf{x}) = \sum_{i=1}^{D-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right), \quad D \geq 2,$ $x_i \in [-2.048, 2.048], \qquad f^*_{ros} = f_{ros}(1, \ldots, 1) = 0.$

Continued on Next Page…

Table A.1 – Continued

Function	**Definition, domain and global optimum** (f^*)
Salomon [119]	$f_{sal}(\mathbf{x}) = -\cos\left(2\pi \sum_{i=1}^{D} x_i^2\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2} + 1,$ $x_i \in [-100, 100], \qquad f^*_{sal} = f_{sal}(0, \ldots, 0) = 0.$
Schwefel 2.22 [187]	$f_{sch2.22}(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert, \qquad x_i \in [-10, 10], \qquad f^*_{sch2.22} = f_{sch2.22}(0, \ldots, 0) = 0.$
Schwefel 2.26 [187]	$f_{sch2.26}(\mathbf{x}) = -\sum_{i=1}^{D} \left(x_i \sin(\sqrt{\lvert x_i \rvert})\right),$ $x_i \in [-500, 500], \qquad f^*_{sch2.26} = f_{sch2.26}(420.9687, \ldots, 420.9687).$
Six-hump camel-back [187]	$f_{6h}(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$ $x_i \in [-5, 5], \quad f^*_{6h}(0.08983, -0.7126) = f^*_{6h}(-0.08983, 0.7126) = -1.0316285.$
Skew Rastrigin [53]	$f_{skr}(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left(y_i^2 - 10\cos(2\pi y_i)\right), \quad \text{where} \quad y_i = \begin{cases} 10x_i & \text{if } x_i > 0, \\ x_i & \text{otherwise,} \end{cases}$ $x_i \in [-5, 5], \qquad f^*_{skr} = f_{skr}(0, \ldots, 0) = 0.$
Spherical [28]	$f_{sph}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2, \qquad x_i \in [-100, 100], \qquad f^*_{sph} = f_{sph}(0, \ldots, 0) = 0.$
Step [187]	$f_{stp}(\mathbf{x}) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor\right)^2, \qquad x_i \in [-20, 20], \qquad f^*_{stp} = f_{stp}(0, \ldots, 0) = 0.$
Weierstrass [96]	$f_{wei}(\mathbf{x}) = \sum_{i=1}^{D} \sum_{k=0}^{20} \left[0.5^k \cos(2\pi 3^k (x_i + 0.5))\right] - D \sum_{k=0}^{20} \left[0.5^k \cos(2\pi 3^k 0.5)\right],$ $x_i \in [-0.5, 0.5], \qquad f^*_{wei} = f_{wei}(0, \ldots, 0) = 0.$
Zakharov [96]	$f_{zak}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + \left[\sum_{i=1}^{D} i x_i / 2\right]^2 + \left[\sum_{i=1}^{D} i x_i / 2\right]^4, \quad D \geq 2,$ $x_i \in [-5, 10], \qquad f^* = f(0, \ldots, 0) = 0.$



(a) f_{ack}: Ackley

(b) f_{alp}: Alpine

(c) f_{alp}: Alpine 2D

(d) f_{bea}: Beale

(e) f_{boh}: Bohachevsky

(f) f_{boh}(smaller domain)

(g) f_{egg}: Egg Holder

(h) f_{gp}: Goldstein-Price

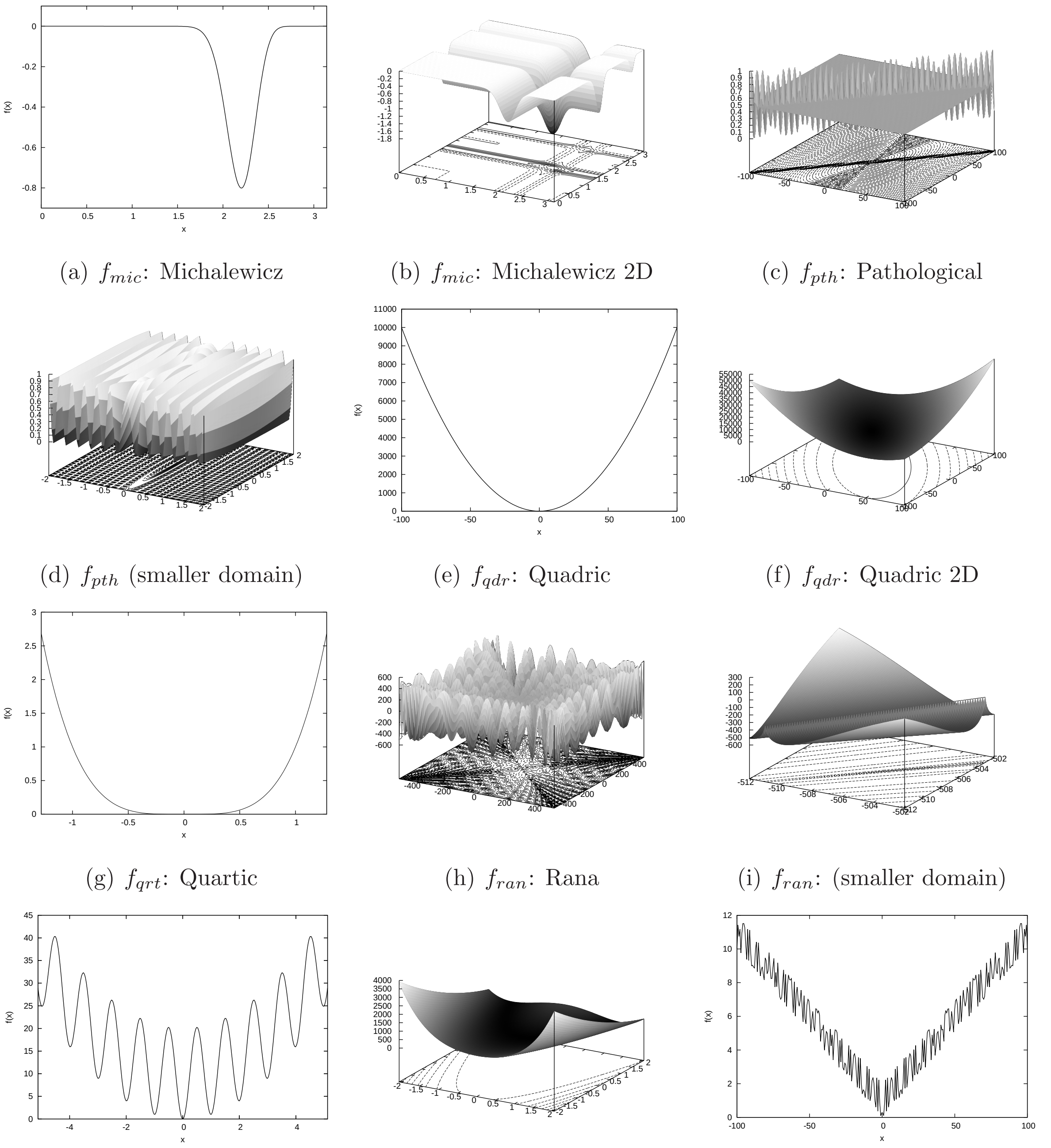
(i) f_{grw}: Griewank

(j) f_{grw} (smaller domain)

(k) f_{lvy}: Levy 13

(l) f_{lvy} (smaller domain)

Figure A.1: Graphical plots of the benchmarks defined in Table A.1. Functions that are not defined for 1D are plotted in 2D. When the landscape structure is not clear for the defined domain, the function is also plotted with a reduced domain. Selected functions are plotted in 1D and 2D. Plots are continued in Figures A.2 and A.3.



(a) f_{mic}: Michalewicz

(b) f_{mic}: Michalewicz 2D

(c) f_{pth}: Pathological

(d) f_{pth} (smaller domain)

(e) f_{qdr}: Quadric

(f) f_{qdr}: Quadric 2D

(g) f_{qrt}: Quartic

(h) f_{ran}: Rana

(i) f_{ran}: (smaller domain)

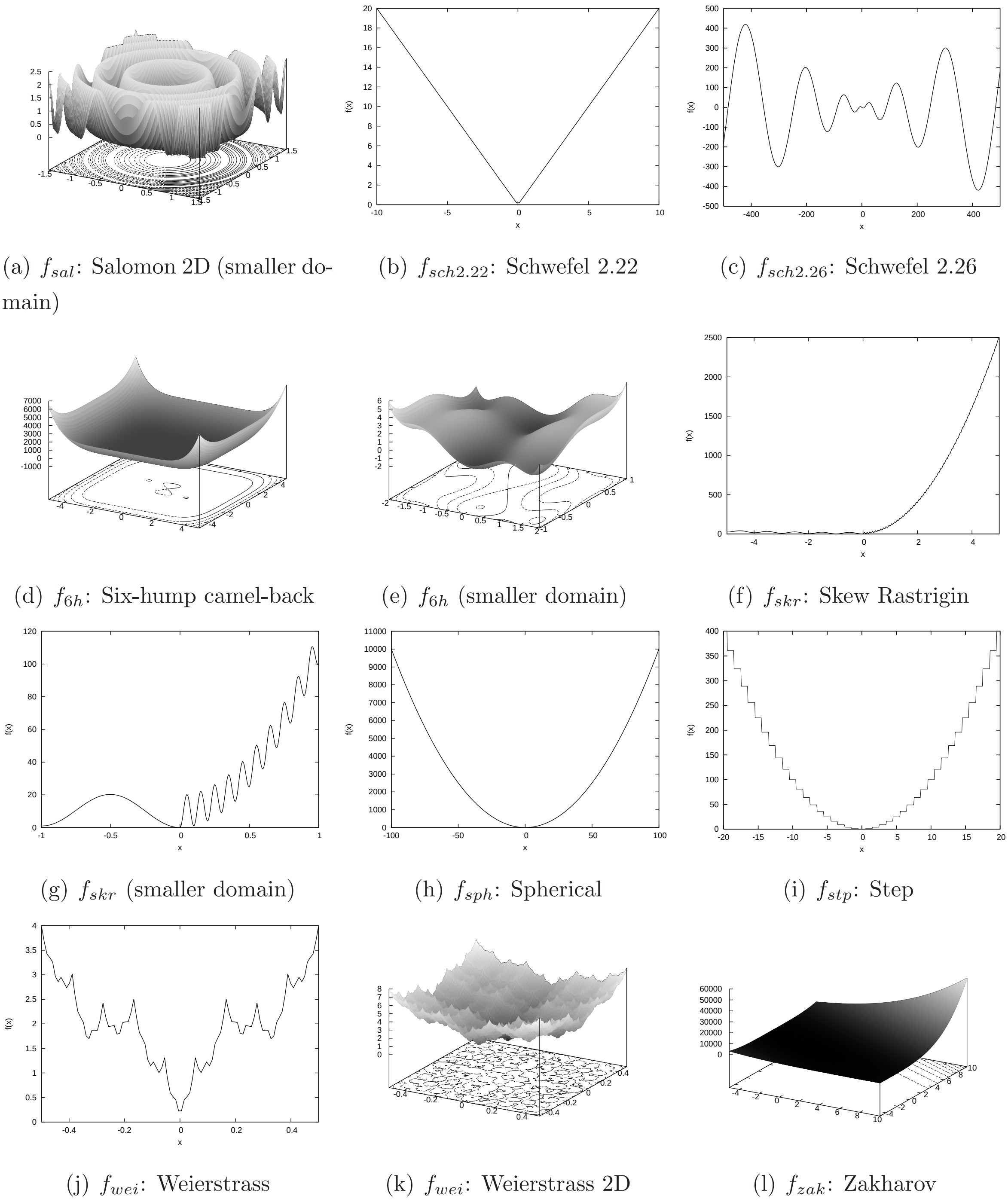
(j) f_{ras}: Rastrigin

(k) f_{ros}: Rosenbrock

(l) f_{sal}: Salomon

Figure A.2: Graphical plots of the benchmarks defined in Table A.1. Functions that are not defined for 1D are plotted in 2D. When the landscape structure is not clear for the defined domain, the function is also plotted with a reduced domain. Selected functions are plotted in 1D and 2D. This figure is a continuation of Figure A.1 and is continued in Figure A.3.

.



(a) f_{sal}: Salomon 2D (smaller domain)

(b) $f_{sch2.22}$: Schwefel 2.22

(c) $f_{sch2.26}$: Schwefel 2.26

(d) f_{6h}: Six-hump camel-back

(e) f_{6h} (smaller domain)

(f) f_{skr}: Skew Rastrigin

(g) f_{skr} (smaller domain)

(h) f_{sph}: Spherical

(i) f_{stp}: Step

(j) f_{wei}: Weierstrass

(k) f_{wei}: Weierstrass 2D

(l) f_{zak}: Zakharov

Figure A.3: Graphical plots of the benchmarks defined in Table A.1. Functions that are not defined for 1D are plotted in 2D. When the landscape structure is not clear for the defined domain, the function is also plotted with a reduced domain. Selected functions are plotted in 1D and 2D. This figure is a continuation of Figures A.1 and A.2.

.

Appendix B

Proposed Fitness Landscape Metrics

Based on the investigation into fitness landscape measures for continuous problems in Chapters 4 and 5, ten fitness landscape metrics are proposed for characterising the ruggedness, gradients, presence of funnels, general searchability and searchability with respect to PSO search. Table B.1 summarises these metrics in terms of the parameters required, the result produced and the efficiency.

Table B.1: Summary of proposed fitness landscape metrics

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
FEM$_{0.01}$ FEM$_{0.1}$	First entropic measures of micro and macro ruggedness.	Number of steps in the random walk, ns. Step bounds of 1% and 10% of the range of the domain used for FEM$_{0.01}$ and FEM$_{0.1}$, respectively.	$[0, 1]$: where 0 indicates a flat landscape and 1 indicates maximal ruggedness	$ns + 1$ function evaluations and 20 entropy calculations (for each discrete ε).
DM	Dispersion metric.	(1) Size of sample, n, (2) Size of sub-sample of best solutions, s.	$[-disp_D, \sqrt{D} - disp_D]$: where $disp_D$ is the dispersion of a large uniform random sample of a D-dimensional space normalised to $[0, 1]$ in all dimensions. A positive value for DM indicates the presence of multiple funnels.	n fitness evaluations, sorting of n elements ($n\log(n)$) and $\frac{s^2-s}{2}$ Euclidean distance calculations.
G$_{avg}$	Average estimated gradient.	(1) Number of steps in the Manhattan progressive random walk, ns, (2) step size, s (a proportion of the range of the domain).	A positive real number, where a higher value indicates higher average gradients.	$ns + 1$ function evaluations and ns gradient estimations (Equation 4.12).

Continued on Next Page. . .

Table B.1 – Continued

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
G$_{dev}$	Standard deviation from the average estimated gradient.	(1) Number of steps in the Manhattan progressive random walk, ns, (2) step size, s (a proportion of the range of the domain).	A positive real number, where a higher value indicates higher deviations from average gradients.	$ns + 1$ function evaluations and ns gradient estimations (Equation 4.12).
FDC$_s$	Fitness distance correlation searchability measure.	size of sample, n.	$[-1, 1]$: For a minimisation problem, 1 indicates the highest measure of searchability (perfect correlation between fitness values and distance to the fittest solution).	n fitness evaluations and n distance calculations in solution space.
IL$_{ns}$	Information landscape negative searchability measure.	size of sample, n.	$[0, 1]$: A value of 0 indicates maximum searchability (no difference from the reference landscape vector $\mathbf{v}_r$).	$2n$ fitness evaluations for problem and reference landscape; $(n-1)(n-2)$ memory requirement for both information landscape vectors.
FCI$_{cog}$ FCI$_{soc}$	Fitness cloud index based on cognitive or social PSO updates.	(1) size of sample, n, (2) inertia weight, w, (3) acceleration constants, c_1 (for FCI$_{cog}$) or c_2 (for FCI$_{soc}$).	$[0, 1]$: indicating the proportion of fitness improving solutions after two PSO updates.	$3n$ fitness evaluations, $2n$ solution updates.

Continued on Next Page...

Table B.1 – Continued

Proposed Measure		**Parameters**	**Result: range and interpretation**	**Efficiency**
$\text{FCI}_{\overline{\sigma}}$	Fitness cloud index mean standard deviation.	(1) size of sample, n, (2) inertia weight, w, (3) acceleration constants, c_1 and c_2.	$[0, 0.509]$: indicating the level of deviation from the FCI mean for both update strategies.	30 × ($3n$ fitness evaluations, $2n$ solution updates).

Appendix C

Training and Testing Datasets

Tables C.1 and C.2 list the training and testing datasets used as the basis for the decision tree induction described in Chapter 6. The column headings are as follows:

- f indicates the function as defined in Appendix B, Table A.1.
- D indicates the dimension of the problem.
- Columns $\text{FEM}_{0.01}$ to $\text{FCI}_{\overline{\sigma}}$ are the fitness landscape metrics as described in Table B.1.
- The last seven columns provide the performance class (described in Section 3.3.4) resulting from solving the problem with the applicable PSO algorithm. The algorithms are abbreviated as follows: 'gbest' for global best PSO, 'cog' for cognitive PSO, 'soc' for social PSO, 'lbest' for local best PSO, 'async' for asynchronous global best PSO, 'bb' for barebones PSO and 'mbb' for modified barebones PSO.

Section 6.2 gives the detail of how the dataset was generated and split into testing and training sets.

Table C.1: Training data set

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{ack}	2	0.858	0.780	-0.357	33.562	22.058	0.774	0.191	0.770	0.907	0.015	S++	S-	S++	S++	S++	S++	S++
f_{ack}	5	0.867	0.832	-0.333	35.314	20.681	0.701	0.243	0.737	0.935	0.015	S++	F	S++	S++	S++	S++	S++
f_{ack}	10	0.869	0.839	-0.300	21.925	11.971	0.572	0.341	0.728	0.927	0.024	S++	F	S	S++	S++	S++	S++
f_{ack}	30	0.870	0.846	-0.270	3.818	3.094	0.431	0.367	0.716	0.853	0.032	S	F	F	S++	S	S	S++
f_{alp}	1	0.488	0.849	-0.151	7.897	6.216	0.608	0.279	0.827	0.835	0.016	S++	S++	S++	S++	S++	S++	S++
f_{alp}	10	0.605	0.861	-0.178	10.940	8.774	0.300	0.405	0.742	0.777	0.057	S++	F	S++	S++	S++	S++	S++
f_{alp}	15	0.606	0.861	-0.179	10.712	8.693	0.295	0.415	0.730	0.771	0.059	S++	F	S	S++	S++	S++	S++
f_{alp}	30	0.601	0.862	-0.172	9.686	7.790	0.266	0.416	0.714	0.641	0.068	S++	F	S-	S++	S++	S++	S++
f_{boh}	2	0.522	0.581	-0.353	2.071	1.425	0.922	0.101	0.913	0.906	0.014	S++	S-	S++	S++	S++	S++	S++
f_{boh}	5	0.473	0.635	-0.341	2.068	1.447	0.850	0.206	0.852	0.931	0.018	S++	F	S	S++	S++	S++	S++
f_{boh}	15	0.354	0.645	-0.331	2.043	1.290	0.637	0.302	0.775	0.848	0.038	S	F	S	S++	S	S	S++
f_{boh}	30	0.290	0.643	-0.331	2.011	1.243	0.559	0.334	0.702	0.694	0.048	S-	F	S-	S	S-	S	S
f_{egg}	2	0.557	0.846	0.119	11.315	11.753	0.006	0.506	0.749	0.688	0.101	S	F	S	S	S	S	S
f_{gp}	2	0.271	0.350	-0.269	1.647	2.929	0.521	0.244	0.887	0.689	0.018	S++	S	S++	S++	S++	S++	S++
f_{grw}	1	0.788	0.550	-0.289	7.962	4.152	0.967	0.017	0.929	0.862	0.013	S++	S++	S++	S++	S++	S++	S++
f_{grw}	5	0.478	0.644	-0.358	2.148	1.312	0.903	0.118	0.843	0.935	0.018	S	F	S-	S-	S-	S	S
f_{grw}	10	0.387	0.646	-0.347	2.065	1.205	0.728	0.272	0.804	0.898	0.029	S-	F	S-	S	S-	S	S
f_{grw}	30	0.286	0.641	-0.328	2.009	1.201	0.567	0.313	0.703	0.705	0.066	S	F	S	S	S	S	S
f_{lvy}	2	0.761	0.766	-0.354	16.227	25.702	0.916	0.097	0.812	0.925	0.015	S++	S	S++	S++	S++	S++	S++
f_{lvy}	5	0.822	0.807	-0.348	23.811	30.063	0.845	0.210	0.767	0.946	0.021	S++	F	S++	S++	S++	S++	S++
f_{lvy}	15	0.829	0.820	-0.314	4.993	5.749	0.641	0.305	0.730	0.889	0.034	S	F	S	S++	S	S	S++

Continued on Next Page…

Table C.1 – Continued

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{lvy}	30	0.830	0.816	-0.301	4.434	4.118	0.545	0.334	0.701	0.755	0.047	S	F	S	S++	S	S	S++
f_{mic}	2	0.313	0.575	-0.162	4.010	6.568	0.432	0.309	0.877	0.876	0.013	S++	S-	S++	S++	S++	S++	S++
f_{pth}	2	0.415	0.405	-0.065	18.061	82.369	0.023	0.504	0.704	0.639	0.057	S+	S	S++	S	S++	S++	S
f_{pth}	5	0.490	0.641	-0.013	11.034	31.148	0.011	0.498	0.692	0.703	0.056	S-	F	F	F	S-	S-	S-
f_{pth}	10	0.678	0.783	-0.006	10.520	29.025	0.002	0.501	0.687	0.784	0.050	F	F	F	F	F	F	F
f_{pth}	15	0.751	0.818	-0.001	8.980	22.655	0.005	0.501	0.686	0.821	0.048	F	F	F	F	F	F	F
f_{pth}	30	0.820	0.849	0.005	4.050	8.554	0.003	0.500	0.687	0.878	0.043	F	F	F	F	F	F	F
f_{qdr}	2	0.519	0.597	-0.329	1.373	0.963	0.648	0.227	0.906	0.857	0.012	S++	S	S++	S++	S++	S++	S++
f_{qdr}	5	0.452	0.567	-0.213	0.906	0.863	0.342	0.376	0.876	0.828	0.015	S++	S-	S++	S++	S++	S++	S++
f_{qdr}	10	0.374	0.521	-0.133	0.685	0.734	0.186	0.456	0.864	0.797	0.023	S++	F	S++	S++	S++	S++	S++
f_{qdr}	15	0.355	0.482	-0.087	0.726	0.739	0.115	0.460	0.861	0.806	0.023	S++	F	S++	S+	S++	S++	S+
f_{qrt}	1	0.334	0.435	-0.296	2.000	2.269	0.867	0.002	0.963	0.880	0.012	S++	S++	S++	S++	S++	S++	S++
f_{qrt}	15	0.419	0.699	-0.287	1.859	2.636	0.519	0.337	0.742	0.724	0.052	S++	F	S++	S++	S++	S++	S++
f_{qrt}	30	0.371	0.707	-0.277	1.729	2.528	0.459	0.358	0.666	0.509	0.057	S++	F	S++	S++	S++	S++	S++
f_{ran}	2	0.470	0.816	0.051	12.801	19.217	0.017	0.484	0.727	0.610	0.148	S	F	S	S	S	S	S-
f_{ran}	10	0.717	0.869	0.055	18.589	18.431	0.003	0.496	0.698	0.650	0.074	F	F	F	F	F	F	F
f_{ran}	15	0.730	0.871	0.044	17.465	16.779	0.008	0.499	0.696	0.645	0.061	F	F	F	F	F	F	F
f_{ras}	5	0.602	0.865	-0.239	14.190	7.078	0.499	0.317	0.731	0.807	0.041	S	F	F	S	S	S	S++
f_{ras}	15	0.596	0.864	-0.232	16.765	8.395	0.393	0.373	0.714	0.766	0.047	F	F	F	F	F	F	S
f_{ras}	30	0.591	0.865	-0.227	16.894	8.387	0.358	0.381	0.711	0.675	0.063	F	F	F	F	F	F	F
f_{ros}	2	0.372	0.507	-0.220	1.340	1.700	0.546	0.287	0.894	0.674	0.020	S++	S	S++	S++	S++	S++	S++
f_{ros}	5	0.467	0.640	-0.311	1.157	1.378	0.687	0.249	0.846	0.806	0.035	S	S-	S	S	S-	S-	S-

Continued on Next Page. . .

Table C.1 – Continued

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{ros}	10	0.444	0.679	-0.291	1.162	1.324	0.630	0.306	0.785	0.811	0.036	S-	F	S	S-	S-	S-	S-
f_{ros}	15	0.411	0.687	-0.280	1.061	1.199	0.555	0.301	0.756	0.757	0.037	S-	F	S	S-	S-	S-	S
f_{sal}	10	0.889	0.803	-0.336	33.673	23.190	0.709	0.286	0.733	0.884	0.035	F	F	F	F	F	F	F
f_{sal}	15	0.889	0.805	-0.324	24.737	18.815	0.627	0.285	0.710	0.840	0.041	F	F	F	F	F	F	F
f_{sal}	30	0.885	0.802	-0.318	20.055	14.220	0.534	0.323	0.655	0.673	0.049	F	F	F	F	F	F	F
$f_{sch2.22}$	1	0.589	0.638	-0.297	2.000	0.000	1.000	0.003	0.964	0.878	0.011	S++	S++	S++	S++	S++	S++	S++
$f_{sch2.22}$	2	0.528	0.606	-0.329	2.069	0.988	0.864	0.135	0.891	0.955	0.011	S++	S-	S++	S++	S++	S++	S++
$f_{sch2.22}$	5	0.318	0.431	-0.244	0.830	1.174	0.569	0.286	0.836	0.959	0.017	S++	S-	S++	S++	S++	S++	S++
$f_{sch2.22}$	15	0.249	0.377	-0.230	0.025	0.078	0.150	0.376	0.774	0.861	0.037	S++	S-	S	S++	S++	S++	S++
$f_{sch2.22}$	30	0.270	0.382	-0.230	0.000	0.000	0.048	0.382	0.751	0.797	0.075	S++	S-	S-	S++	S++	S++	S++
$f_{sch2.26}$	2	0.537	0.846	0.035	7.640	4.765	0.300	0.405	0.783	0.523	0.027	S	S-	S	S++	S	S++	S++
$f_{sch2.26}$	5	0.564	0.852	0.038	9.731	6.079	0.171	0.447	0.771	0.629	0.048	S	F	S	S	S	S	S++
$f_{sch2.26}$	15	0.576	0.855	0.021	12.611	8.028	0.080	0.475	0.770	0.638	0.038	F	F	F	F	F	F	S
$f_{sch2.26}$	30	0.577	0.855	0.024	13.502	8.927	0.065	0.485	0.771	0.643	0.044	F	F	F	F	F	F	S
f_{6h}	2	0.374	0.529	-0.346	1.992	3.107	0.778	0.138	0.914	0.817	0.021	S++	S	S++	S++	S++	S++	S++
f_{skr}	1	0.439	0.355	-0.242	1.266	1.605	0.552	0.288	0.877	0.633	0.019	S++	S++	S++	S++	S++	S++	S++
f_{skr}	2	0.527	0.468	-0.298	2.133	2.650	0.732	0.209	0.889	0.732	0.037	S	S-	S	S	S	S	S++
f_{skr}	5	0.545	0.593	-0.333	2.232	2.395	0.828	0.187	0.888	0.918	0.030	S	F	S-	S	S	S	S
f_{skr}	15	0.460	0.636	-0.261	2.132	2.255	0.801	0.235	0.823	0.970	0.018	S-	F	F	S-	S-	S-	S-
f_{skr}	30	0.386	0.643	-0.235	2.297	2.482	0.683	0.308	0.773	0.954	0.024	F	F	F	S-	S-	S-	S-
f_{sph}	1	0.467	0.543	-0.296	2.000	1.155	0.968	0.000	0.965	0.874	0.011	S++	S++	S++	S++	S++	S++	S++
f_{sph}	2	0.509	0.598	-0.358	2.078	1.199	0.971	0.012	0.907	0.913	0.013	S++	S	S++	S++	S++	S++	S++

Continued on Next Page. . .

Table C.1 – Continued

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{sph}	5	0.477	0.643	-0.355	2.059	1.194	0.900	0.124	0.845	0.939	0.021	S++	F	S++	S++	S++	S++	S++
f_{sph}	10	0.385	0.649	-0.345	2.060	1.212	0.732	0.271	0.799	0.900	0.028	S++	F	S++	S++	S++	S++	S++
f_{sph}	15	0.347	0.647	-0.338	2.045	1.204	0.666	0.281	0.770	0.842	0.036	S++	F	S++	S++	S++	S++	S++
f_{stp}	1	0.515	0.549	-0.293	2.000	11.375	0.966	0.029	0.849	0.829	0.017	S++	S++	S++	S++	S++	S++	S++
f_{stp}	5	0.812	0.664	-0.357	2.085	4.955	0.894	0.134	0.839	0.935	0.021	S++	F	S++	S++	S++	S++	S++
f_{stp}	15	0.702	0.662	-0.336	2.073	2.299	0.665	0.280	0.769	0.863	0.039	S++	F	S	S++	S++	S++	S++
f_{wei}	1	0.790	0.799	-0.294	9.801	6.275	0.911	0.138	0.832	0.832	0.017	S+	S+	S+	S+	S+	S++	S++
f_{wei}	15	0.764	0.797	-0.284	4.933	3.391	0.526	0.338	0.688	0.778	0.053	S	F	F	S++	S	S	S
f_{wei}	30	0.738	0.788	-0.288	3.792	2.421	0.464	0.359	0.614	0.523	0.070	F	F	F	S++	F	S	F
f_{zak}	5	0.305	0.310	-0.138	0.895	1.747	0.535	0.300	0.905	0.898	0.015	S++	S-	S++	S++	S++	S++	S++
f_{zak}	10	0.284	0.306	-0.124	0.646	1.466	0.462	0.331	0.907	0.934	0.013	S++	S-	S++	S++	S++	S++	S++
f_{zak}	30	0.237	0.288	-0.080	0.728	1.274	0.407	0.361	0.913	0.993	0.008	S++	S-	S	S+	S++	S	S

Table C.2: Testing data set

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{ack}	1	0.874	0.769	-0.296	13.703	8.870	0.793	0.152	0.804	0.876	0.015	S+	S++	S+	S+	S+	S++	S++
f_{ack}	15	0.870	0.849	-0.288	3.505	2.969	0.506	0.336	0.727	0.899	0.024	S	F	S	S++	S	S	S++
f_{alp}	2	0.539	0.855	-0.170	9.490	7.454	0.444	0.352	0.781	0.818	0.053	S++	S-	S++	S++	S++	S++	S++
f_{alp}	5	0.594	0.861	-0.189	10.368	8.430	0.342	0.392	0.753	0.764	0.068	S++	F	S++	S++	S++	S++	S++
f_{bea}	2	0.248	0.313	-0.225	0.888	1.624	0.308	0.345	0.901	0.713	0.022	S++	S	S	S++	S++	S++	S++
f_{boh}	10	0.391	0.646	-0.334	2.064	1.326	0.707	0.277	0.804	0.905	0.030	S	F	S	S++	S++	S	S++
f_{grw}	2	0.652	0.606	-0.358	4.499	3.209	0.966	0.037	0.904	0.906	0.014	S	S-	S	S	S	S	S++
f_{grw}	15	0.343	0.644	-0.338	2.048	1.206	0.653	0.276	0.769	0.840	0.040	S	F	S	S	S	S	S
f_{lvy}	10	0.824	0.819	-0.329	16.521	19.860	0.705	0.278	0.741	0.923	0.022	S++	F	S	S++	S++	S++	S++
f_{mic}	5	0.503	0.812	-0.131	8.368	15.553	0.222	0.436	0.767	0.793	0.028	S	F	S	S	S	S	S++
f_{mic}	10	0.648	0.863	-0.114	14.991	28.817	0.159	0.452	0.713	0.699	0.036	F	F	F	F	F	F	S
f_{mic}	30	0.802	0.869	-0.093	18.611	25.859	0.126	0.460	0.692	0.581	0.033	F	F	F	F	F	F	F
f_{qdr}	1	0.467	0.543	-0.297	2.000	1.155	0.968	0.000	0.963	0.875	0.012	S++	S++	S++	S++	S++	S++	S++
f_{qdr}	30	0.297	0.379	-0.058	0.597	0.609	0.071	0.476	0.859	0.802	0.017	S++	S-	S	S-	S++	S	S-
f_{qrt}	2	0.416	0.548	-0.356	1.987	2.469	0.849	0.092	0.919	0.889	0.013	S++	S	S++	S++	S++	S++	S++
f_{qrt}	5	0.452	0.643	-0.339	1.959	2.643	0.763	0.235	0.847	0.873	0.017	S++	S-	S++	S++	S++	S++	S++
f_{qrt}	10	0.439	0.685	-0.301	1.911	2.665	0.589	0.314	0.781	0.809	0.036	S++	F	S++	S++	S++	S++	S++
f_{ran}	5	0.692	0.867	0.069	18.150	19.740	0.012	0.500	0.701	0.665	0.089	F	F	F	F	F	F	F
f_{ran}	30	0.739	0.870	0.032	14.198	12.613	0.005	0.501	0.698	0.645	0.068	F	F	F	F	F	F	F
f_{ras}	1	0.542	0.881	-0.212	10.030	4.996	0.708	0.252	0.775	0.795	0.015	S++	S++	S++	S++	S++	S++	S++
f_{ras}	2	0.591	0.871	-0.224	11.871	5.925	0.641	0.259	0.743	0.827	0.021	S++	S-	S	S++	S++	S++	S++

Continued on Next Page...

Table C.2 – Continued

f	D	FEM$_{0.01}$	FEM$_{0.1}$	DM	G$_{avg}$	G$_{dev}$	FDC$_s$	IL$_{ns}$	FCI$_{cog}$	FCI$_{soc}$	FCI$_{\overline{\sigma}}$	gbest	cog	soc	lbest	async	bb	mbb
f_{ras}	10	0.600	0.866	-0.245	16.185	7.926	0.432	0.370	0.726	0.765	0.039	F	F	F	F	F	S	S
f_{ros}	30	0.350	0.694	-0.273	1.030	1.134	0.477	0.335	0.668	0.615	0.059	S-	F	S-	S-	S-	S-	S-
f_{sal}	1	0.891	0.802	-0.277	61.860	34.228	0.971	0.076	0.824	0.870	0.016	S++	S++	S++	S++	S++	S++	S++
f_{sal}	2	0.888	0.802	-0.346	64.260	41.572	0.960	0.083	0.799	0.900	0.014	S++	F	S	S+	S++	S	S
f_{sal}	5	0.890	0.807	-0.350	51.397	32.287	0.872	0.162	0.770	0.929	0.022	F	F	F	F	F	F	F
$f_{sch2.22}$	10	0.239	0.369	-0.238	0.171	0.396	0.276	0.352	0.794	0.939	0.023	S++	S-	S++	S++	S++	S++	S++
$f_{sch2.26}$	1	0.485	0.822	-0.003	5.853	3.649	0.317	0.397	0.821	0.520	0.018	S++	S++	S++	S++	S++	S++	S++
$f_{sch2.26}$	10	0.572	0.855	0.021	11.563	7.413	0.111	0.478	0.770	0.606	0.051	S	F	F	F	F	F	S
f_{skr}	10	0.504	0.632	-0.291	2.063	2.140	0.827	0.202	0.848	0.977	0.017	S-	F	F	S-	S-	S-	S
f_{sph}	30	0.288	0.642	-0.328	2.004	1.197	0.566	0.316	0.711	0.712	0.045	S++	F	S++	S++	S++	S++	S++
f_{stp}	2	0.703	0.618	-0.359	2.110	8.345	0.967	0.024	0.883	0.904	0.015	S++	S	S++	S++	S++	S++	S++
f_{stp}	10	0.742	0.666	-0.348	2.099	3.204	0.724	0.261	0.797	0.892	0.030	S++	F	S	S++	S++	S++	S++
f_{stp}	30	0.614	0.657	-0.330	2.012	1.441	0.564	0.311	0.701	0.691	0.056	S	F	F	S++	S	S	S++
f_{wei}	2	0.789	0.792	-0.297	9.539	6.812	0.875	0.163	0.800	0.905	0.015	S++	F	S++	S+	S++	S++	S++
f_{wei}	5	0.781	0.797	-0.293	6.957	4.832	0.728	0.283	0.760	0.902	0.034	S++	F	S++	S++	S++	S++	S++
f_{wei}	10	0.773	0.797	-0.289	5.462	3.581	0.575	0.328	0.712	0.815	0.054	S	F	S-	S++	S	S	S
f_{zak}	2	0.291	0.332	-0.334	1.270	2.103	0.670	0.182	0.917	0.918	0.013	S++	S	S++	S++	S++	S++	S++
f_{zak}	15	0.252	0.296	-0.110	0.747	1.497	0.454	0.348	0.907	0.964	0.011	S++	S-	S++	S++	S++	S++	S++

Appendix D

Acronyms

This appendix provides a list of acronyms frequently used in this thesis. Acronyms are listed alphabetically, typeset in bold, with the meaning alongside.

CMA-ES	Covariance Matrix Adaptation Evolution Strategy
DM	Dispersion metric
FCI	Fitness cloud index
FDC	Fitness distance correlation
FEM	First Entropic Measure
GA	Genetic Algorithm
MaxFES	Maximum Number of Function Evaluations
NSC	Negative slope coefficient
PSO	Particle Swarm Optimisation
SEM	Second Entropic Measure
SRate	Success rate
SSpeed	Success speed

Appendix E

Symbols

This appendix lists and defines the commonly-used symbols in the thesis. Each symbol is listed under the chapter in which they first appear.

E.1 Chapter 1

x'	A candidate solution found by a search process
$f(x')$	Fitness of candidate solution x'
x^*	Global optimum solution of a problem
$\mathbf{x}$	An n-dimensional candidate solution vector
$\mathcal{S}$	Feasible subregion of $\mathbb{R}^n$
x^{min}	Minimum boundary constraint (the same for all x_i in $\mathbf{x}$)
x^{max}	Maximum boundary constraint (the same for all x_i in $\mathbf{x}$)

E.2 Chapter 3

D	Dimension of a problem
f^*	Fitness of global optimum
$\mathbf{x}_i(t)$	Position of particle i at time step t
$\mathbf{v}_i(t)$	Velocity of particle i at time step t

w	Inertia weight
c_1	Cognitive acceleration constant
c_2	Social acceleration constant
$\mathbf{y}_i(t)$	Particle i's personal best position at time step t
$\hat{\mathbf{y}}(t)$	The global best position at time step t
$\hat{\mathbf{y}}_\mathbf{i}(t)$	The best particle in the neighbourhood of particle
$U(a,b)$	Random number between a and b sampled from a uniform distribution
$\hat{f}$	Estimated maximum fitness value
f^{min}	Best fitness found by the search
MaxFES	Maximum number of fitness evaluations
$\mathbf{FES}_r$	Number of function evaluations to reach the global optimum for run r
$\mathbf{SSpeed}_r$	Success speed of a run r
ns	Number of successful runs
S++	Class symbol for problems that are always solved and fast
S+	Class symbol for problems that are always solved
S	Class symbol for problems that are sometimes solved
S–	Class symbol for problems that are almost solved
F	Class symbol for problems that are not solved

E.3 Chapter 4

$N(\mathbf{x})$	Neighbourhood set of **x**
ε	Sensitivity margin for deciding when fitness values are different
$H(\varepsilon)$	Measure of entropy
e^*	Information stability
$\mathbf{FEM}_{0.01}$	First entropic measure of micro ruggedness
$\mathbf{FEM}_{0.1}$	First entropic measure of macro ruggedness
DM	Dispersion metric
$\mathbf{G}_{avg}$	Average estimated gradient within a walk
$\mathbf{G}_{dev}$	Standard deviation of gradient measures from the mean

E.4 Chapter 5

V_{max}	Reference information landscape vector
$\mathbf{FDC}_s$	Fitness distance correlation searchability measure
$\mathbf{IL}_{ns}$	Information landscape negative searchability measure
$\mathbf{FCI}_{cog}$	Fitness cloud index based on cognitive updates
$\mathbf{FCI}_{soc}$	Fitness cloud index based on social updates
$\mathbf{NSC}_{cog}$	Negative slope coefficient based on cognitive updates
$\mathbf{NSC}_{soc}$	Negative slope coefficient based on social updates
$\mathbf{FCI}_{\overline{\sigma}}$	Fitness cloud index mean standard deviation

Appendix F

Derived Publications

The following are the publications derived from this study:

1. K. M. Malan and A. P. Engelbrecht. Quantifying Ruggedness of Continuous Landscapes using Entropy. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1440-1447, 2009.

2. K. M. Malan and A. P. Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

3. K. M. Malan and A. P. Engelbrecht. Ruggedness, Funnels and Gradients in Fitness Landscapes and the Effect on PSO Performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 963–970, 2013.

4. K. M. Malan and A. P. Engelbrecht. Steep Gradients as a Predictor of PSO Failure. In *Proceedings of the Fifteenth International Conference on Genetic and Evolutionary Computation, Conference Companion*, pages 9–10, 2013.

5. K. M. Malan and A. P. Engelbrecht. Fitness Landscape Analysis for Metaheuristic Performance Prediction. In Hendrik Richter and Andries P. Engelbrecht, editors, *Recent advances in the theory and application of fitness landscapes*, Emergence, Complexity and Computation, volume 6, pages 103–132. Springer, 2014.

6. K. M. Malan and A. P. Engelbrecht. Fitness landscape evolvability for part-prediction of PSO performance. *Swarm Intelligence*, Submitted September 2013.

Index

$\text{FCI}_{\overline{\sigma}}$, 166, 228
FCI_{cog}, 144, 227
FCI_{soc}, 144, 227
FDC_{s}, 135, 227
$\text{FEM}_{0.01}$, 97, 226
$\text{FEM}_{0.1}$, 97, 226
G_{avg}, 102, 112, 226
G_{dev}, 103, 112, 227
IL_{ns}, 136, 227
NSC_{cog}, 144
NSC_{soc}, 144

accumulated escape probability, 46
Ackley function, 57, 219
adaptive walks, 26, 32
algorithm run, 64
Alpine function, 57, 219
amplitude spectra, 40
asynchronous global best PSO, 60
autocorrelation function, 35
average bit certainty measure, 32

bare bones PSO, 60
Beale function, 56, 219
benchmarks, 56
bit-wise epistasis, 41
Bohachevsky function, 56, 219

C4.5 algorithm, 175
cognitive PSO, 59
complete search independence, 33
consensus sequence plots, 32
correlation coefficient, 31
correlation length, 36
coverage measure, 82

dataset generation, 172
deception, 25, 47
deception for PSO, 72
deceptiveness coefficient, 25
decision tree induction, 175
density of states, 39
dispersion metric, 44, 98, 100, 226
dispersion of a sample, 98
DM, 100, 226
dynamic measures, 31

Egg Holder function, 219
entropic measure, 40, 90
entropy, 86
epistasis, 22
epistasis variance, 37
escape rate measure, 25, 32
evolvability, 29, 48

F performance class, 71
FCI, 142
FDC, 127
FEM, 40, 95
FEM algorithm, 95
fitness cloud, 43, 130
fitness distance correlation, 38, 127
fitness distribution, 24
fitness evolvability portraits, 42
fitness function, 18
fitness landscape, 18
fitness range, 63
fitness range estimation, 64
fitness range order, 65
fitness-probability cloud, 45
fixed accuracy level, 62, 64
formae variance, 37
funnels, 26, 98

GA-deception, 35
global landscape structure, 26
Goldstein-Price function, 56, 219
Griewank function, 57, 219

HDBL, 41
HDIL, 41
high search independence, 34

information landscape, 128
information landscape hardness measure, 44
information stability, 90

Kolmogorov complexity, 31

Levy 13 function, 220
local best PSO, 59
local optima network, 25
low search independence, 34

Manhattan progressive random walk, 103
MaxFES, 64, 66
medium search independence, 34
metaheuristics, 1
Michalewicz function, 56, 220
modality, 24
modified bare bones PSO, 61

negative slope coefficient, 14, 43, 131
neighbourhood set, 75
neutral network measures, 45
neutral walk, 42
neutrality, 27
NKp landscapes, 28
noise, 23
non-linear separability, 22
normalisation of fitness, 63
NSC, 131

optimisation, 1

particle swarm optimisation, 57
Pathological function, 220
performance classes, 70
performance measures, 61
performance metric, 63
progressive random walk, 77
progressive random walk algorithm, 81

PSO, 57

QMetric, 65
Quadric function, 56, 220
Quartic function, 56, 220

Rana function, 220
random walk, 74
random walk coverage, 82
Rastrigin function, 26, 57, 64, 99, 220
retrospective measures, 32
Rosenbrock function, 57, 157, 220
ruggedness, 26
ruggedness measures, 86
run of an algorithm, 64

S performance class, 70
S+ performance class, 70
S++ performance class, 70
S– performance class, 70
Salomon function, 57, 221
Schwefel 1.2 function, 220
Schwefel 2.22 function, 56, 221
Schwefel 2.26 function, 26, 57, 64, 99, 221
search independence, 33, 50
searchability, 29, 48, 125
SEM, 40
separability, 23
simple random walk, 75
simple random walk algorithm, 77
site-wise optimisation measure, 23, 31
Six-hump camel-back function, 56, 221
Skew Rastrigin function, 221
smoothness, 26
social PSO, 59
solution accuracy, 65
solution quality, 62, 65
Spherical function, 56, 221
SRate, 66
SSpeed, 67
static-ϕ metric, 39
Step function, 22, 221
success rate, 62, 66
success speed, 67
successful run, 64
symmetry, 28, 47

testing data, 229
theoretical measures, 31
traditional global best PSO, 58
training data, 229

variable interdependency, 22

Weierstrass function, 221
WEKA tool, 175
Wright's fitness landscape, 18

Zakharov function, 221