Tuning Optimization Algorithms under Multiple Objective Function Evaluation Budgets

Antoine Dymond

Tuning Optimization Algorithms under Multiple Objective Function Evaluation Budgets

by

Antoine Dymond

A thesis submitted in partial fulfillment
of the requirements for the degree

PhD of Mechanical Engineering

in the

Department of Mechanical and Aeronautical Engineering
Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria
South Africa

Abstract

The performance of optimization algorithms is sensitive to both the optimization problem's numerical characteristics and the termination criteria of the algorithm. Given these considerations two tuning algorithms named tMOPSO and MOTA are proposed to assist optimization practitioners to find algorithm settings which are approximate for the problem at hand. For a specified problem tMOPSO aims to determine multiple groups of control parameter values, each of which results in optimal performance at a different objective function evaluation budget. To achieve this, the control parameter tuning problem is formulated as a multi-objective optimization problem. Furthermore, tMOPSO uses a noise-handling strategy and control parameter value assessment procedure, which are specialized for tuning stochastic optimization algorithms. The principles upon which tMOPSO were designed are expanded into the context of many objective optimization, to create the MOTA tuning algorithm. MOTA tunes an optimization algorithm to multiple problems over a range of objective function evaluation budgets. To optimize the resulting many objective tuning problem, MOTA makes use of bi-objective decomposition. The last section of work entails an application of the tMOPSO and MOTA algorithms to benchmark optimization algorithms according to their tunability. Benchmarking via tunability is shown to be an effective approach for comparing optimization algorithms, where the various control parameter choices available to an optimization practitioner are included into the benchmarking process.

Acknowledgements

When I began my masters, I was adamant that I was not going to do a PhD. However as my masters progressed, questions arose beyond the scope of that masters. Questions, which I could not leave alone. So I enrolled for my PhD, and began to investigate the intricacies of engineering optimization which had so extensively captured my imagination. I would like to acknowledge the following individuals, who assisted me along my journey to this PhD:

My Supervisors P. Stephan Heyns and Schalk Kok

My Friends In particular Wha Suck Lee, who deserves special recognition for the many hours of encouragement he bestowed upon me during the course of my PhD studies

My Parents Ian and Elise Dymond for their love and support, not only during my PhD studies but throughout my entire life.

Formal acknowledgements are as follows; The South African center for high performance computing (CHPC) and the high performance computing centre (HPCC) of the Department of Electrical, Electronic and Computer Engineering (EECE) at the University of Pretoria are acknowledged for the computing resources they made available for this research. The financial assistance of the National Research Foundation (NRF) of South Africa towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

List of Publications

Conference publications:

- Dymond, A.S.D., Engelbrecht, A.P., and Heyns, P.S. (2011). The sensitivity of single objective optimization algorithm control parameter values under different computational constraints. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1412–1419. IEEE.
- Dymond, A.S.D., Kok., S., and Heyns, P.S. (2013). The sensitivity of multi-objective optimization algorithm performance to objective function evaluation budgets. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1868–1875. IEEE.

Journal publication:

- Dymond, A.S.D., Engelbrecht, A.P., Kok., S., and Heyns, P.S. (2014). Tuning optimization algorithms under multiple objective function evaluation budgets. *IEEE Transactions on Evolutionary Computation*, Accepted for publication on 11 April 2014.

Submissions in progress:

- Chapter 3 from this thesis has been submitted to the *MIT Press Journal on Evolutionary Computation* under the title 'Many Objective Tuning using Bi-objective Decomposition', and is under review.
- Chapter 4 from this thesis has been submitted to the *MIT Press Journal on Evolutionary Computation* under the title 'Benchmarking Optimization Algorithms According to Their Tunability', and is under review.

CONTENTS

LIST OF SYMBOLS

α	confidence level for pre-emptively terminating resampling
β	OFE budget
β^+	assessment run OFE budget
β_δ	MOTA OFE budget perturbation factor
$\beta_{\max}$	maximum OFE budget to tune under
Δ_{n_s}	pre-emptively terminating resampling increments
ϵ	solution error in terms of difference of objective function values
$\hat{\epsilon}$	normalized ϵ
γ	number of objective function calls made by the algorithm being tuned
λ	target OFE budget overshoot factor
μ_f	parent selection fraction
ω	inertia factor
σ_r	ratio of maximum step size to the search initialization bounds
τ	Hypervolume of a bi-objective preference articulation in the normalized objective space
B	OFE budgets to tune under
$\mathbf{b}^L$	decision variable lower bounds
$\mathbf{b}^U$	decision variable upper bounds
c_β	tMOPSO OFE perturbation factor
c_g	global acceleration constant

c_p	personal acceleration constant
CPV	control parameter value
C_r	crossover probability
EGO	efficient global optimization algorithm
$\mathbf{F}$	multi-objective function
F	scaling factor
FBM	flexible budget method
$\mathbf{F}^n$	multi-objective Nadir point
$\mathbf{F}^u$	multi-objective Ideal point
$\mathbf{g}$	inequality constraints vector
$\mathbf{h}$	equality constraints vector
I/F-race	iterated Friedman race algorithm
I_f	I/F-race sample size at which Friedman tests begin
I_i	I/F-race number of new CPV tuples assessed at each iteration
I_r	I/F-race search bound reduction ratio
$\mathbf{I}^L$	lower initialization bound
$\mathbf{I}^U$	upper initialization bound
MOTA	many objective tuning algorithm
m_s	FBM mutation strength
N	population or swarm size
n_f	number of objectives
n_g	number of inequality constraints
n_h	number of equality constraints
N_O	number of points used to fit EGO's first model
n_s	number of resampling runs
n_u	number of utility indicators to tune under multiple OFE budgets
n_x	number of decision variables
OFE	objective function evaluation

REVAC	relevance estimation and value calibration algorithm
R_i	REVAC initial number of CPV tuples
R_p	REVAC parent fraction
SPO	sequential parameter optimization algorithm
S_k	SPO number of data points used to fit the Kriging model
S_n	SPO number of new CPV tuples assessed at each iteration
S_o	SPO optimal computing budget allocation
tMOPSO	tuning multiple objective particle optimization algorithm
$\mathbf{u}$	multi-objective utility metric
$\mathbf{x}$	decision variable vector
$\mathbf{x}_c$	candidate decision variable vector
$\mathbf{y}$	CPV tuple

CHAPTER 1

INTRODUCTION

Numerical optimization forms a pivotal part of many design processes. The optimization process can be broadly broken up into the three parts of modeling, searching for the optimum of the generated model, and validation. Searching for the optimum of the generated model, or solving the optimization problem, is the aspect of numerical optimization which is the focus of this thesis.

To solve an optimization problem, a practitioner often applies an optimization algorithm to search for the optimal design(s) or decision vector(s). Numerous factors need to be considered when selecting an optimization algorithm and that algorithm's control parameter values (CPVs). These factors include the characteristics of the objective function of the optimization problem (Wolpert and Macready, 1997), the constraints imposed, and the termination criteria used (Dymond et al., 2011). The characteristics of the objective function, which refer to properties such as dimensionality, degree of multi-modality, scaling, and noise presence, need to be considered as search mechanisms which are useful for certain characteristics are detrimental for others. Sensitivity to termination criteria, which typically is imposed in the form of an objective function evaluation (OFE) budget, warrants consideration since depending on the application, OFE budgets vary widely. Moreover, OFE budgets are influenced by the computational cost of an OFE, and the computational resources available to the practitioner, where computational resources consist of computing power multiplied by the computing time. For success at solving an optimization problem, an optimization practitioner therefore needs to select an optimization algorithm and CPVs which are well suited to the objective function, constraints, and termination criteria of the problem at hand.

Selecting an appropriate optimization algorithm and CPVs is not a trivial task, however. Optimization algorithms and their default CPVs are typically benchmarked on standardized problems. These problems, although exhibiting challenging and varying numerical characteristics, are not necessarily representative of the optimization problem a practitioner is engaged with. If possible therefore, an optimization practitioner should rather aim to use an automated algorithm configuration (López-Ibáñez and Stützle, 2012) approach, as to determine CPVs which work well on testing problems representative of the problem at hand. Specifically,

representative testing problems have similar numerical characteristics to the problem to be tackled (in terms of dimensionality, level of modality, etcetera), have similar constraints imposed, and are numerically cheap. Central to automated algorithm configuration and other tools for performing CPV studies, is the use of tuning algorithms. Motivated by the desire to aid optimization practitioners in the task of selecting an optimization algorithm and CPVs which are appropriate for the problem they are engaged with, two new tuning algorithms are proposed.

The outline of this thesis is as follows: Chapter 2 presents a new algorithm (tMOPSO) for tuning the control parameter values of stochastic optimization algorithms under a range of OFE budget constraints. Specifically, for a given problem, tMOPSO aims to determine multiple groups of control parameter values, each of which results in optimal performance at a different OFE budget. To achieve this, the control parameter tuning problem is formulated as a multi-objective optimization problem. Additionally, tMOPSO uses a noise-handling strategy and control parameter value assessment procedure, which are specialized for tuning stochastic optimization algorithms.

In Chapter 3, another new tuning algorithm named MOTA is proposed. MOTA tunes an optimization algorithm to multiple performance measures over a range of OFE budgets. The tuning formulation used by MOTA consists of a decision vector comprised of CPVs together with an auxiliary decision variable which controls the OFE budget used to assess those CPVs. These decision variables are optimized for multiple objectives, consisting of one objective for each performance measure used, together with a speed objective. To optimize the resulting many objective tuning problem, MOTA makes use of bi-objective decomposition.

In Chapter 4, an investigation is conducted into benchmarking optimization algorithms according to their tunability. Benchmarking is complicated by many factors such as the sensitivity of optimization search processes to the objective function, constraint function, and termination criteria used for the problem being tackled. Additionally, the effects of an optimization algorithm's CPVs must be considered, as to discern if the performance measured is an artifact of the CPVs chosen, or actually representative of the optimization algorithm itself. Given these considerations, an investigation is conducted into benchmarking optimization algorithms according to their tunability to differing problem characteristics and termination criteria.

Chapter 5 then concludes this thesis, with chapter specific- and general conclusions, together with recommendations for future work.

CHAPTER 2

TUNING OPTIMIZATION ALGORITHMS UNDER MULTIPLE OBJECTIVE FUNCTION EVALUATION BUDGETS

Optimization practitioners often refer to parameter tuning studies when selecting an algorithm's CPVs. These studies are usually restricted to a single OFE budget constraint. This is problematic since the performance of optimization algorithms is dependent not only on the problem's fitness landscape (Wolpert and Macready, 1997; Malan and Engelbrecht, 2009), but also on the OFE budget available to explore that landscape. The sensitivity of control parameter tuning to the OFE budget under which the algorithm is tuned has been directly investigated in Dymond et al. (2011). For the sensitivity investigation in Dymond et al. (2011) multiple single-objective tuning problems were solved, each of which tuned selected optimization algorithms under a different OFE budget. The solutions from these tuning problems showed that different CPV tuples were found to be optimal depending on the OFE budget under which the algorithm was tuned. Additionally, evidence was given that the greater the difference between the OFE budget under which the algorithm was tuned and the OFE budget used to assess that algorithm's performance, the poorer the relative performance. The sensitivity investigation of Dymond et al. (2011) does not prove that all single OFE budget CPV tuning is sensitive to the OFE budget under which an algorithm is tuned. The sensitivity investigation does, however, indicate that algorithms do exist for which control parameter tuning should be conducted under multiple OFE budget constraints.

Tuning an optimization algorithm under multiple OFE budget constraints could be achieved by setting up multiple tuning problems, each focused on a different OFE budget, as done in Dymond et al. (2011). However, solving multiple tuning problems is computationally wasteful as no information sharing occurs between these problems. More specifically, utilizing information from solutions to tuning problems that are focused on an OFE budget close to the budget under which an algorithm is being tuned should enhance tuning efficiency. This information is not

utilized if multiple independent tuning problems are used to tune an optimization algorithm under multiple OFE budgets.

In order to efficiently tune optimization algorithms under multiple OFE budgets, a new tuning algorithm named tMOPSO is proposed. tMOPSO directly incorporates sensitivity to OFE budgets into the tuning process through the use of multi-objective optimization. The first objective of the tuning formulation is to minimize the solution error obtained by the algorithm being tuned, or the cost function value if the problem optimum is unknown. The second objective of the tuning formulation is to minimize the number of OFEs required to determine that solution error. Furthermore, when tuning a stochastic algorithm, multiple sample runs are required to determine which CPV tuple results in best mean solution error, given a specified confidence level. For each of these sample runs, the solution error is calculated by running the algorithm being tuned, from initialization to the OFE budget at which the solution error is to be determined. As such, the solution error calculation actually provides information on solution errors obtained for a range of OFE budgets. tMOPSO exploits this information in conjunction with a noise-handling strategy which uses Mann-Whitney U tests, in order to efficiently tune stochastic algorithms to multiple OFE budgets.

This chapter's outline is as follows: related work is discussed in Section 2.1, after which the proposed tuning algorithm is described in Section 2.2. Then the numerical setup used to gauge the effectiveness of tMOPSO is given in Section 2.3, followed by the numerical results in Section 2.4.

2.1 Related Work

Control parameter tuning entails finding which algorithm's CPVs are optimal according to a specified utility metric (Smit and Eiben, 2010b). The chosen utility metric measures a specified aspect of the performance of the algorithm being tuned. To help distinguish between the different parts of the control parameter tuning process, a three-layered hierarchy can be used (Smit and Eiben, 2009), where:

- the application layer refers to the problem instance(s) used in CPV tuple utility calculation,
- the algorithm layer refers to the optimization algorithm being tuned, and
- the design layer refers to the tuning algorithm used.

Using this terminology the tuning process can be described as follows: the design layer optimizes the algorithm layer to the application layer according to the specified utility measure. As such, the CPVs produced by the tuning process depend on these three layers as well as the utility metric used.

An extension of control parameter tuning is automated algorithm configuration (López-Ibáñez and Stützle, 2012). Automated algorithm configuration aims to improve an algorithm's performance on an unseen problem, by tuning that algorithm to a set of training problems believed to be representative of, or similar to, the unseen problem. The definition of a tuning

problem in automated algorithm configuration community is stricter than that of Smit and Eiben (2010b). In particular, evaluation on unseen problems is included into the definition as to guard against over tuning or over fitting. The definition used in this thesis is that of Smit and Eiben (2010b), since it allows for tuning to describe a tool used in automated algorithm configuration process and as a tool for performing CPV studies on well understood testing problems.

A sometimes mistakenly assumed alternative to control parameter tuning is to make use of adaptive optimization algorithms (Eiben et al., 1999). Adaptive algorithms adjust their CPVs through the course of an optimization run as to tune themselves to the problem being optimized. Superficially, such algorithms eliminate the need for control parameter tuning, since CPVs are tuned online typically by using feedback from the optimization process itself. However in reality, the practitioner's task simply changes from selecting approximate CPVs to selecting appropriate parameter control strategies for the problem at hand (Pedersen and Chipperfield, 2008). Moreover, since parameter control strategies can be expressed parametrically, the task of selecting appropriate control strategies can be expressed as a control parameter tuning problem itself.

One of the first examples of control parameter tuning in evolutionary computation was using a genetic algorithm to improve another genetic algorithm's performance on five testing problems (Grefenstette, 1986). Since then, many other control parameter tuning algorithms have been proposed (Birattari et al., 2002; den Besten, 2004; Bartz-Beielstein et al., 2005; Nannen and Eiben, 2007; Hutter et al., 2009a,b; Wagner and Wessing, 2012; Smit et al., 2010), and numerous other tuning studies have been performed (Bartz-Beielstein et al., 2004; Smit and Eiben, 2010a; López-Ibáñez and Stützle, 2011; Yuan et al., 2011). The M-FETA algorithm (Smit et al., 2010), just like the proposed tMOPSO, is a multi-objective tuning algorithm. tMOPSO and M-FETA differ however, since M-FETA tunes an optimization algorithm to multiple problems each at a specified OFE budget, whereas tMOPSO tunes an algorithm to a single problem under multiple OFE budgets. Tuning to find anytime (Radulescu et al., 2013) CPVs which perform well on average over a range of OFE budgets has also been done. The proposed tMOPSO does not aim to find anytime CPV tuples but rather to find multiple CPV tuples each of which is optimal for a different OFE budget.

Tuning is normally computationally expensive since each utility calculation requires performing an optimization run of the algorithm being tuned using the CPV tuple being assessed. Even for cases where the application layer consists of computationally cheap problem instances a high computational cost can result, since typically each utility evaluation entails the optimization algorithm being tuned, calling the objective function(s) in the application layer thousands of times. Given the high computational cost of calculating the utility of CPV tuples, a strong focus among tuning algorithms is efficiency. Examples are the Relevance Estimation and VAlue Calibration (REVAC, Nannen and Eiben, 2007) and the Sequential Parameter Optimization (SPO, Bartz-Beielstein et al., 2005) tuning algorithms, both of which use surrogate models of the tuning problem's fitness landscape to enhance convergence towards promising CPVs.

In the context of tuning stochastic optimization algorithms, the computational cost of tuning can be further reduced through the use of an efficient noise-handling strategy. In the context

of tuning under a single OFE budget work has been done in the form of algorithms such as SPO, M-FETA and F-race (Balaprakash et al., 2007). To emphasize the advantage of using an efficient noise-handling strategy the mechanics of F-race are briefly discussed. F-race is an iterative approach in which an initial group of candidates race against one another. During each iteration, an additional sample run is generated for each candidate still in the race. After the sample runs are generated, a Friedman statistical test (Conover, 1999) is applied to determine which candidates should be eliminated from the race, given a specified confidence level. This early elimination process by F-race saves considerable computational resources, compared with first generating large samples for each candidate and then running statistical tests.

When tuning an optimization algorithm under multiple OFE budgets, tuning efficiency can be further increased by using the history information from the CPV tuple assessment optimization runs (Branke and Elomari, 2012). The Flexible Budget method (Branke and Elomari, 2012) incorporates this history information by assessing a CPV tuple according to the resulting OFEs made, versus solution accuracy achieved, curve. Assessing CPV tuples in this manner boosts tuning efficiency since one run of the algorithm being tuned is used to gauge performance at multiple OFE budgets. A limitation of the Flexible Budget method is that each CPV tuple being assessed is run up to the maximum OFE budget to which the algorithm is being tuned under, which is wasteful if the CPV tuple being assessed is effective at OFE budgets lower than that maximum OFE budget. Multi-objective tuning can be used to overcome this shortcoming.

Multi-objective tuning according to the conflicting criteria of speed versus accuracy (Dréo, 2008) was proposed in Dréo (2009). Dréo (2009) demonstrated the concept by setting up tuning problems, whereby the algorithm being tuned would terminate if stagnation occurs and no improvement was made over the last 100 OFEs. The mean runtime and the mean accuracy achieved as a function of the CPV tuple chosen were then used as the tuning objectives, which were optimized using the NSGA-II algorithm (Deb et al., 2002). The Dréo (2009) proof-of-concept algorithm, although similar to, is not a multiple OFE budget tuning algorithm. Specifically, the Dréo (2009) proof-of-concept algorithm can be used to tune an algorithm as to determine CPV tuples each which results in optimal mean accuracy given an average OFE usage. Given that termination due to a lack of improvement occurs at differing OFE usages, this mean solution accuracy achieved cannot be compared to the mean solution accuracy achieved given an OFE budget constraint. Dréo (2009) did however introduce the idea of tuning according to speed versus accuracy, an idea which the tMOPSO algorithm uses to tune under multiple OFE budgets.

The contribution of this chapter is to combine the aforementioned concepts in one algorithm. tMOPSO uses multi-objective optimization as to directly incorporate sensitivity to OFE budgets into the tuning problem formulation, by using the speed and accuracy objectives. However, unlike Dréo (2009) no stagnation termination criterion is added to the CPV tuple assessment runs, but rather an OFE budget which is controlled through a decision variable which is optimized by tMOPSO. In addition, tMOPSO uses the history information from the utility calculations to enhance tuning efficiency. These concepts are combined with a noise handling strategy, as to further increase efficiency when tuning stochastic algorithms.

Before tMOPSO is presented, the prerequisite multi-objective optimization concepts and nomenclature are given. Solving a constrained multi-objective minimization problem (Engelbrecht, 2007) entails determining the decision vectors $\mathbf{x}$ that minimize an objective function $\mathbf{F}$, subject to that problem's inequality and equality constraints $\mathbf{g}$ and $\mathbf{h}$ respectively. Formally a constrained multi-objective minimization problem is defined as:

$$minimize\ \mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_{n_f}(\mathbf{x}) \end{bmatrix} \tag{2.1}$$

$$subject\ to\ g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, n_g \tag{2.2}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \dots, n_h \tag{2.3}$$

$$x_k \in [b_k^L, b_k^U] \quad k = 1, \dots, n_x, \tag{2.4}$$

where $f_1, f_2, \dots, f_{n_f}$ are the conflicting sub-objectives, n_x is the dimensionality of $\mathbf{x}$, n_g is the number of inequality constraints, n_h is the number of equality constraints and $\mathbf{b}^L$ and $\mathbf{b}^U$ define the search bounds.

Multi-objective problems have multiple solutions, each of which is optimal for a different trade-off among the conflicting sub-objectives. Multi-objective algorithms commonly use the principle of Pareto dominance to identify these optimal solutions. According to Pareto dominance, a decision vector $\mathbf{x}_1$ dominates another vector $\mathbf{x}_2$ ($\mathbf{x}_1 \prec \mathbf{x}_2$), when $\mathbf{x}_1$ is better or equal in all objectives while being better in at least one objective. For minimization problems, $\mathbf{x}_1 \prec \mathbf{x}_2$ when:

$$f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2), \forall\, k \in 1, 2, \dots, n_f, \tag{2.5}$$

and

$$\exists k \in 1, 2, \dots, n_f : f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2). \tag{2.6}$$

The set of all Pareto non-dominated decision vectors for a multi-objective optimization problem is referred to as the Pareto-optimal set (PS), while the set of objective function values corresponding to the PS is referred to as the Pareto-optimal front (PF). Since the PS often consists of infinite points, multi-objective evolutionary algorithms typically aim to determine a finite set of non-dominated decision vectors that accurately approximate the PF. Two special points in the objective space which are commonly used by multi-objective optimization algorithms, are the Ideal point $\mathbf{F}^u$ and the Nadir point $\mathbf{F}^n$, where for minimization problems $F_i^u = \min\{F_i \,\forall\, \mathbf{F} \in \text{PF}\}$ and $F_i^n = \max\{F_i \,\forall\, \mathbf{F} \in \text{PF}\}$.

2.2 Proposed Tuning Algorithm

In this section the key concepts behind the proposed algorithm tMOPSO are first presented, followed by a complete outline of the algorithm. The first concept presented entails incorporating

CPV sensitivity to OFE budgets into a tuning problem formulation, through the use of a multi-objective utility metric.

When tuning stochastic algorithms such as evolutionary or swarm intelligence optimization algorithms, utility metrics based on mean performance values are typically used. Two examples of commonly used single-objective utility metrics are:

1. the mean solution error or mean cost function value obtained after a fixed number of OFEs, and
2. the mean number of OFEs required to solve a problem within a specified solution error tolerance.

These single objective utility metrics require that either the desired solution tolerance or the desired OFE budget be known before tuning is conducted. Depending on the choice of solution tolerances or OFE budget, different CPVs are optimal for the tuning problem, as illustrated in Figure 2.1(a).

To quantify a CPV tuple's utility in a general sense without presupposing a solution tolerance or OFE budget, a multi-objective utility metric (Dréo, 2008) is used. The first criterion is the mean solution error obtained, or the mean cost function value if the optimum of the application layer's problem is unknown. The second criterion is the number of OFEs required to obtain that solution error. This multi-objective utility metric $\mathbf{u}$, as a function of the CPVs being assessed, $\mathbf{y}$, and the OFE budget used for that assessment, β, is:

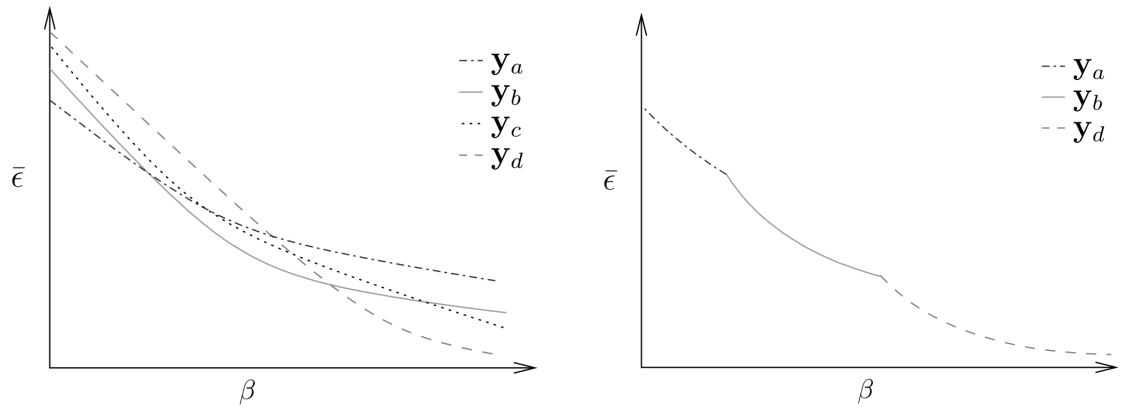
$$\mathbf{u}(\mathbf{y}, \beta) = \begin{bmatrix} \bar{\epsilon}(\mathbf{y}, \beta) \\ \beta \end{bmatrix}, \tag{2.7}$$

where $\bar{\epsilon}(\mathbf{y}, \beta)$ is the mean solution error obtained by the algorithm being tuned on the application layer as a function of $\mathbf{y}$ and β. The PF of a multi-objective tuning problem formulated with $\mathbf{u}$ as its utility metric contains multiple CPV tuples, each of which is optimal for different solution tolerances or OFE budgets, as illustrated in Figure 2.1(b). As such, $\mathbf{u}$ successfully captures the conflicting needs of the practitioner who wants both a quick and an accurate solution to the optimization problem at hand.

Using the $\mathbf{u}$ utility metric, the multi-objective problem formulation used by tMOPSO for tuning control parameter values under multiple OFE budget constraints is formally defined as: Determine $\mathbf{y}$ and β as to:

$$\begin{aligned} &\text{minimize} && \mathbf{u}(\mathbf{y}, \beta) \\ &\text{subject to} && 0 < \beta \leq \beta_{\max} \\ & && g_i(\mathbf{y}) \leq 0, \quad i = 1, \ldots, n_g, \end{aligned} \tag{2.8}$$

where $\beta_{\max}$ denotes the largest OFE budget of interest, and g_i represents each of the CPV inequality constraints. Contrary to the constrained multi-objective definition presented earlier, the multi-objective tuning problem formulation is not necessarily bound constrained for every decision variable. This constraint relaxation is motivated by the fact that some CPV bounds are difficult to determine before tuning. Consider, for example, specifying bounds for the population



(a) The mean solution error obtained $\bar{\epsilon}$ versus the OFE budget available β.

(b) The non-dominated CPVs found when using a multi-objective utility metric.

Figure 2.1: Illustration of why single objective utility metrics require a priori knowledge regarding the OFE budget or solution accuracy before tuning is applied, and how using a multi-objective utility metric overcomes this limitation. The performances of four CPV tuples namely, $\mathbf{y}_a$, $\mathbf{y}_b$, $\mathbf{y}_c$ and $\mathbf{y}_d$ are compared in this example. The Pareto-optimal front indicates that for low OFE budget applications $\mathbf{y}_a$ is a good choice, $\mathbf{y}_b$ is effective for intermediate OFE budgets and $\mathbf{y}_d$ is effective at high OFE budgets. $\mathbf{y}_c$ should not be used as the CPV tuple is dominated for all OFE budgets.

size parameter of an evolutionary algorithm. Although it is clear that this population size should be a positive integer, it is less obvious what a sensible maximum size should be.

The proposed tuning formulation can be solved by any standard multi-objective optimization algorithm, provided that the mean solution error obtained can be determined analytically. However, this is normally not the case when tuning stochastic algorithms, as analytical expressions for determining the solution error obtained as a function of the CPVs and OFE budget are often unavailable. As such, using a sample of multiple independent runs as to approximate the mean solution error is often the only viable choice. Approximating the mean solution error in this manner is troublesome, however, as these approximations introduce noise into the fitness landscape of the tuning problem.

2.2.1 Handling the Noise Resulting from Tuning Stochastic Algorithms

tMOPSO employs a noise-handling strategy tailored for tuning stochastic optimization algorithms, for which the mean solution error objective needs to be approximated numerically. For these cases, noise is induced on the first objective of $\mathbf{u}$, which is to minimize the solution error objective, while the second objective of $\mathbf{u}$, which is to minimize the number of OFEs used, remains noise free. Given that the first objective of $\mathbf{u}$ is solution error based, its distribution has the following properties:

- The mean of the distribution decreases for Pareto-optimal decision vectors as the OFE budget available increases.
- A probability density of zero for negative solution error values, since it is impossible to get a solution error less than zero.

Based on these properties, it is reasonable to assume that for the majority of tuning applications, the variance of the solution error obtained decreases for Pareto-optimal decision vectors as the

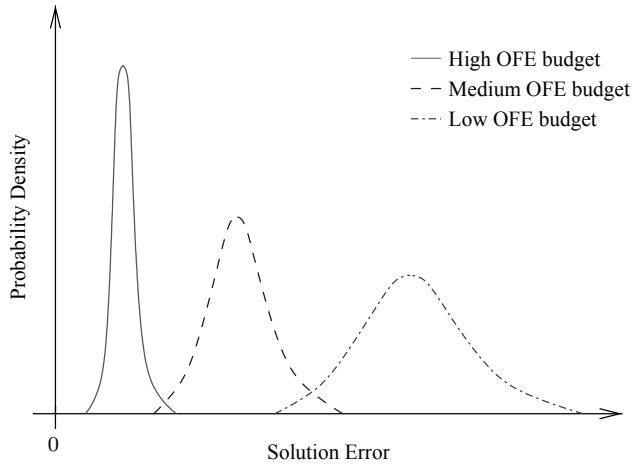


Figure 2.2: Illustration of the expected decrease in solution error mean and variance as the available OFE budget increases, for CPVs close to the Pareto-optimal front.

OFE budget increases, as illustrated in Figure 2.2.

The noise characteristics of the tuning problem prevent the use of many already established multi-objective optimization algorithms which are designed for noisy environments, as the suspected noise characteristics violate the assumptions upon which these algorithms are based. More specifically, many of these algorithms assume that noise variance and distribution are the same throughout the objective space (Bui et al., 2009). Also, although there are established multi-objective algorithms such as methods based on local models (Bui et al., 2009), which are designed to handle varying noise distributions, it is not clear how these algorithms can be modified to incorporate the enhancements which are described in the next subsection. As such, it was decided to rather use noise-handling techniques used by single objective tuning algorithms and extend them into the context of the multi-objective tuning formulation proposed.

tMOPSO's noise-handling strategy is based upon the resampling strategy (Beyer, 2000) which is commonly used by single objective tuning algorithms. The resampling strategy entails using a standard optimization algorithm, designed for static environments without noise, to search a noise-reduced version of the original problem. The noise strength is reduced by evaluating each decision vector n_s times and returning its approximated mean function value. Since the noise magnitude is reduced, the resampling strategy improves the performance of non-noisy optimization algorithms on noisy optimization problems. However, since the strategy decreases the noise strength by a factor of $\sqrt{n_s}$ (Bui et al., 2009), the strategy cannot completely eliminate noise. Another significant disadvantage of the resampling strategy is that it is expensive, with the cost of each decision vector evaluation multiplied by a factor of n_s. Although little can be done regarding the resampling strategy's inability to completely eliminate noise, the computational cost associated with the method can be decreased significantly.

A more efficient alternative to the standard resampling is to make use of a pre-emptively terminating resampling strategy. Pre-emptively terminating resampling strategies work on the basis that evaluating each decision vector n_s times is often unnecessary. Statistical tests can be used during the sample gathering process to determine the likelihood of the decision vector being assessed, being an improvement on the decision vectors already assessed. If the decision vector

being assessed is unlikely to yield any improvement, the sample gathering process is interrupted to save computational resources. Tuning algorithms which use single objective utility measures often employ racing (Maron and Moore, 1997) or racing equivalent (Pedersen, 2010) methods in order to achieve pre-emptively terminating resampling.

tMOPSO employs a pre-emptively terminating resampling strategy which uses the Mann-Whitney U test (MWUT, Conover, 1999). Specifically, at user-specified sampling intervals (Δ_{n_s}), a MWUT is conducted to determine if the difference between the mean of the CPV tuple currently being assessed and the mean values of the CPV tuples already assessed is statistically significant. If the MWUT shows, with a confidence level of α, that the CPV tuple being assessed is worse than the CPV tuples already assessed, then the sample gathering process is interrupted to save computational resources. tMOPSO users can select different values for α and Δ_{n_s}, depending on how aggressively or conservatively resampling interruption should take place. A high α reduces the risk of good CPV tuples being discarded, but also increases computational resources spent on bad CPV tuples. By contrast, a low α increases the likelihood of good CPV tuples being mistakenly discarded, but the computational resources saved through using a low α allow more CPV tuples to be assessed.

Pre-emptively terminating resampling in the context of a general multi-objective noisy environment can be achieved by checking the decision vector being assessed against the decision vectors in the current approximation of the PF. The sampling gathering process is interrupted if there is a decision vector in the current PF approximation, which for all objective space dimensions has a sample mean value which is better than that of the decision vector currently being assessed, given a specified confidence level. Formally for minimization problems, the sample gathering process is interrupted for a candidate decision vector $\mathbf{x}_c$, if there is a decision vector in the current approximation of the PF $\mathbf{x}_p$, such that:

$$\tilde{F}(\mathbf{x}_p)_k \leq_\alpha \tilde{F}(\mathbf{x}_c)_k \qquad \forall k \in 1, 2, \ldots, n_f, \tag{2.9}$$

where $\tilde{F}(\mathbf{x})_k$ are the k'th components of the objective values from the sample of independent runs of $\mathbf{x}$. The $\leq_\alpha$ operator indicates if the sample mean of $\tilde{F}(\mathbf{x}_1)_k$ is less than the sample mean $\tilde{F}(\mathbf{x}_2)_k$ with a confidence level greater than or equal to α according to the selected statistical test. In the context of the bi-objective function which tMOPSO optimizes, sample gathering of a CPV tuple $\mathbf{y}_1$ assessed at OFE budget of β_1 is interrupted when another CPV tuple $\mathbf{y}_2$, assessed at an OFE budget of β_2 exists in the current approximation of the PF such that:

$$\beta_2 \leq \beta_1 \tag{2.10}$$

and

$$\tilde{\epsilon}(\mathbf{y}_2, \beta_2) \leq_\alpha \tilde{\epsilon}(\mathbf{y}_1, \beta_1), \tag{2.11}$$

where $\tilde{\epsilon}(\mathbf{y}, \beta)$ denotes the sample of solution errors obtained during the independent runs of the algorithm being tuned when using $\mathbf{y}$ CPVs with an OFE budget of β.

2.2.2 Specialization for Tuning Stochastic Algorithms under Multiple Objective Function Evaluation Budgets

Approximating solution errors through numerical experimentation provides additional information which can be exploited when tuning an algorithm for multiple OFE budgets. In order to calculate the solution error obtained by a stochastic algorithm for an OFE budget of β, the algorithm being tuned is run from zero to β OFEs using the specified CPVs and random state. This method of calculation therefore also provides the solution errors obtained for OFE budgets of less than β. For example, when calculating the utility metric $\mathbf{u}(\mathbf{y}, \beta)$ for an evolutionary optimization algorithm with a population of size N, the solution error calculation also provides information on:

$$\mathbf{u}(\mathbf{y}, N) \rightarrow \mathbf{u}(\mathbf{y}, 2N) \rightarrow \cdots \rightarrow \mathbf{u}(\mathbf{y}, \beta - N). \tag{2.12}$$

This information is used by tMOPSO to enhance tuning efficiency. Another appealing aspect of using this additional information is that it accommodates the scenario when the algorithm being tuned terminates due to a stopping criterion other than reaching the OFE budget. This accommodation happens naturally, since the utility values for an OFE budget less than the number of OFE at termination are available. Furthermore, the manner in which solution errors are calculated can be exploited to provide solution errors for OFE budgets higher than β, at a reduced cost, since the calculations need not start from zero OFEs again, but rather simply continue from β OFEs.

Exploiting the additional information from the solution error calculations has a drawback in terms of increasing the computational overhead. Increased overhead becomes detrimental when tuning an optimization algorithm with low computational overhead to a cheap optimization problem, in which case the majority of computational resources are spent on internal overhead for the tuning algorithm, instead of assessing new CPV tuples. For such a scenario, the best option would be to utilize only some of the information from the solution error calculation. The converse is also true when tuning an algorithm with high computational overhead, in which case all the information from the solution error calculation can be parsed without significantly detracting resources from assessing new CPV tuples. Due to computational overhead considerations, a control parameter B is introduced to specify on which OFE budgets tMOPSO should focus on. The tMOPSO method therefore calculates the following CPV utility metrics when assessing a CPV tuple:

$$\mathbf{u}(\mathbf{y}, b_i) \, \forall \, b_i \in B : b_i \leq \beta' \tag{2.13}$$

where the upper OFE budget β' is specified by the tMOPSO CPV assessment procedure. If B is not set, all OFE budgets up to $\beta_{\max}$ are focused on and all the solution error information as presented in (2.12) is used.

The tMOPSO CPV assessment procedure increases tuning efficiency by exploiting the additional information from the solution error calculations, in conjunction with the pre-emptively terminating resampling strategy described in (2.10)-(2.11). Given a set of candidate groups of CPVs Y, the utility values from (2.13) are first roughly approximated for each $\mathbf{y} \in Y$. This rough approximation entails using the small initial sample size specified by Δ_{n_s}, and a target

OFE budget of:

$$\beta' = \min(\lambda \cdot \beta,\ \beta_{max}) \tag{2.14}$$

where λ is the target OFE budget overshoot factor, which is a user-specified control parameter value for tMOPSO. The value of λ only affects the initial value of β' for each $\mathbf{y}$, after which the tMOPSO noise-handling strategy is used to adjust β'.

After the initial samples are generated, tMOPSO's CPV assessment procedure then conducts resampling interruption checks as to determine which of the $\mathbf{u}$ values from (2.13) should be refined further. These interruption checks are conducted against the utility measure approximations in the current PF approximation. If it is the first iteration of tMOPSO and no PF approximation exists, then the interruption checks are conducted using the other utility value approximations currently being refined. According to the results from resampling interruption checks, β' is reduced to match the largest OFE budget at which each CPV tuple $\mathbf{y}$ ($\mathbf{y} \in Y$) may be effective. Reducing β' results in a large saving of computational resources, especially where the CPVs being assessed are effective at OFE budgets far lower than the original β' value. This refinement of β' is repeated multiple times according to the step increments specified by Δ_{n_s}. After the sampling loop is completed, the utility values which reached the required resampling sample size are used to update the PF approximation of the tuning problem at hand. The tMOPSO CPV tuple assessment procedure is summarized in Figure 2.3.

In order to efficiently perform the resampling interruption checks and update PF approximations, tMOPSO exploits the bi-objective nature of the proposed utility metric as detailed in the next subsection.

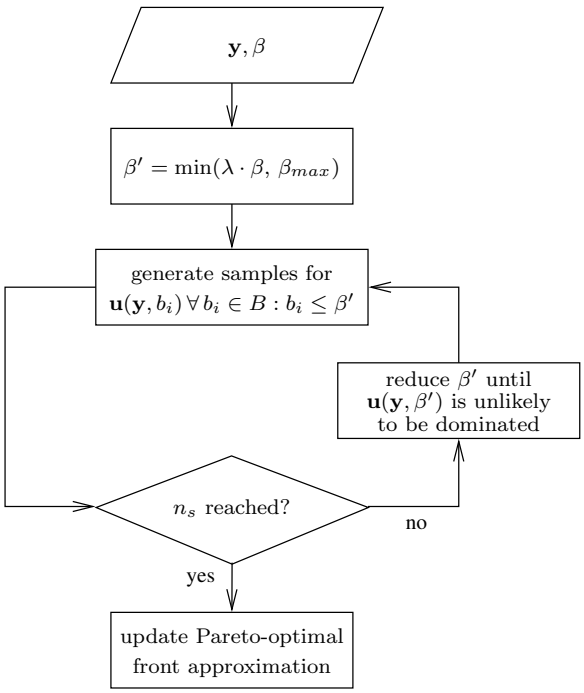


Figure 2.3: Flow chart of tMOPSO's CPV tuple assessment procedure.

2.2.3 Performing Resampling Interruption Checks and Pareto-optimal Front Approximation Updates

Most multi-objective optimization algorithms maintain a set of relatively non-dominated decision vectors. This set is updated during the optimization process, and is often used to store the approximation of the PF, as is the case with tMOPSO. The object responsible for updating and storing the non-dominated decision vector set is traditionally referred to as the Pareto archive or repository. Conventional archives update or maintain the non-dominated decision vector set through the use of the linear list approach (Mostaghim and Teich, 2005), as follows: a candidate decision vector $\mathbf{x}_c$ is added to the archive when it is not dominated by any decision vector in that archive. Additionally, if $\mathbf{x}_c$ is added, then the decision vectors in the archive need to be checked against $\mathbf{x}_c$, with all decision vectors dominated by $\mathbf{x}_c$ being discarded. Formally,

$$\text{add } \mathbf{x}_c \text{ to } A \text{ if } \nexists \mathbf{x} \prec \mathbf{x}_c \, \forall \mathbf{x} \in A, \tag{2.15}$$

$$\text{discard from } A \text{ all } \mathbf{x} \text{ where } \mathbf{x}_c \prec \mathbf{x}, \tag{2.16}$$

where A is the set of relatively non-dominated decision vectors stored by the archive.

For the linear list approach updating a set of relatively non-dominated decision vectors is of computational complexity $\mathcal{O}(|A|)$ for each decision vector inspection, where $|A|$ is the number of decision vectors stored in A. The resulting computational overhead of the linear list approach is too high for tMOPSO's CPV tuple assessment procedure, and would greatly reduce the number of OFE budgets which tMOPSO can tune under, i.e. $|A|$. To allow tMOPSO to focus on a large number of OFE budgets, a fast-checking archive capable of determining dominance statuses and dominance likelihood statuses faster than $\mathcal{O}(|A|)$ is required.

The fast-checking archive used by tMOPSO is based on the work in Berry and Vamplew (2006). For bi-objective minimization problems, a candidate decision vector $\mathbf{x}_c$ is non-dominated by any decision vector in A, when it is not dominated by its neighboring decision vector $\mathbf{x}_n$. Here $\mathbf{x}_n$ is the decision vector in A with the smallest improvement relative to $\mathbf{x}_c$, according to the minimize OFE budget objective. Formally, for bi-objective minimization problems,

$$\mathbf{x}_c \not\prec A \text{ if } (\mathbf{x}_c \not\prec \mathbf{x}_n), \tag{2.17}$$

where $\mathbf{x}_n$ has the property

$$f_2(\mathbf{x}_n) = \max\big\{ f_2(\mathbf{x}_c) - f_2(\mathbf{x}) \, \forall \, \mathbf{x} \in A : f_2(\mathbf{x}_c) - f_2(\mathbf{x}) \leq 0 \big\}. \tag{2.18}$$

The approach of Berry and Vamplew (2006) therefore results in a significant reduction in computational requirements compared to the linear list approach, since the number of Pareto dominance checks required is reduced from $|A|$ to only one.

Resampling interruption checks can also be sped up using the bi-objective property described in (2.17)-(2.18) due to the noise characteristics of tMOPSO's utility measure. In particular, since the second objective of the utility measure used is the OFE budget allocated, which does not change during the resampling process, $\mathbf{x}_n$ is known without having to wait for the

resampling process to complete. Which decision vector the candidate decision vector is going to be compared with to determine its dominance status against the entire bi-objective front is therefore known. As such, only one dominance likelihood check as described in (2.10)-(2.11) needs to be performed. This is significant, as it reduces the computational overhead drastically by decreasing the number of dominance likelihood checks from the size of the Pareto archive to only one.

Another aspect of Pareto archives that is relevant when tuning optimization algorithms for multiple OFE budgets is size limiting. Normally size limiting is required to keep the computational overhead of maintaining the archive down. The loss of some non-dominated vectors is often considered insubstantial, provided that the retained decision vectors are adequately spaced as to be able to represent the PF of the multi-objective problem being solved. Direct size limiting is not applied to the tMOPSOs archive, since the CPV assessment procedure will limit the archive to a size of B, or to the size of $\beta_{\max}$ if B is unspecified.

With all its core elements described, the complete tMOPSO algorithm is presented in the next subsection.

2.2.4 Tuning Optimization Algorithm

The tuning multi-objective particle swarm optimization (tMOPSO) algorithm is a particle swarm based algorithm (Engelbrecht, 2007). A number of multi-objective PSO variants have been presented (Coello et al., 2004; Sierra and Coello, 2005; Engelbrecht, 2007), indicating that PSO operations should be sufficient for purposes of tuning under multiple OFE budgets.

PSO algorithms explore the search space by utilizing a swarm of particles, where each particle's search is influenced by both a local and a global guide. Each particle's local guide is selected according to information which that particle has personally experienced, while the global guide is selected according to information that the particle's neighborhood has experienced. Many different neighborhood topologies exist, each resulting in a different information flow through the swarm. tMOPSO is a global best PSO where each particle's neighborhood spans the entire swarm.

When applied to single objective optimization problems, global best PSO algorithms need only store the personal best decision vectors that particles have explored and the global best decision vector the swarm has explored. The personal and global best values are updated after each search iteration according to the function values of the particle's new position in the search space. However, for multi-objective optimization where decision vectors can be relatively non-dominated, the personal best experienced by each particle and the global best experienced by the swarm cannot be fully captured without using Pareto non-dominated sets or Pareto archives. tMOPSO therefore also stores each particle's local approximation of the PF, in addition to having a Pareto archive to store the swarm's approximation of the PF. This approach is tractable from a computational overhead perspective since tMOPSO is designed for the presented bi-objective tuning formulation only, and can therefore use the fast-checking bi-objective archive described in the previous subsection to efficiently capture each particle's approximation of the PF. In the event that the pre-emptively terminating resampling approach interrupts the approximation of the utility values, the mean values from the interrupted samples

are used to update the local approximations of the PF.

In order to search for the solutions to the control parameter tuning problem formulation presented in (2.8), tMOPSO uses a decision vector of the following form:

$$\mathbf{x} = \begin{bmatrix} \ln \beta \\ y_1 \\ y_2 \\ \vdots \\ y_{n_y} \end{bmatrix}, \tag{2.19}$$

where $y_1, y_2, \ldots, y_{n_y}$ denote the CPVs being tuned, and β is the OFE budget allocated to those CPVs. The natural logarithm of β is used by tMOPSO in the decision variable definition, as this transformation helps to improve the scaling of the search space. Scaling of the search space is improved, as previous sensitivity studies such as Dymond et al. (2011) indicate that the optimal CPVs as a function OFE budget often follow logarithmic trends.

tMOPSO begins its search by assigning to each of the swarm's N particles a position that is generated randomly inside the search initialization bounds. Formally, the j'th particle's initial position decision vector $\mathbf{x}_0^j$, is generated as follows:

$$\mathbf{x}_0^j = \mathbf{I}^L + \mathbf{r}() \circ (\mathbf{I}^U - \mathbf{I}^L), \tag{2.20}$$

where $\mathbf{I}^L$ and $\mathbf{I}^U$ are the lower and upper initialization bounds respectively, $\circ$ is the Hadamard product operator, and $\mathbf{r}()$ is a function which returns a vector of dimension $|\mathbf{I}^U|$ whose components are each randomly generated between 0 and 1 using a uniform probability density distribution. After initialization, each particle's position for the i'th iteration is updated as:

$$\mathbf{x}_i^j = \mathbf{x}_{i-1}^j + \mathbf{v}_i^j, \tag{2.21}$$

where $\mathbf{v}_i^j$ is the j'th particle's velocity at iteration i, with each particle having a zero initial velocity. No position or velocity limiting is applied to tMOPSO particles. Instead, when the result from a particle's position update is invalid, the particle's velocity and position are recalculated until a valid solution is found. This approach, although normally undesirable for general constraint handling where constraint evaluations can be expensive, is acceptable here since tuning constraints are normally computationally cheap.

Traditionally in single objective PSO, the j'th particle's velocity is updated as:

$$\begin{aligned} \mathbf{v}_{i+1}^j = \omega \mathbf{v}_i^j &+ c_p\, \mathbf{r}() \circ \left(\mathbf{x}_p^j - \mathbf{x}_i^j\right) \\ &+ c_g\, \mathbf{r}() \circ \left(\mathbf{x}_g - \mathbf{x}_i^j\right), \end{aligned} \tag{2.22}$$

where ω is the inertia factor, c_p and c_g are the personal and global acceleration constants, $\mathbf{x}_p$ is the personal best decision vector and $\mathbf{x}_g$ is the global best decision vector of the swarm. However, in the context of tMOPSO where the swarm's global best and every particle's local best are PF approximations, additional heuristics are required to determine which decision vectors from these PF approximations to use in velocity updates.

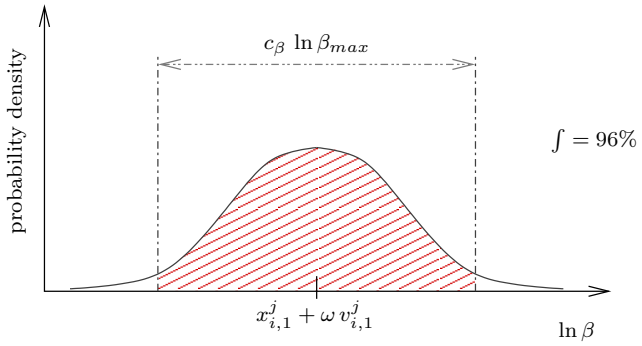


Figure 2.4: Probability density of the OFE budget value used to select the guiding decision vectors in tMOPSO's velocity update rule.

tMOPSO selects guiding vectors from the local and global PF approximations which perform well at OFE budgets close to a particle's expected future assessment OFE budget. The guide selection OFE budget β_g is calculated based on the particle's current position $x^j_{i,1}$ and velocity $v^j_{i,1}$ in the assessment OFE budget dimension as follows:

$$\ln \beta_g = x^j_{i,1} + \omega\, v^j_{i,1} + c_\beta\, r_g()\, \ln \beta_{max} \tag{2.23}$$

where c_β is the target OFE perturbation factor, β_{max} is the maximum OFE budget of interest and the $r_g()$ function returns a random scalar generated using a Gaussian distribution with a zero mean and a standard deviation of 0.25. c_β is a user-specified parameter which influences tMOPSO's behavior in terms of exploration versus exploitation. A near zero c_β would result in guides being selected which are effective at the expected future assessment OFE budget of a particle, and therefore favors exploitation. In contrast, a c_β close to one would result in guides being selected randomly from the entire local or global approximation of the PF, and hence favor exploration. Figure 2.4 shows the probability density function which is used to generate β_g.

Each particle's velocity is updated as follows:

$$\begin{aligned}\mathbf{v}^j_{i+1} = \omega \mathbf{v}^j_i &+ c_p\, \mathbf{r}() \circ \left(\mathbf{x}^j_p() - \mathbf{x}^j_i\right) \\ &+ c_g\, \mathbf{r}() \circ \left(\mathbf{x}_g() - \mathbf{x}^j_i\right) + \beta_c,\end{aligned} \tag{2.24}$$

where the $\mathbf{x}^j_p()$ and $\mathbf{x}_g()$ functions each return a non-dominated decision vector from the local and global approximations of the PF respectively, with each decision vector selected using its own β_g value. The β_c component of the tMOPSO velocity update rule is a velocity correction factor, which keeps the mean of the expected future OFE budget dimension value equal to $x^j_{i,\beta} + \omega \cdot v^j_{i,\beta}$. As such β_c is a zero vector of equal dimension to $\mathbf{x}$, with the exception of the first element of β_c, which is:

$$-0.5(c_p + c_g)\, \omega\, v^j_{i,1}. \tag{2.25}$$

The swarm continues to explore the search space using the position and velocity update rules, until the application layer evaluation budget $\gamma_{\max}$ is exhausted. A termination criterion

```
procedure tMOPSO
    Generate particles initial positions                          ▷ (2.20)
    assess generated CPV tuples                                   ▷ Subsection 2.2.1
    update local PF approximations
    i ← 1
    while γ < γ_max do                                            ▷ main loop
        for j ∈ {1, 2, ..., N} do
            repeat
                v_i^j                                             ▷ compute velocity (2.24)
                x_i^j                                             ▷ update position (2.21)
            until x_i^j valid
        end for
        assess generated CPV tuples                               ▷ Subsection 2.2.1
        update local PF approximations
        i ← i + 1
    end while
end procedure
```

Figure 2.5: tMOPSO pseudocode

based on γ which is the number of application layer evaluations made by the algorithm being tuned, is well suited for controlling the computational resources used during tuning where pre-emptively terminating resampling occurs. The pseudo-code for tMOPSO is given in Figure 2.5.

2.3 Numerical Setup

Numerical experiments are conducted to gauge tMOPSO's effectiveness at tuning optimization algorithms under multiple OFE budgets. In particular, tMOPSO is compared against other algorithms for tuning under multiple OFE budgets, as well as tuning algorithms focused on a single OFE budget. Comparison against the single OFE budget tuning algorithms aims to help determine if tMOPSO is a viable alternative for tuning under multiple OFE budgets, compared to setting up and solving multiple tuning problems, each focused on a different single OFE budget.

The comparison between tMOPSO and the single OFE budget tuning algorithms is conducted by comparing the best minimum solution error found for an OFE budget of β_{max}. For this comparison, the single OFE budget tuning algorithms are set up to find CPV tuples which result in the lowest solution error for an OFE budget of β_{max}, while tMOPSO is configured to determine CPV tuples for multiple OFE budgets all the way up to β_{max}. Even though tMOPSO focuses on multiple OFE budgets instead of only one budget, tMOPSO may still be competitive against the single OFE budget tuning algorithms. tMOPSO may be competitive since it has information on CPVs which work well at lower OFE budgets, which is made accessible cheaply via the additional history information from the numerical solution error calculations. As such, tMOPSO may still be comparable to methods focused on a single OFE budget, even though tMOPSO is tuning an optimization algorithm under multiple OFE budgets. If tMOPSO is comparable to the single OFE budget tuning methods in this manner, then by extension using

tMOPSO to tune under multiple OFE budgets should be a more efficient alternative then setting up multiple independent tuning runs each focused on a single OFE budget. Furthermore, tMOPSO would then be an example of using multiobjectivization to enhance performance on a single objective problem (Handl et al., 2008).

The outline of this section follows: The optimization problems used in the application layers are described first. Then the algorithms tuned under multiple OFE budgets are presented, followed by the description of the tuning algorithms that tMOPSO will be compared to.

2.3.1 Application Layers

Selected optimization algorithms are tuned to problems from the CEC 2005 special session on real-parameter optimization (Suganthan et al., 2005). The CEC problems are unconstrained, real-valued, and static noise-free single-objective minimization problems, except for problems 4 and 17 which have noise added to their objective function values. To solve the single objective CEC'05 problems, an optimization algorithm needs to determine the decision vector $\mathbf{x}$ which minimizes a scalar objective function $f(\mathbf{x})$, where $\mathbf{x} \in S \subseteq \mathbb{R}^n$, and n is the dimension of the search space. The CEC'05 competition problems were chosen because they are commonly used in the literature (Wang et al., 2011). Each of the 25 problems of the competition has a unique shifted global optimum and is of a generalizable search space dimension.

Five problems were selected, as to create five tuning problems per algorithm tuned under multiple OFE budgets. These problems, which were used in 30 dimensions, are:

- problem 3, a shifted and rotated high conditioned elliptic problem
- problem 5, Schwefel's problem 2.6 with the global optimum on the bounds
- problem 6, a shifted Rosenbrock problem
- problem 8, a shifted and rotated Ackley problem with the global optimum on the bounds
- problem 10, a shifted and rotated Rastrigin problem.

It is expected that for each of these selected problems, the optimization algorithms being tuned will require different CPVs in order to achieve good performance, since each problem has different fitness landscape characteristics. Another reason for the selection of these problems is that they are computationally cheap relative to many of the other CEC'05 problems, allowing for more extensive numerical experiments to be conducted. Noisy problems were not considered as the algorithms to be tuned to the CEC'05 problems are all configured for noise-free optimization.

Selected optimization algorithms are tuned according to the normalized solution error value from each of these CEC problems. The normalization of the solution error values, although not required by tMOPSO or any other tuning algorithm which is going to be assessed, simplifies the interpretation and presentation of the results obtained by the numerical experiments. The normalized solution errors, $\hat{\epsilon}$, are calculated using a weight scalar, $\hat{w}$, as follows

$$\hat{\epsilon} = \hat{w} \cdot \epsilon \tag{2.26}$$

The value of $\hat{w}$ was numerically approximated so that a mean value of 1.0 is obtained for $\hat{\epsilon}$ when selecting a decision vector randomly, with a uniform probability density, from inside the search space of the problem being used in the application layer. The $\hat{w}$ values used for CEC problems 3, 5, 6, 8 and 10 are 1.506×10^{-10}, 1.175×10^{-5}, 3.461×10^{-12}, 4.590×10^{-2} and 4.907×10^{-4} respectively.

Each of the single-objective algorithms tuned, are tuned to each one of the selected CEC'05 problems separately. Given that three algorithms are tuned, a total of 15 tuning problems are used to compare the chosen tuning algorithms. For all of the tuning problems a $\beta_{\max}$ of 30 000 OFEs is used. Although this value of β_{max} is lower than the 300 000 specified in the CEC'05 competition, the chosen β_{max} is considered to be sufficient to determine CPV tuples effective for both high and low solution accuracy requirements, since optimization algorithms are to be tuned directly to each problem instance.

2.3.2 Algorithms Tuned

Well known population-based optimization algorithms are tuned to the selected CEC'05 problem instances. These algorithms are a differential evolution (DE) algorithm, a particle swarm optimization (PSO) algorithm and a covariance matrix adaption evolutionary strategy (CMA-ES; Hansen and Ostermeier, 2001) optimization algorithm. A brief description of each algorithm, together with information on which control parameters are tuned, follow.

DE

Differential evolution was developed to optimize non-differentiable, non-linear cost functions which are multi-modal (Storn and Price, 1997; Das and Suganthan, 2010). The *rand/1/bin* (Storn and Price, 1997) variation of DE, using the bound constraint handling mechanism proposed in Zhang and Sanderson (2009), is tuned under multiple OFE budgets by altering:

- the population size N,
- the scaling factor F, and
- the crossover probability parameter C_r.

Based on Wang et al. (2011), the initialization bounds of the DE tuning problems are $N \in [5, 200]$, $F \in [0, 2]$ and $C_r \in [0, 1]$, and the DE tuning problem's constraints are:

$$5 \leq N \tag{2.27}$$

$$0 \leq C_r \leq 1 \tag{2.28}$$

$$0 \leq F. \tag{2.29}$$

PSO

The single objective PSO variant tuned uses a global neighborhood typology, zero initial velocities, a fixed inertia factor, and the bound constraint handling mechanism of Zhang and Sanderson (2009). The four PSO control parameter values tuned are:

- the swarm size N,
- the personal best acceleration constant c_p,
- the global best acceleration constant c_g, and
- the inertia factor ω.

The initialization bounds of the PSO tuning problems were chosen as $N \in [5, 200]$, $\omega \in [0, 1]$, $c_p \in [0, 3]$, $c_g \in [0, 3]$ based upon the studies presented in Shi and Eberhart (1998); Clerc and Kennedy (2002). The PSO tuning is constrained as follows:

$$5 \leq N \tag{2.30}$$
$$0 \leq c_p \tag{2.31}$$
$$0 \leq c_g \tag{2.32}$$
$$0 \leq \omega. \tag{2.33}$$

Additional constraints such as $\omega \leq 1$ and $c_p + c_g \leq 4$ (Clerc and Kennedy, 2002) are omitted, since these common recommendations may be detrimental for low OFE budgets were swarm explosion may be beneficial.

CMA-ES

The covariance matrix adaptation evolutionary strategy (Hansen and Ostermeier, 2001) was developed to handle badly scaled quadratic problems and is invariant against linear transformations of the search space (Auger and Hansen, 2005). Version 0.9.56 of the Python implementation of CMA-ES written by the algorithm's original author was tuned by adjusting the following CPVs:

- the population size N,
- the parent selection fraction μ_f, and
- the maximum step size as a ratio of the search initialization bound size σ_r.

The tuning initialization bounds are $N \in [5, 200]$, $\mu_f \in [0.1, 0.9]$ and $\sigma_r \in [0.1, 0.9]$. N, μ_f and σ_r are constrained to

$$5 \leq N \tag{2.34}$$
$$1 \leq \lfloor N \cdot \mu_f \rfloor \tag{2.35}$$
$$\mu_f \leq 1 \tag{2.36}$$
$$0.01 \leq \sigma_r. \tag{2.37}$$

2.3.3 Tuning Algorithms Compared

The multiple OFE budget algorithms compared are tMOPSO, two tMOPSO variants and the Flexible Budget method (FBM;Branke and Elomari, 2012). The tMOPSO variants each of which have core elements of tMOPSO removed, are:

- tMOPSO^{-} which uses standard resampling instead of the MWUT-based resampling strategy for handling noise,
- tMOPSO$^{=}$ which uses standard resampling for handling noise, and also does not use the additional history information from the solution error calculations.

If the theoretical basis upon which tMOPSO is constructed is correct, tMOPSO$^{=}$ should be outperformed by tMOPSO^{-} which in turn should be outperformed by tMOPSO. The Dréo (2009) proof-of-concept algorithm is not compared against the tMOPSO and FBM algorithms, since it is not a multiple OFE budget tuning algorithm, as was discussed in the related work section.

The single OFE budget tuning algorithms used in the numerical experiments are REVAC, SPO, iterated F-race (I/F-race, López-Ibáñez et al., 2011) and a single objective variant of tMOPSO, named tPSO. tPSO is a stripped down version of tMOPSO which has the OFE target auxiliary variable removed, to reduce the algorithm to a single objective tuning method focused on one OFE budget only.

A resampling size of 25 for approximating mean utility values is used by all the compared tuning algorithms. Given that the tuning problems' application layers are noise free, a size of 25 should be sufficient to approximate the mean utility values within reasonable statistical confidence levels.

Similarly to the algorithms they are tuning, the performance of the compared tuning algorithms is suspected to be sensitive to both their control parameter values and their computational budget. To account for sensitivity to computational budgets, the tuning algorithms are compared over a range of application layer evaluation (γ) budgets. The maximum comparison gamma used is 15×10^7, which corresponds to performing 5 000 CPV tuple assessment runs up to a β_{max} of 30 000. For the standard resampling methods, which generate 25 samples for each CPV tuple evaluated, this gamma budget translates to assessing 200 CPV tuples at β_{max}. To account for sensitivity to control parameters, parameter sweeps are conducted for each tuning algorithm, before they are compared against each other. The pre-comparison parameter sweeps aim to ensure that each tuning algorithm uses parameters which are well suited to the DE tuning problems used in these experiments. Performance on the PSO and CMA-ES tuning problem is not used in the parameter sweeps, to both save computational resources, and to see if any of the algorithms suffer from over-tuning.

The procedure for selecting parameters for each of the compared tuning algorithms, entails the use of a Friedman test. The candidates for the Friedman test are generated using parameter sweeps, with each candidate being gauged according to five criteria. For the multiple OFE budget tuning algorithms, the five criteria are the hypervolume (HV; Zitzler et al., 2003) achieved on each of the DE tuning problems, for a γ budget of 6×10^7. Since the CEC function values are normalized, the HV reference point used for all problems is $[\beta_{max}, 1]$. For the single OFE budget tuning algorithms, the five criteria are the minimum solution error achieved for the DE tuning problems, also for a γ budget of 6×10^7. A resampling size of 10 is used to approximate these performance measures, for each of the CPV tuples investigated. The CPV tuple with the highest Friedman rank is then used by the respective tuning algorithm for the remainder of the experiments.

After the individual parameter sweeps for each of the compared tuning algorithms are completed, those algorithms are applied to the DE, PSO and CMA-ES tuning problems using the winning CPV tuples. Since these tuning algorithms are stochastic, samples of 20 independent runs for each algorithm on each tuning problem are generated to compare performances. Descriptions of each of the tuning algorithms compared, and the parameters varied for the CPV sweeps, follow in the remainder of this section.

tMOPSO and its variants

The CPV tuple candidates for the CPV sweeps for tMOPSO and its variants were chosen according to selected PSO literature (Coello et al., 2004; Shi and Eberhart, 1998; Clerc and Kennedy, 2002; Pedersen, 2010). The parameters varied are the inertia factor and the swarm size, with the 110 combinations resulting from $\omega \in \{0.0, 0.1, \ldots 1.0\}$ and $N \in \{5, 10, \ldots 50\}$ being assessed for favorable performance on the DE tuning problems. The fixed CPVs for tMOPSO and its variants include the OFE perturbation factor $c_\beta = 0.1$, a global acceleration constant of $c_g = 2.0$ and a personal or local acceleration constant value of $c_p = 2.0$. tMOPSO and tMOPSO$^-$ use a target OFE overshoot factor of $\lambda = 2.0$ for calculating the solution errors. For handling noise, tMOPSO and tPSO use an interruption confidence of 90% and sample size increments of $\Delta_{n_s} = \{2, 3, 5, 15\}$ for the MWUT-based strategy. As for tMOPSO's control parameter, B, which specifies the OFE budgets for which the single objective algorithm are to be tuned under, 100 OFE budgets logarithmically spaced between 30 and 30 000 are used. The implementations of tMOPSO and its variants, are available in version 0.10 of the optTune Python package[1].

FBM

The Flexible Budget method is a population based tuning algorithm, which uses the number of OFEs made versus solution error curves to tune under multiple OFE budgets (Branke and Elomari, 2012). Each of the N individuals has a CPV tuple as its decision vector, which is randomly generated inside the initialization bounds using a uniform distribution at the start of the tuning optimization. FBM's individuals are ranked according to their OFEs used versus solution error curves, where a curve is calculated by running the algorithm being tuned up to β_{max}, using the corresponding individual's CPV tuple. The ranking procedure begins by assigning the highest rank to each individual with a curve which is optimal for any OFE budget under consideration. Thereafter, the rank counter is increased and the unranked individuals are compared in isolation. At the next ranking iteration, all unranked curves which are optimal compared to the other unranked curves for an OFE budget, are assigned a rank equal to that of the rank counter. This iterative process is repeated until all individuals are ranked. If individuals are compared and they have the same rank, three options are available for tie-breaking, namely the number of OFE budgets for which the curve was optimal for at their rank, the area under the curve, and the area lost if the curve is removed. Based on Branke and Elomari (2012), the implemented FBM uses area under the curve for tie-breaking. At each generation, size 2

[1] http://code.google.com/p/opt-tune-python-package/.

tournaments are conducted, one fold crossover and Gaussian mutation are used to generate offspring. The N offspring then compete against their parents for survival, so that only N out the $2N$ individuals survive. FBM uses standard resampling to handle noise.

For computational overhead considerations, FBM is modified to only focus on specified OFE budgets, instead of all the OFE budgets up to β_{max}. These target OFE budgets are the same as those specified in tMOPSO's B control parameter. Additionally, in accordance with the tuning problem formulation in (2.8), FBM is not bound constrained and is free to explore outside the CPV initialization bounds. FBM uses the same constraint handling approach as tMOPSO and its variants, whereby candidate CPV tuple generation repeats until all the constraints are satisfied, after which the CPV tuple is assessed. Our implementation of FBM is available in version 0.10 of the optTune Python package.

For FBM, different combinations of N and the control parameter m_s are assessed in the CPV sweep. m_s controls the standard deviation of FBM's Gaussian mutation, with the standard deviation being equal to m_s multiplied by the range of the initialization bounds, $\mathbf{I}^U - \mathbf{I}^L$. The 110 combinations resulting from $N \in \{5, 10, \ldots, 50\}$ and $m_s \in \{0.0, 0.05, \ldots, 0.5\}$ are assessed for good overall performance on the DE tuning problems.

REVAC

The relevance estimation and value calibration method is a single objective tuning algorithm which uses Shannon entropy models to guide the tuning process (Nannen and Eiben, 2007). At the start of the tuning optimization, R_i CPV tuples are generated randomly throughout the search space. Thereafter, R_p CPV tuples are used to fit the Shannon entropy models as to generate a new candidate CPV tuple. This fitting and CPV tuple generating process is repeated until the computational budget is exhausted. REVAC uses standard resampling to handle noise. The REVAC implementation from the algorithm's original paper (Nannen and Eiben, 2007) is used.

The 99 combinations of $R_i \in \{10, 20, \ldots 100\}$ and $R_p \in \{0.1R_i, 0.2R_i \ldots, 1.0R_i\}$ (minus the combination where R_p is one) are assessed for the pre-comparison tuning. REVAC enforces search bound constraints. As such, the population size of the algorithm being tuned is bound between $N \in [5, 400]$, with the following algorithm-specific tuning search bounds for DE, $F \in [0, 2]$, $C_r \in [0, 1]$, for PSO $\omega \in [0, 1]$, $c_1 \in [0, 4]$, $c_2 \in [0, 4]$ and for CMA-ES: $\mu_f \in [0.01, 1]$, $\sigma_r \in [0.01, 1]$. REVAC also uses these bounds as the initialization bounds.

SPO

The sequential parameter optimization framework uses surrogate modeling to tune an optimization algorithm (Bartz-Beielstein et al., 2005). Numerous surrogate modeling options and initialization strategies are available. In these numerical experiments, SPO is set up to use Kriging Gaussian models (Zhang et al., 2010) and Latin Hypercube sampling to generate the initial candidate CPV tuples. After generating an initial group of CPV tuples, SPO splits computational resources between gathering additional sample points for CPV tuples already evaluated and assessing new CPV tuples. Specifically, the optimal computing budget allocation (OCBA) approach is used to decide which CPV tuples are promising and how many additional samples

should be generated for those promising candidates. A modified version of the SPO algorithm in SPOT version 1.0.2667[2] is used in these experiments. SPO was modified as to adhere to an upper limit for resampling. This modification was required as otherwise SPO would spend computational resources refining CPV tuple samples above the desired maximum of 25.

The SPO parameters investigated for good overall performance on the DE tuning problems, are the number of points used to fit the Kriging model S_k, the number of new CPVs assessed for each iteration S_n, and the OCBA budget S_o. In particular, the 120 combinations resulting from $S_k \in \{6, 12, \ldots, 30\}$, $S_n \in \{1, 2, \ldots, 8\}$ and $S_o \in \{9, 15, 21\}$ are assessed. Based on a recent SPO paper (Wagner and Wessing, 2012), the initial number of CPV tuples assessed is fixed at 30, and four initial samples are generated for each CPV tuple being assessed. SPO is also a bound constrained method, and is setup to use the same bounds as REVAC.

I/F-race

The iterated F-race method tunes an algorithm by performing successive F-races. The results from each F-race are used to reduce the search space used for generating candidate CPV tuples for the next F-race, as to home in on promising CPV tuples. The I/F-race implementation used in these experiments is from version 1.04 of the irace package[3], which was modified as to enforce a resampling size limit, and to allow for an user-specified rate of search space reduction. Initially, I/F-race generates I_n candidate CPV tuples inside the search bounds, and conducts an F-race amongst these CPV tuples. For each F-race thereafter, I_n new candidate CPV tuples are generated, and raced against the winner or winners from the previous race. These new candidates are randomly generated around the winners from the previous F-race, using a Gaussian distribution with a standard deviation of I_σ, subject to those new candidates being in the search bounds. The rate of reduction of I_σ is controlled through the parameter I_r, which specifies the desired ratio of the final I_σ to that of the initial I_σ. As a function of the fraction of tuning budget used ζ,

$$I_\sigma = 0.5\,(\mathbf{b}^U - \mathbf{b}^L)\,e^{\zeta \ln I_r}, \tag{2.38}$$

where $\mathbf{b}^U$ and $\mathbf{b}^L$ are the search bounds. As for the F-races themselves, Friedman tests for eliminating CPV tuple candidates unlikely to be competitive begin after I_f samples.

The danger of search bounds reduction approaches such as I/F-race is that bounds are reduced incorrectly, with the search homing in on wrong regions of the search space. For this reason the I/F-race parameters varied for the pre-comparison tuning, influence the search space reduction of I/F-race. The I_r values assessed are based upon I/F-race's search bounds, which are the same as REVACs. Specifically, the I_r values assessed are in $\{4^{-2}, 8^{-2}, \ldots, 24^{-2}\}$, where $1/20^2$ is approximately equal to $1/395$, where 395 is the difference in range of populations of I/F-race's search bounds for the DE, PSO and CMA-ES tuning problems. The I_f values of $\{2, 5, 10\}$ are assessed, where an I_f of 2 is computationally the cheapest but has the highest risk of leading I/F-race astray, and an I_f of 10 being the opposite. Additionally, the number of new candidates generated is varied, $I_n \in \{5, 10, \ldots, 30\}$, giving a total of 108 CPV tuples assessed. The confidence level of the I/F-race Friedman test is fixed to 90%.

[2] http://cran.r-project.org/web/packages/SPOT/index.html
[3] http://cran.r-project.org/web/packages/irace/index.html

2.4 Numerical Results

The results from numerical experiments constructed to gauge the effectiveness of tMOPSO are presented and discussed in this section. The comparison of the multiple OFE budget tuning algorithms is presented first. Thereafter follows the comparison of tMOPSO with the single OFE budget tuning algorithms, as gauge to the use of tMOPSO as an alternative to setting up multiple uncoupled single OFE budget tuning problems. Lastly, tMOPSO's results for tuning DE and PSO are scrutinized against previous studies.

2.4.1 Comparison of Tuning Algorithms Focused on Multiple OFE Budgets

The parameter tuples for tMOPSO, tMOPSO$^-$, tMOPSO$^=$ and FBM which were found to result in the best overall performance on the DE tuning problems are presented in Table 2.1. For all the parameter sweeps, the Friedman test conducted showed that the performance difference between the candidate CPV tuples was statistically significant given a confidence level of 90%. The candidate CPV tuples with best Friedman rank, using the HV achieved on the DE tuning problems as the five criteria, were then applied to the DE, PSO and CMA-ES tuning problems.

On 14 out of the 15 CEC'05 tuning problems tMOPSO achieved the greatest mean HV over all γ considered, the exception being the tuning of CMA-ES to CEC'05 problem 8, as summarized in Table 2.2 and plotted in Figure 2.6 to Figure 2.8. For the DE and CMA-ES CEC'05 problem 8 tuning problems, Mann-Whitney U tests showed that on the difference between the mean HV achieved by tMOPSO and the means of the other tuning algorithms compared are not statistically significant with a confidence level of 95%. In particular the mean difference between tMOPSO and tMOPSO$^-$ was not statistically significant for the tuning of DE to CEC'05 problem 8, and the difference of tMOPSO, tMOPSO$^-$ and FBM was not significant when tuning CMA-ES to CEC'05 problem 8.

The poorer performance of tMOPSO$^-$ and tMOPSO$^=$ compared to tMOPSO is expected, since these algorithms are stripped versions of tMOPSO with core elements removed. In particular, tMOPSO$^=$'s worse performance compared with tMOPSO and tMOPSO$^-$ is expected, since tMOPSO$^=$ does not use the additional history information from the solution error calculations as tMOPSO$^-$ and tMOPSO do. The out-performance of tMOPSO$^-$ by tMOPSO is also expected, because tMOPSO$^-$ uses standard resampling instead of resampling which makes use of MWUTs to interrupt the sample gathering process as tMOPSO does. These results although expected are important as they provide supporting evidence for the theoretical basis upon which tMOPSO was developed.

Table 2.1: The parameters which were found to result in the best overall performance on the DE tuning problems, for the multiple OFE budget tuning algorithms.

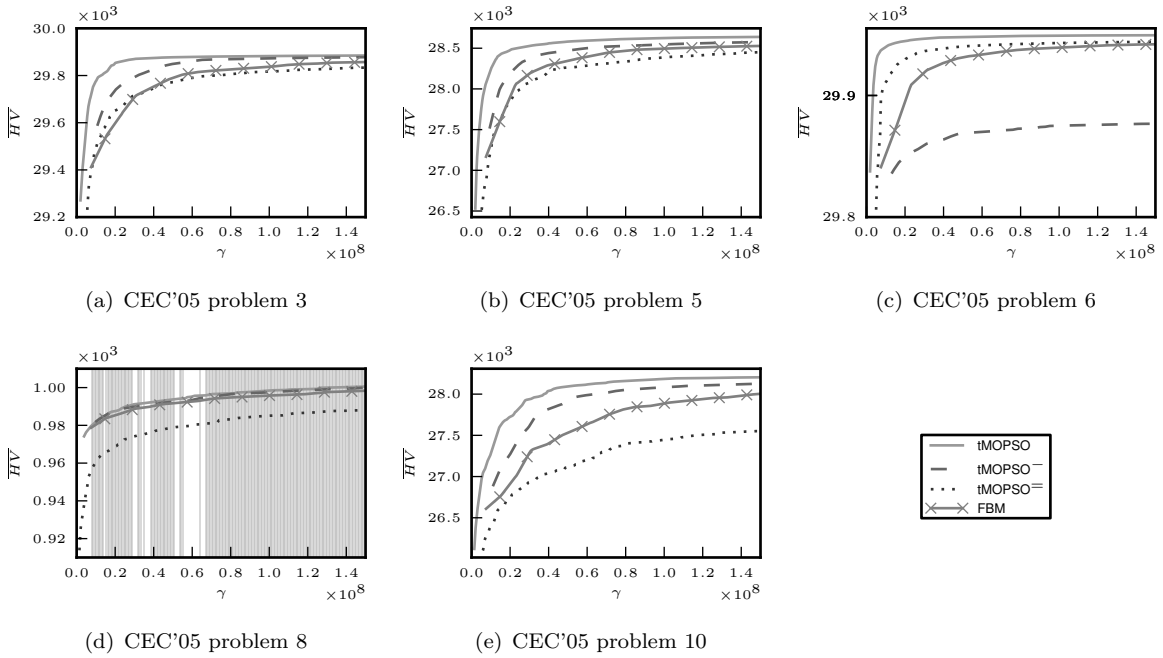
algorithm	best CPVs found	
tMOPSO	$N = 10$	$\omega = 0.2$
tMOPSO$^-$	$N = 35$	$\omega = 0.0$
tMOPSO$^=$	$N = 5$	$\omega = 0.3$
FBM	$N = 10$	$m_s = 3^{-2}$

Table 2.2: The mean hypervolume found by the multi-objective tuning methods on the chosen tuning problems, for various application layer evaluation (γ) budgets.

		mean HV found ($\times 10^3$) for $\gamma = 3 \times 10^7$ by			
algorithm	CEC problem	tMOPSO	tMOPSO$^-$	tMOPSO$^=$	FBM
DE	3	**29.870**	29.783	29.714	29.666
	5	**28.530**	28.337	28.076	28.084
	6	**29.947**	29.855	29.936	29.910
	8	**0.991**	0.990	0.974	0.987
	10	**27.937**	27.557	26.927	27.073
PSO	3	**29.871**	29.844	29.806	29.816
	5	**27.599**	27.413	27.122	27.165
	6	**29.953**	29.947	29.944	29.930
	8	**1.867**	1.686	1.479	1.487
	10	**27.789**	27.616	27.281	27.476
CMA-ES	3	**29.906**	29.894	29.885	29.859
	5	**29.563**	29.535	29.471	29.497
	6	**29.945**	29.939	29.942	29.905
	8	*0.990*†	**0.997**	0.975	*0.988*
	10	**29.262**	29.223	29.045	29.144

		mean HV found ($\times 10^3$) for $\gamma = 15 \times 10^7$ by			
algorithm	CEC problem	tMOPSO	tMOPSO$^-$	tMOPSO$^=$	FBM
DE	3	**29.885**	29.878	29.834	29.856
	5	**28.639**	28.576	28.449	28.527
	6	**29.949**	29.877	29.944	29.942
	8	**1.001**	*1.000*	0.988	0.998
	10	**28.204**	28.124	27.551	27.997
PSO	3	**29.891**	29.885	29.851	29.874
	5	**27.795**	27.712	27.441	27.605
	6	**29.955**	29.954	29.950	29.950
	8	**1.968**	1.953	1.736	1.863
	10	**27.974**	27.892	27.626	27.847
CMA-ES	3	**29.914**	29.911	29.904	29.902
	5	**29.602**	29.589	29.550	29.592
	6	**29.948**	29.947	29.944	29.942
	8	*1.008*	**1.011**	0.989	1.000
	10	**29.332**	29.321	29.232	29.300

† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 95% confidence level.



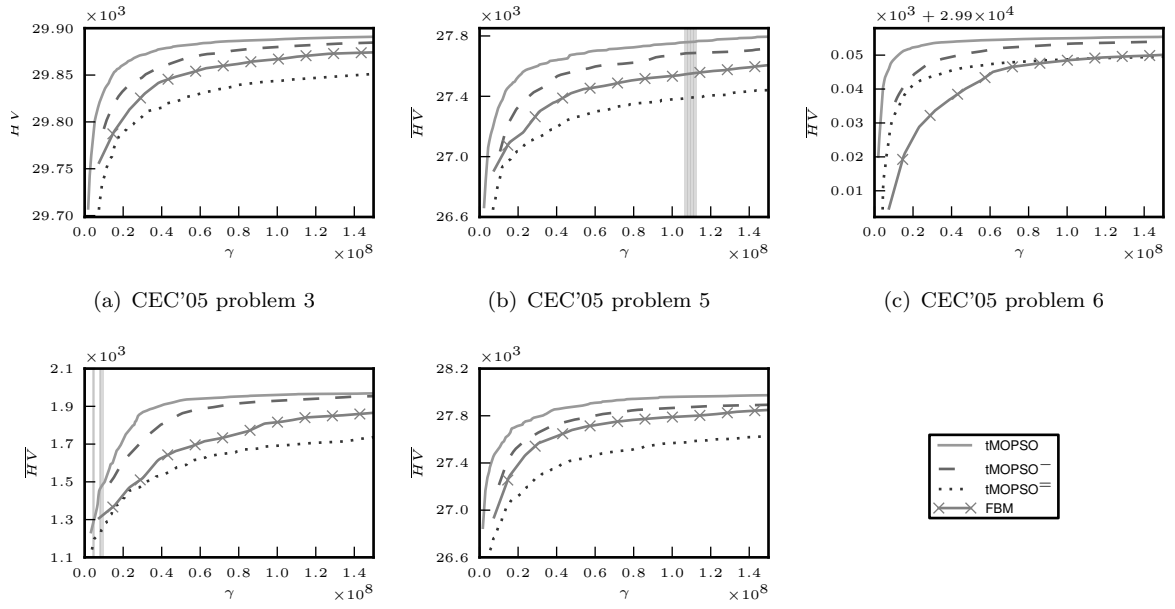
(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10

Figure 2.6: Mean hypervolume obtained ($\overline{HV}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the DE tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.

The out-performance of FBM by tMOPSO on the tuning problem used in these experiments could be for various reasons. Firstly, the evolutionary operators used by FBM could be poorly suited to the selected tuning problems compared to particle swarm operators used by tMOPSO and its variants. However, if these mechanics were the only differentiating factor, tMOPSO$^{=}$ would perform better then FBM, which it does not. tMOPSO$^{=}$ being outperformed by FBM is most likely due to FBM using the history information from the OFE budget solution error calculations, which tMOPSO$^{=}$ does not do. Another important difference between FBM and tMOPSO, is that tMOPSO does not evaluate every CPV tuple assessed up to the maximum OFE budget of interest. tMOPSO multi-objective formulation allows for tMOPSO to predict which OFE budgets a CPV tuple will be competitive at, thereby saving computational resources. If the number of CPV tuples assessed in the DE, PSO and CMA-ES tuning problems are compared, tMOPSO^{-} evalutes between 4 and 10 times more CPV tuples than FBM. tMOPSO's noise handling strategy based on MWUTs further increases this ratio, with tMOPSO evaluating between 10 and 30 times more CPV tuples than FBM, as shown in Table 2.3. Even though tMOPSO may have miscalculated a large portion of OFE budgets at which some of these CPV tuples assessment are likely to be effective at, this order increase in the number of CPV tuples assessed, is suspected to have strongly contributed to FBM being outperformed by tMOPSO.

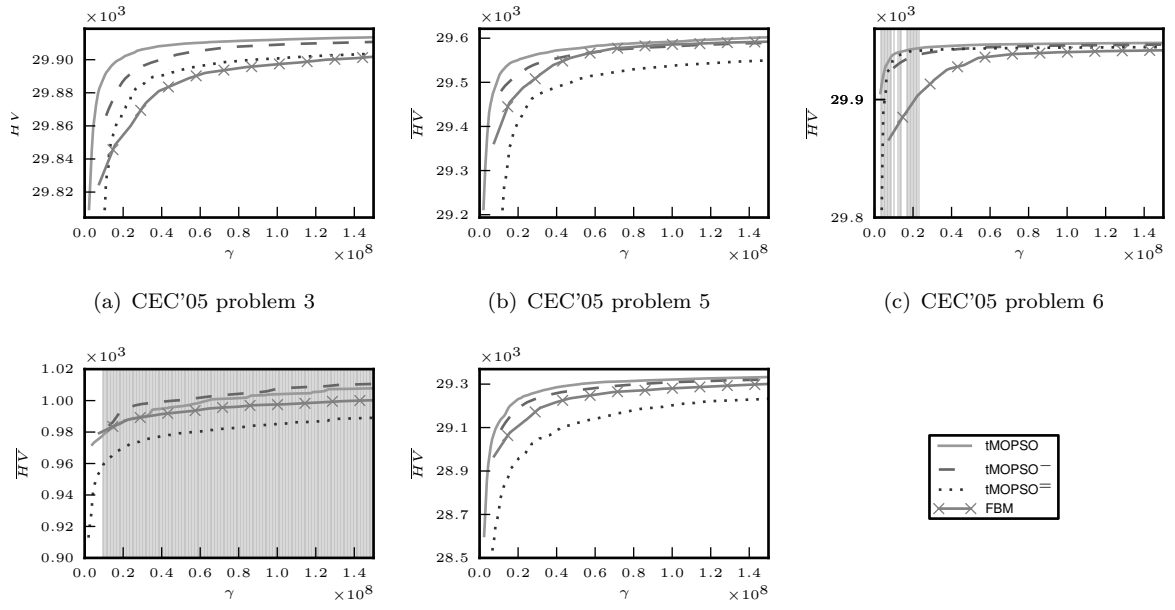
These numerical results are also used to investigate tMOPSO's computational overhead in practice. Gauging computational overhead is of particular interest since a large portion of tMOPSO's design is focused on reducing computational overhead. To gauge computational overhead, the overhead ratios of tMOPSO runs were calculated, where the overhead ratio is calculated by dividing the computing time used by the tuning algorithm, by the computing



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10

Figure 2.7: Mean hypervolume obtained ($\overline{HV}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the PSO tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10

Figure 2.8: Mean hypervolume obtained ($\overline{HV}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the CMA-ES tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.

Table 2.3: Mean number of CPV tuples assessed by the multiple OFE budget tuning algorithms for the DE, PSO and CMA-ES problems

algorithm	CEC problem	tMOPSO	tMOPSO^{-}	tMOPSO$^{=}$	FBM
DE	3	5582.5	1737.8	4468.0	205.5
	5	4747.5	1197.0	4513.8	205.4
	6	4088.0	1125.2	4263.2	206.8
	8	2123.5	1228.5	3907.5	204.9
	10	5923.0	1314.2	6421.0	205.5
PSO	3	5195.5	1389.5	4686.5	205.4
	5	3859.5	1279.2	4496.5	204.8
	6	4026.5	1330.0	3325.8	205.9
	8	3564.0	918.8	3186.2	205.7
	10	4833.5	1109.5	4009.3	205.1
CMA-ES	3	3142.0	985.2	7258.2	205.5
	5	3765.0	868.0	9606.0	205.5
	6	4592.0	1062.2	11801.5	204.8
	8	2113.0	1253.0	3925.8	205.5
	10	4321.5	964.2	14731.5	206.2

time used by the algorithm being tuned. The extreme cases arise for the DE and PSO tuning problems, for which computationally cheap, highly optimized Fortran code is tuned. For the DE tuning to CEC'05 problems 3, 5, 6, 8 and 10 overhead ratios of 28%, 58%, 80%, 2% and 14% were recorded with similar overheads for the PSO tuning problems. These ratios are considered low given the highly optimized code which is tuned. For scenarios considered more typical, where a non-optimized code is tuned, such as the tuning of CMA-ES algorithm, which is a Python code, overhead ratios of 0.4%, 0.3%, 0.3%, 0.2%, and 0.4% were recorded for CEC'05 problems 3, 5, 6, 8 and 10, respectively. tMOPSO's computational overhead is higher than that of the other compared tuning algorithms which either use standard resampling or only focus on one OFE budget. That mentioned, tMOPSO's overhead is still low enough to be considered acceptable for standard use case scenarios. Moreover, if computational overhead is an issue, users can adjust tMOPSO's parameters controlling the pre-emptively terminating resampling as to reduce the number of MWUTs performed by tMOPSO by changing Δ_{n_s}, as well as reduce the number of OFE budgets tMOPSO tunes under by changing the B control parameter.

2.4.2 Comparison with Tuning Algorithms Focused on a Single OFE Budget

The selected single OFE budget tuning algorithms are compared against tMOPSO, each using parameters found to be well-suited to the DE tuning problems. As with the multiple OFE budget algorithms, Friedman tests showed that the parameters varied did have a statistically significant effect on each of the single OFE budget algorithms performances. The parameters found by CPV sweeps for each of the compared single OFE budget tuning algorithms are shown in Table 2.4.

The comparison of tMOPSO against these single OFE budget tuning algorithms shows that no tuning algorithm outperforms the rest, as depending on the tuning problem and γ, different tuning algorithms performed better as summarized in Table 2.5. Furthermore, for

Table 2.4: The parameters which were found to result in the best overall performance on the DE tuning problems, for the single OFE budget tuning algorithms.

algorithm	best CPVs found		
tPSO		$N = 10$	$\omega = 0.5$
REVAC		$R_p = 30$	$R_i = 9$
I/F-race	$I_n = 10$	$I_r = 8^{-2}$	$I_f = 2$
SPO	$S_k = 30$	$S_n = 8$	$S_o = 21$

many of the tuning problems over a large range of γ, the difference between the best sample mean and the other sample means of the compared algorithms is not statistically significant with a confidence level of 95%, as shown in Figure 2.9 to Figure 2.11. To quantify the relative performances of the compared tuning algorithms, a rank based analysis similar to that used in the CEC'05 competition was done. For each tuning problem, the competing algorithms were ranked according to the mean of the minimum solution error found at an OFE budget of $\beta_{\max}$, determined after a γ of 3×10^7 and 15×10^7, respectively. After the ranks were calculated, the rank sum over all the tuning problems was used to gauge each algorithm's performance. The ranks which are summarized in Table 2.5 show that for a γ of 3×10^7, tPSO performed the best with a rank sum of 28, followed by I/F-race with 33, SPO with 43, tMOPSO with 53 and REVAC with 68. For a γ of 15×10^7, tPSO again performed the best with a rank sum of 27, follow by I/F-race with 34, tMOPSO with 51, REVAC with 55 and SPO with 58. If these rank sums are taken into consideration with the fact that tPSO achieves the best mean solution error on 5 out of the 15 tuning problems for a γ of 3×10^7, and 10 out of 15 tuning problems for a γ of 15×10^7, it is concluded that tPSO results in the best overall performance on the tuning problems.

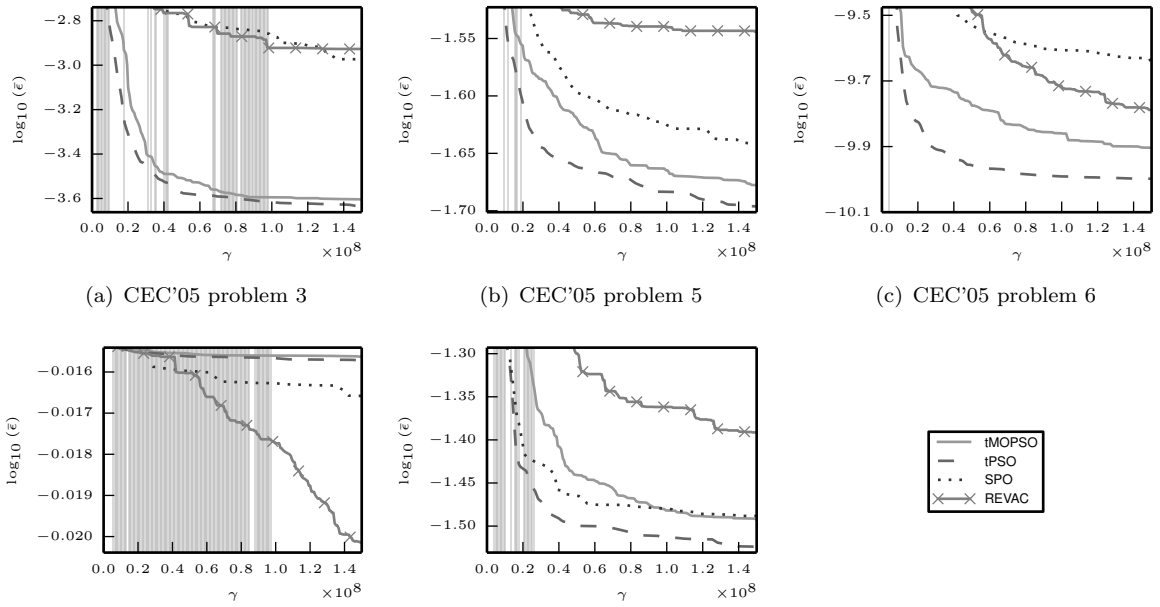
Analysis of the tuning algorithm performances allows the research question, regarding if it would be more efficient to use tMOPSO to tune an algorithm under multiple OFE budgets, compared to setting up multiple tuning problems each focused on a single OFE budget, to be answered. Specifically, the extra computational effort required to run a single OFE budget tuning algorithm on the 100 logarithmically spaced OFE budgets which tMOPSO tunes under in these experiments, is compared to the extra computational effort required by tMOPSO to tune under those 100 OFE budgets and compete against the single OFE budget algorithm in terms of best solution error at β_{max}. As shown in Table 2.6, the additional computational effort for tMOPSO measured in terms of γ varies depending upon both the tuning problem and the γ budget allocated to the single OFE budget tuning algorithms. On one third of the data points generated tMOPSO can compete with less then 100% extra γ. If tMOPSO's extra γ is increased to 250% then tMOPSO produces a better or comparable mean for two thirds of the data points. Increasing the extra γ to 500%, allows tMOPSO to compete on 80 out of the 90 data points investigated in Table 2.6. For the remaining 10 data points, which correspond to tuning DE to CEC'05 problem 8 and when tuning CMA-ES to CEC'05 problem 8, some of the tMOPSO runs got stuck in what could be viewed as a tuning local minimum, and cannot compete no matter how much extra γ is allocated. Increasing tMOPSO's population size is expected to reduce the risk of this occurring for these two Ackley based tuning problems. For

Table 2.5: Comparison between tuning algorithms based on mean of the mean minimum solution error at $\beta_{\max}$ ($\bar{\epsilon}$), for various application layer evaluation budgets (γ). Rankings are given in parenthesis next to the values.

		$-\log_{10}(\bar{\epsilon})$ for $\gamma = 3 \times 10^7$				
algorithm	prob.	tMOPSO	tPSO	I/F-race	SPO	REVAC
DE	3	3.350(2)	**3.440**(1)	*3.079*†(3)	2.726(4)	2.648(5)
	5	1.585(3)	*1.644*(2)	**1.653**(1)	1.550(4)	1.498(5)
	6	9.724(3)	**9.901**(1)	*9.892*(2)	8.661(5)	8.835(4)
	8	*0.016*(5)	*0.016*(4)	*0.016*(2)	**0.016**(1)	*0.016*(3)
	10	1.366(4)	*1.468*(2)	**1.479**(1)	1.426(3)	1.149(5)
PSO	3	3.049(2)	**3.174**(1)	3.049(3)	3.010(4)	2.859(5)
	5	1.210(4)	*1.250*(2)	**1.252**(1)	*1.243*(3)	1.184(5)
	6	*9.172*(2)	**9.372**(1)	*7.428*(3)	6.651(5)	7.284(4)
	8	*0.031*(2)	*0.031*(3)	*0.029*(4)	**0.032**(1)	0.027(5)
	10	1.238(4)	*1.286*(2)	**1.288**(1)	1.244(3)	1.191(5)
CMA-ES	3	7.312(4)	**7.570**(1)	7.454(3)	7.459(2)	7.125(5)
	5	5.263(4)	*5.620*(2)	5.523(3)	**5.631**(1)	5.008(5)
	6	10.16(4)	*10.26*(2)	**10.27**(1)	*10.23*(3)	10.02(5)
	8	0.015(5)	*0.016*(2)	0.016(4)	**0.017**(1)	*0.016*(3)
	10	2.243(5)	*2.330*(2)	**2.349**(1)	2.274(3)	2.259(4)
$\sum$ Ranks		53	28	33	43	68

		$-\log_{10}(\bar{\epsilon})$ for $\gamma = 15 \times 10^7$				
algorithm	prob.	tMOPSO	tPSO	I/F-race	SPO	REVAC
DE	3	3.604(3)	**3.636**(1)	3.612(2)	2.973(4)	2.927(5)
	5	1.678(3)	**1.696**(1)	1.696(2)	1.641(4)	1.544(5)
	6	9.902(3)	**9.997**(1)	9.969(2)	9.641(5)	9.788(4)
	8	0.016(5)	0.016(4)	0.016(3)	0.017(2)	**0.020**(1)
	10	1.491(3)	**1.524**(1)	1.513(2)	1.488(4)	1.391(5)
PSO	3	3.322(2)	**3.351**(1)	3.302(3)	3.194(4)	3.144(5)
	5	1.269(5)	*1.291*(4)	**1.297**(1)	*1.296*(2)	*1.292*(3)
	6	9.649(2)	*9.614*(3)	**9.682**(1)	8.632(5)	8.752(4)
	8	0.033(2)	**0.033**(1)	0.033(3)	0.032(5)	0.032(4)
	10	1.292(3)	**1.324**(1)	1.311(2)	1.285(4)	1.265(5)
CMA-ES	3	7.672(3)	**7.792**(1)	7.712(2)	7.634(5)	7.654(4)
	5	5.655(5)	**5.787**(1)	*5.770*(2)	5.747(3)	5.659(4)
	6	10.33(5)	*10.43*(3)	*10.46*(2)	10.41(4)	**10.47**(1)
	8	0.016(4)	0.019(3)	0.016(5)	0.020(2)	**0.026**(1)
	10	2.367(3)	**2.403**(1)	*2.401*(2)	2.337(5)	2.342(4)
$\sum$ Ranks		51	27	34	58	55

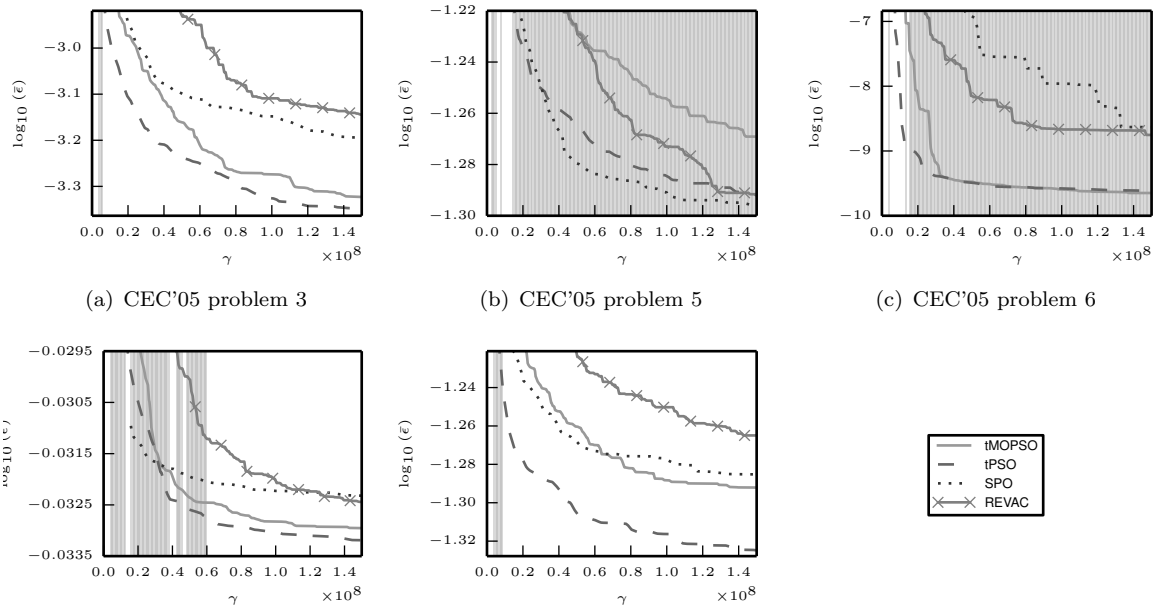
† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 95% confidence level. I/F-race searches differently depending on the specified γ, therefore the results for $\gamma = 3 \times 10^7$ and $\gamma = 15 \times 10^7$ were generated using different runs.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

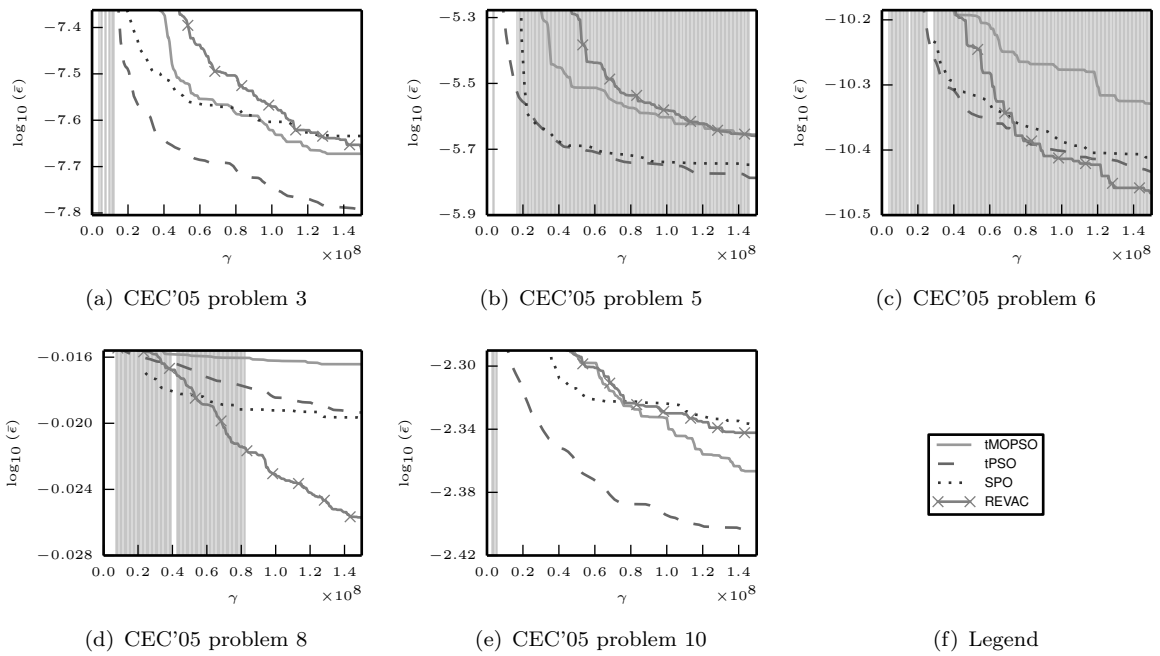
Figure 2.9: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the DE tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means. I/F-race's performance for various γ is not shown since I/F-race searches differently depending on the specified γ, and therefore the I/F-race plots are very computationally expensive to calculate.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

Figure 2.10: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the PSO tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means. I/F-race's performance for various γ is not shown since I/F-race searches differently depending on the specified γ, and therefore the I/F-race plots are very computationally expensive to calculate.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

Figure 2.11: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the CMA-ES tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means. I/F-race's performance for various γ is not shown since I/F-race searches differently depending on the specified γ, and therefore the I/F-race plots are very computationally expensive to calculate.

13 out of 15 problems where tMOPSO does not get stuck, tMOPSO's extra γ requirements need to be compared to the 1 380% extra γ required to run a single OFE budget algorithm on all of the 100 logarithmically spaced OFE budgets tMOPSO tunes under. Therefore for 13 out of 15 problems of these numerical experiments, it is more efficient to tune an algorithm under multiple OFE budget using tMOPSO, compared to setting up multiple tuning problems each focused on a different OFE budget.

tMOPSO's effectiveness at tuning under multiple OFE budgets, compared to a single OFE budget algorithm solving multiple uncoupled tuning problem is attributed to two primary factors. Firstly, tMOPSO has an advantage in that it has information of which CPV tuples perform well at different OFE budgets, which is exploited using multi-objective optimization in order to boost tuning efficiency. Secondly, as tMOPSO uses the history information from the OFE budget solution error calculations, one CPV tuple's assessment run is used to gauge performance at multiple OFE budgets. Given these reasons, tuning an optimization algorithm under a range of OFE budgets using tMOPSO should be more efficient compared with setting up multiple tuning problems, each focused on a different OFE budget, and then solving each of those problems using a single OFE budget tuning algorithm. This general conclusion holds provided that tMOPSO does not get stuck in a local minimum, which occurred for some of the tMOPSO runs on the DE and CMA-ES Ackley based tuning problems.

Table 2.6: Table showing what fraction extra γ tMOPSO requires to achieve a mean best solution error at β_{max} equal to that of the best single OFE budget tuning algorithms. The best single OFE budget algorithm which tMOPSO was compared to varies depending upon the tuning problem and the γ.

		Single OFE tuning algorithm γ budget					
algorithm	prob.	1×10^7	2×10^7	3×10^7	4×10^7	5×10^7	6×10^7
DE	3	0.71	0.45	0.31	0.54	0.84	1.10
	5	0.76	1.24	1.54	1.90	3.21	3.64
	6	1.10	3.98	3.05	3.95	4.24	3.64
	8	1.22	0.97	$> 9.00^{\dagger}$	> 6.50	> 5.00	> 4.00
	10	0.65	1.71	3.70	3.35	4.04	> 4.00
PSO	3	0.65	0.70	0.91	0.88	0.83	0.69
	5	0.76	2.01	2.21	2.63	2.34	2.22
	6	0.48	0.48	0.50	0.18	0.00	0.00
	8	0.25	0.34	0.15	0.41	0.51	0.49
	10	1.95	1.74	1.89	1.25	2.81	2.82
CMA-ES	3	0.88	1.16	1.29	2.16	1.92	2.11
	5	1.05	1.68	2.50	2.58	2.88	2.76
	6	0.25	0.83	1.15	1.91	2.03	1.57
	8	3.37	5.04	> 9.00	> 6.50	> 5.00	> 4.00
	10	2.80	2.31	3.65	2.38	1.98	3.31

† For the generation of this table, the maximum γ tMOPSO was allowed to run to was 30×10^7. The '$>$ entries' indicate where this limit was reached.

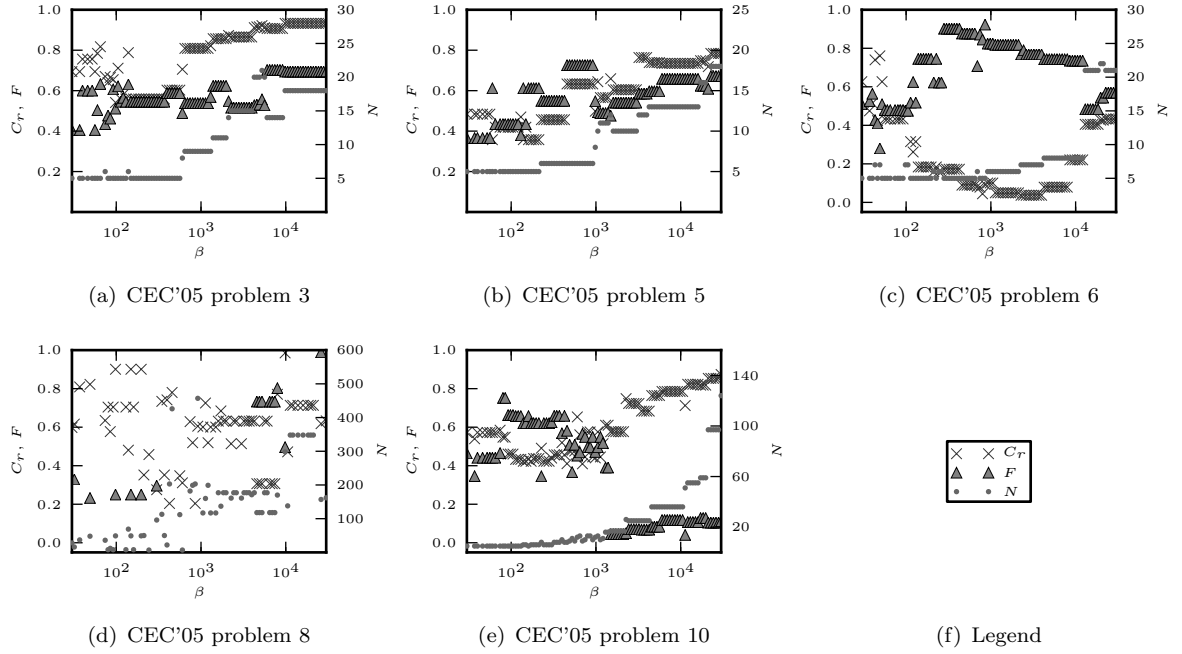
2.4.3 Scrutinization of the Tuning Results

Attention is focused next on scrutinizing the tuning results obtained by tMOPSO. Since the DE and PSO algorithm implementations tuned are sensitive to OFE budgets, tMOPSO should recommend different CPV tuples for different OFE budgets. Specifically, an increase in the optimal population size should be observed as the OFE budget increases (Dymond et al., 2011).

For the majority of tMOPSO's independent runs on the DE tuning problems an increasing population size was found optimal as the OFE budget increases, as shown in Figure 2.12 and Figure 2.13. The DE tuning results do vary from each other, but this variance is expected since DE is stochastic and the utility values were approximated numerically. The only exception where the expected trend of increasing population size was not observed, was CEC'05 problem instance 8. The DE tuning results for CEC'05 problem instance 8 vary largely from tMOPSO run to tMOPSO run, and no clear CPV trends were observed. Satisfactorily, the tMOPSO tuning results for the PSO tuning problems shown in Figure 2.14 and Figure 2.15, also recommend an increasing optimal swarm size as the OFE budget increases. Furthermore, in agreement with a literature recommendation (Clerc and Kennedy, 2002), the sum of local and global acceleration constants found by tMOPSO is less than or equal to four, except at low OFE budgets. The tMOPSO tuning results indicate that the CMA-ES optimization algorithm may also be sensitive to OFE budget constraints, as shown in Figure 2.16 and Figure 2.17. However to be certain, an extensive sensitivity study such as in (Dymond et al., 2013) should be conducted to verify the sensitivity of CMA-ES to OFE budgets. In addition to sensitivity to OFE budgets, the tuning results also indicate that optimal CPVs are sensitive to the fitness landscape of the optimization problem being tackled, an observation which was expected.

Given the sensitivity of the tuned optimization algorithms to the termination criteria and fitness landscapes of the problem being tackled, the usefulness of the tuning results themselves is limited. Moreover, practitioners are only guaranteed of achieving favorable performance using the CPVs recommended by the tuning results, should they tackle problems similar to one of the CEC'05 problems used in these experiments, and make use of the same implementations of the DE, PSO or CMA-ES algorithms which were tuned. Therefore instead of using CPVs found to be optimal in these numerical experiments, practitioners should rather apply tuning algorithms as to determine CPVs which are well-suited to testing problems representative of the optimization problem which they are tackling. Algorithm developers can assist in this regard by equipping algorithms with control parameters as to allow the algorithm to be effectively tuned to a large variety of fitness landscapes and OFE budget constraints.

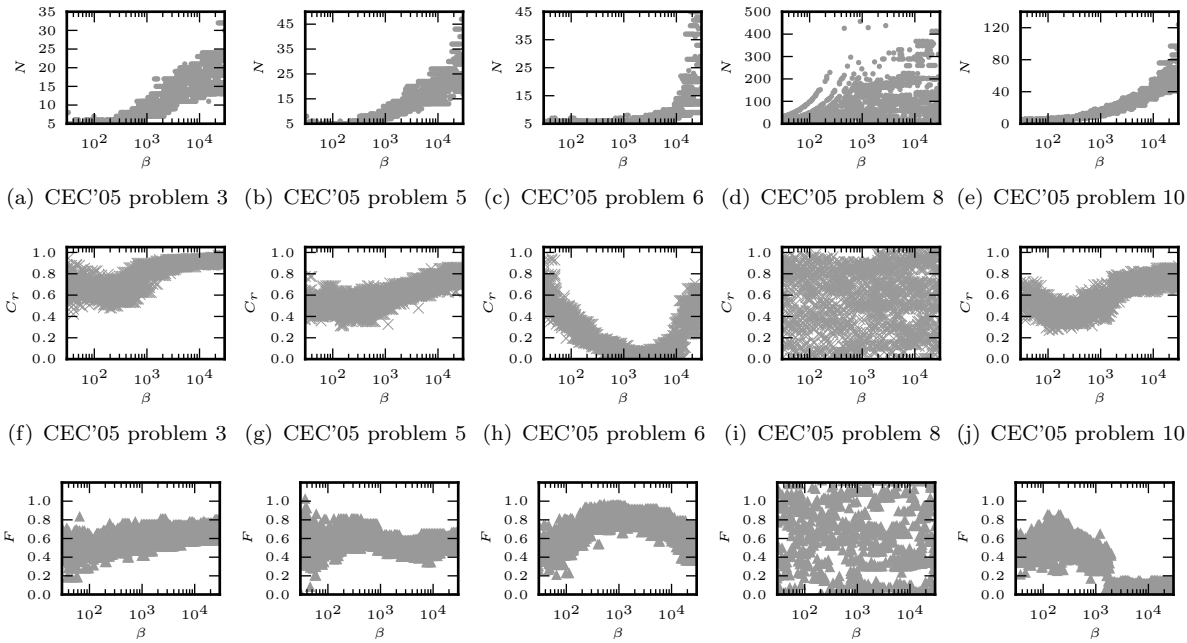
In regard to developing algorithms which are tunable to a vast range or problems, cross-examination of the tuning results indicates possible areas of improvement among the tuned algorithms. For instance, the tuned performances of the DE and CMA-ES algorithms are worse than that of PSO for the Ackley based CEC'05 problem instance 8, as shown in Table 2.2 and Table 2.5. This worse tuned performance combined with the inconsistent tMOPSO CPV recommendations, indicate that the search mechanics required for favorable performance on Ackley type problems can either not be manifested, or are very difficult to determine. This deficiency in the DE and CMA-ES tuning formulations, is either because inappropriate CPVs were tuned, or because the versions of DE and CMA-ES tuned are not capable of producing



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

Figure 2.12: tMOPSO's results for tuning DE under multiple OFE budgets, where the results shown are for the tMOPSO runs with the greatest hypervolume.

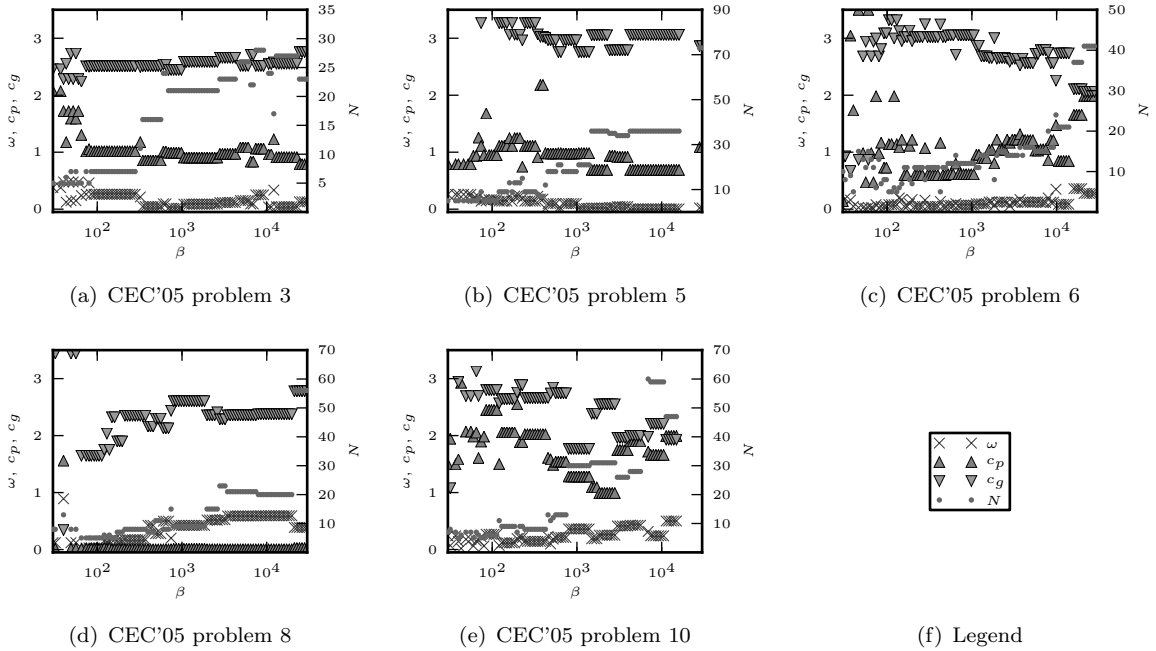


(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6 (d) CEC'05 problem 8 (e) CEC'05 problem 10

(f) CEC'05 problem 3 (g) CEC'05 problem 5 (h) CEC'05 problem 6 (i) CEC'05 problem 8 (j) CEC'05 problem 10

(k) CEC'05 problem 3 (l) CEC'05 problem 5 (m) CEC'05 problem 6 (n) CEC'05 problem 8 (o) CEC'05 problem 10

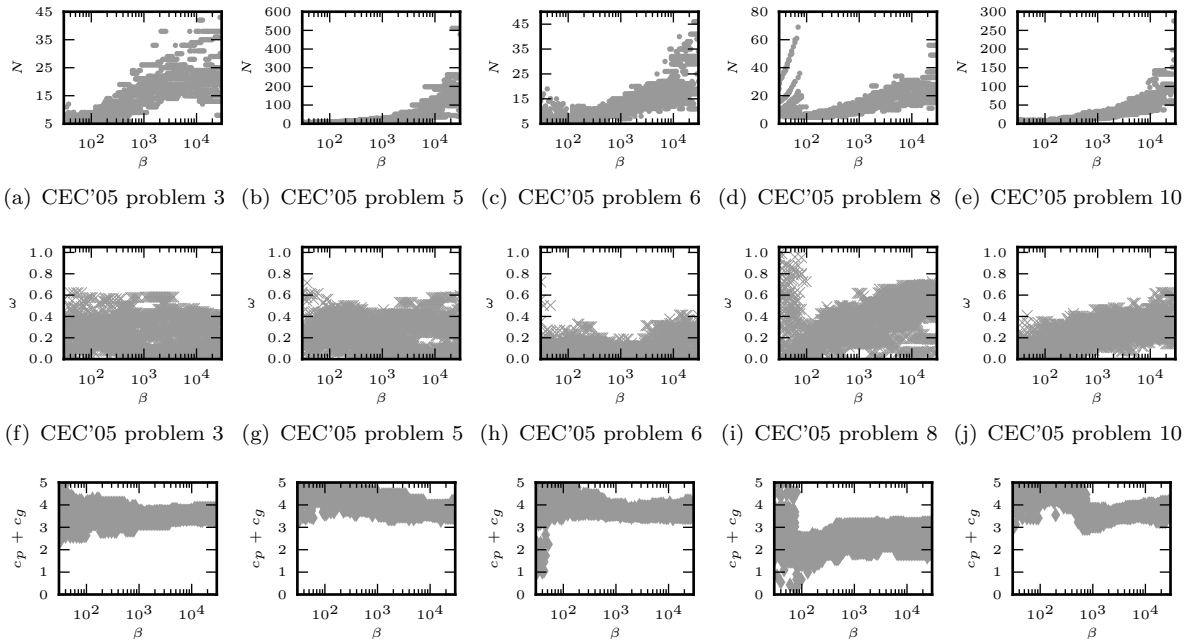
Figure 2.13: Scatter plots of the combined results from all of tMOPSO's independent tuning runs, showing the optimal population size versus the OFE budget available. Each subfigure shows the results for DE tuned to a different CEC'05 optimization problem.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

Figure 2.14: tMOPSO's results for tuning PSO under multiple OFE budgets, where the results shown are for the tMOPSO runs with the greatest hypervolume.

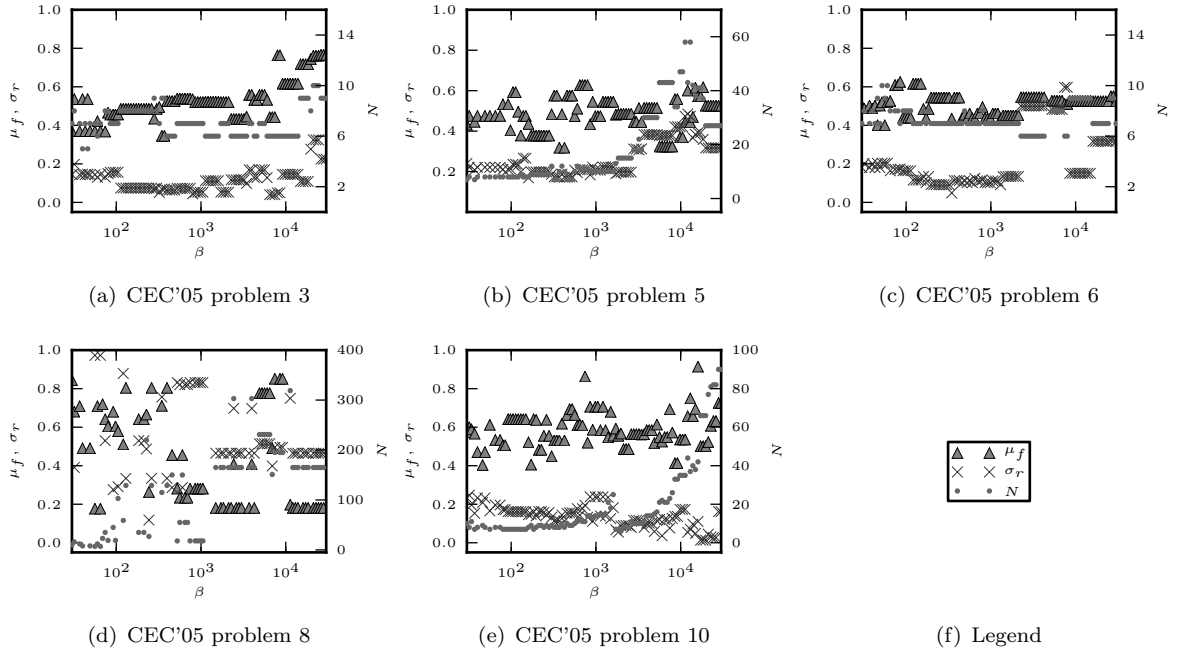


(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6 (d) CEC'05 problem 8 (e) CEC'05 problem 10

(f) CEC'05 problem 3 (g) CEC'05 problem 5 (h) CEC'05 problem 6 (i) CEC'05 problem 8 (j) CEC'05 problem 10

(k) CEC'05 problem 3 (l) CEC'05 problem 5 (m) CEC'05 problem 6 (n) CEC'05 problem 8 (o) CEC'05 problem 10

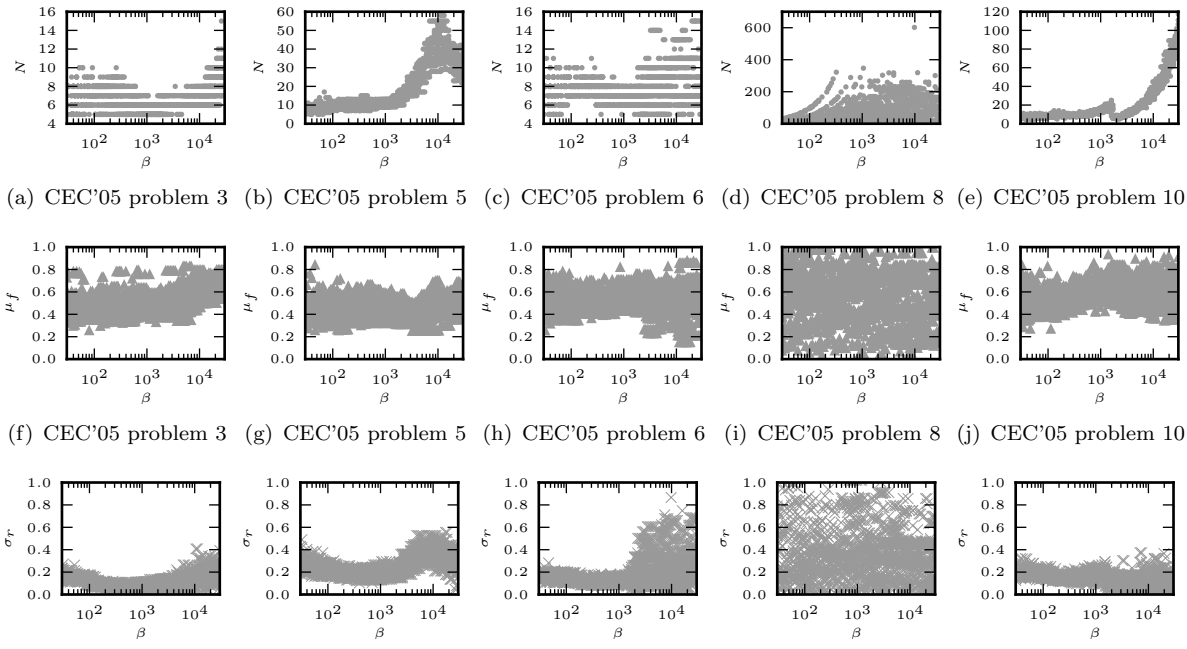
Figure 2.15: Scatter plots of the combined results from all of tMOPSO's independent tuning runs, showing the optimal swarm size and acceleration constant sum versus the OFE budget available. Each subfigure shows the results for PSO tuned to a different CEC'05 optimization problem.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6

(d) CEC'05 problem 8 (e) CEC'05 problem 10 (f) Legend

Figure 2.16: tMOPSO's results for tuning CMA-ES under multiple OFE budgets, where the results shown are for the tMOPSO runs with the greatest hypervolume.



(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6 (d) CEC'05 problem 8 (e) CEC'05 problem 10

(f) CEC'05 problem 3 (g) CEC'05 problem 5 (h) CEC'05 problem 6 (i) CEC'05 problem 8 (j) CEC'05 problem 10

(k) CEC'05 problem 3 (l) CEC'05 problem 5 (m) CEC'05 problem 6 (n) CEC'05 problem 8 (o) CEC'05 problem 10

Figure 2.17: Scatter plots of the combined results from all of tMOPSO's independent tuning runs, showing the optimal population size versus the OFE budget available. Each subfigure shows the results for CMA-ES tuned to a different CEC'05 optimization problem.

a search which is as effective on Ackley type problems as the tuned PSO algorithm is. Should the latter case be true, additional search mechanics could be added to DE and CMA-ES as to improve their tunability to Ackley type problems.

tMOPSO is a bi-objective tuning algorithm, and is therefore limited to only tuning an optimization algorithm to one problem at a time under multiple OFE budgets. In order to tune an optimization algorithm to multiple problems under multiple OFE budgets holistically, an algorithm designed for four or more tuning objectives is required. In the next chapter, the principles upon which tMOPSO is built are extended as to develop a many objective tuning algorithm.

CHAPTER 3

MANY OBJECTIVE TUNING USING BI-OBJECTIVE DECOMPOSITION

To aid control parameter studies, a new evolutionary algorithm named MOTA (many objective tuning algorithm) is proposed. MOTA aims to efficiently tune an optimization algorithm according to multiple performance measures over a range of OFE budgets. Even though no such algorithm has been proposed before, tuning an optimization algorithm to multiple performance measures for multiple OFE budgets could be achieved by using existing tuning algorithms. Specifically, existing tuning algorithms can be used to solve multiple subproblems, where each subproblem is focused on a different performance measure preference articulation. However, segregating a multi-objective problem in this manner is wasteful since no information is shared between the created subproblems. Consider two subproblems each focused on tuning an algorithm to the same problem, only at different OFE budgets. Or consider the case where common CPV trends exist between these subproblems, such as a larger optimal population size as the OFE budget increases. For these scenarios, information flow between subproblems should be beneficial to the tuning process. MOTA overcomes these segregation limitations through the use of multi-objective optimization.

The design of MOTA is motivated by the in-depth control parameter studies a many objective tuning algorithm would allow. Consider studies investigating robust or generalist CPV tuples which perform well over numerous problems (Smit and Eiben, 2010b). Multi-objective tuning can efficiently search for these robust CPV tuples, by solving a tuning problem with an objective corresponding to each problem that the robust CPVs are required to perform well on. After tuning is completed, the generalist CPV tuples can be found by examining the CPVs tuples found during the multi-objective optimization, each of which is optimal for a different trade-off compromise amongst the tuning objectives. Furthermore, common practice when assessing a multi-objective algorithm's performance is to make use of a series of unary performance indicators (Zitzler et al., 2003), each of which measures a different aspect of the solution quality. As such, tuning according to multiple performance indicators would allow multi-objective al-

gorithms to be tuned more holistically compared to tuning them according to one performance metric only. Moreover, even if an optimization algorithm is to be tuned to multiple problems separately as to determine CPVs well-suited to individual problems only, tuning an algorithm to all problems congruently may result in a higher efficiency compared to handling each problem in isolation, since common CPV trends may be present.

The outline of this chapter is as follows: related work and MOTA's contribution is first discussed in Section 3.1. The MOTA algorithm is then described in Section 3.2. Thereafter, Section 3.3 presents the numerical setup used to assess MOTA's performance, with the results from those numerical experiments following in Section 3.4.

3.1 Related Work

The proposed tuning algorithm is related to the fields of control parameter tuning and many objective optimization. Currently multi-objective tuning algorithms can be split into two groups, namely tuning to multiple problems each for a single termination criterion, and tuning algorithms designed to tune to a single problem under multiple termination criteria. Smit et al. (2010) proposed the M-FETA algorithm which is designed for tuning an optimization algorithm to multiple problems, each using one termination criterion. With regard to tuning under multiple OFE budgets the Flexible Budget Method Branke and Elomari (2012) and tMOPSO have been proposed. Here, the MOTA algorithm is proposed for tuning an algorithm to multiple performance measures under multiple OFE budgets. Such a tuning problem has an utility measure consisting of at least three objectives. When the utility measure consists of four or more objectives, then MOTA needs to solve a many objective optimization problem.

For optimization problems which consist of two or three objectives, multi-objective evolutionary algorithms typically aim to determine a finite evenly spaced set of non-dominated decision vectors as to accurately approximate the entire PF. However, approximating the entire PF for many objective optimization is intractable for two reasons. Firstly, the computational overhead of maintaining the PF approximations (Mostaghim and Teich, 2005) grows linearly as the size of the approximation increases. Consequently, the computational overhead requirements are too high to approximate the entire PF of many objective optimization, since the size of the set required to represent the entire PF grows exponentially with the number of objectives. Secondly, even if a huge Pareto-optimal front approximation (PFA) could be maintained efficiently, the limiting factor would be the OFE budget of the multi-objective optimization algorithm. Suppose that an ultimate multi-objective optimization algorithm existed, which with every new decision vector evaluation is able to find a new non-dominated decision vector. Even this ultimate algorithm's PFA would be limited to the size of the OFE budget assigned to it.

Many objective optimization algorithms therefore do not try to approximate the entire PF, but rather make use of other criteria in addition to Pareto dominance to guide the optimization process. A commonly used approach to differentiate between Pareto non-dominated decision vectors during the course of an optimization run, is to make use of performance indicators (Zitzler and Künzli, 2004). Consider the SMS-EMOA algorithm proposed by Beume et al. (2007), which uses the hypervolume (HV) performance indicator in conjunction with Pareto dominance

in order to optimize a multi-objective problem. Alternatively, Di Pierro et al. (2007) proposed a preference ordering strategy, which considers a decision vector's dominance status according to various subsets of objectives, thereby allowing for further differentiation. Then there are decomposition based approaches (Zhang and Li, 2007), for which the multi-objective problem is divided into subproblems, which are all solved simultaneously. All these many objective approaches ultimately require some *a priori* input from the practitioner, as to which sections of the PF, or preference articulations are more important than others. The indicator based approaches favor decision vectors aligned with indicators chosen, while preference ordering strategies favor decisions close to the center or the edges of the PF depending on the parameters specified, and decomposition based approaches focus on areas of the PF specified in the subproblem construction. This *a priori* input is undesirable as it breaks from the clean *a priori* free approach followed when three or less objectives are optimized. However due to the intractability of approximating the entire PF for many objective optimization problems, some *a priori* input is required.

Objective reduction approaches can in certain scenarios assist with many objective optimization. For certain applications it may occur that not all of the objectives are in conflict, in which case some objectives can be disregarded without changing the PS. For such scenarios, objective reduction approaches are able to identify and remove redundant objectives as to make the optimization problem easier to solve. Brockhoff and Zitzler (2009) cover both the theoretical aspects of objective reduction, as well as presenting ϵ-based objective reduction approaches. More recently, Saxena et al. (2013) presented a framework based on principal component analysis and maximum variance unfolding for objective reduction. An aspect of objective reduction approaches which can prove useful even when applied to non-redundant problems, is the ability of these approaches to identify objectives which are only slightly in conflict. These slightly conflicting objectives can be disregarded without major changes to the PS. Objective reduction is not incorporated into the proposed MOTA algorithm, as MOTA is designed for tuning problems where all the objectives are in conflict.

3.2 MOTA Algorithm

A decomposition based approach is used by MOTA to perform many objective tuning. What distinguishes the decomposition approach used by MOTA from previous approaches, is that instead of decomposing the multi-objective problem into single objective subproblems, the multi-objective problem is decomposed into bi-objective subproblems. A bi-objective decomposition is particularly well-suited when tuning stochastic algorithms under multiple OFE budgets, as will become clear as the core elements of the MOTA algorithm are presented, beginning with the tuning problem formulation used.

3.2.1 Tuning Problem Formulation

The tuning problem formulation solved by MOTA makes use of a decision vector consisting of control parameter values $v_1, v_2, \ldots, v_n$ together with an auxiliary variable β_a, where β_a specifies at which OFE budget the $v_1, v_2, \ldots, v_n$ CPV tuple is to be assessed. MOTA's tuning problem

formulation uses a multi-objective utility measure as to tune an optimization algorithm to multiple utility indicators for multiple OFE budgets. The multi-objective utility measure $\mathbf{u}$ which MOTA minimizes is defined as

$$\mathbf{u} = \begin{bmatrix} \beta_a \\ u_1 \\ u_2 \\ \dots \\ u_{n_u} \end{bmatrix}, \tag{3.1}$$

where $u_1, u_2, \dots, u_{n_u}$ are utility indicator values for which lower values indicate better performance, and β_a is the OFE budget used when determining those $u_1, u_2, \dots, u_{n_u}$ values. Since lower utility indicator values indicate better performance, and a greater number of OFEs allows for lower $u_1, u_2, \dots, u_{n_u}$ values, the β_a objective is in conflict with the $u_1, u_2, \dots, u_{n_u}$ objectives. As such, formulating the tuning problem in this manner allows MOTA to directly incorporate sensitivity to OFE budgets into the multi-objective problem it solves.

The choice of utility indicators for $\mathbf{u}$ depends on the control parameter study being performed. When tuning a single objective algorithm to multiple problems, a sensible choice for the utility indicators would be the lowest solution error achieved for each of those problems. Alternatively when tuning a multi-objective optimization algorithm, a utility indicator for each unary performance indicator of interest could be used.

3.2.2 Specialization for Algorithms whose Utility Indicator Values need to be Numerically Approximated using Sample Runs

Analytical expressions for the utility indicator values as a function of the specified CPV tuple and OFE budget are rarely available. As such, common practice entails numerical calculation of utility indicator values by running the algorithm being tuned from initialization to the assessment OFE budget, using the CPV tuple being assessed. Calculating a utility indicator value in this manner, therefore also allows for determining utility indicator values of the CPV tuple being assessed for OFEs lower than the assessment OFE budget without performing additional sample runs. Consider calculating an arbitrary utility indicator's (u_i's) values using a sampling run, for an evolutionary algorithm with population size of N. If the sample run is set up to record the best solution found after each iteration, i.e. every N OFEs, utility values can then be determined for each OFE usage all the way up to the assessed OFE budget, β_a. Expressed symbolically, one sampling run can be used to generate the following :

$$u_i(N) \rightarrow u_i(2N) \rightarrow \dots \rightarrow u_i(\beta_a), \tag{3.2}$$

where $u_i(j)$ is the utility indicator value after j OFEs. Another factor to consider when performing a sampling run to assess a CPV tuple's utility, is that utility values at OFE budgets higher than β_a, can be determined at a reduced cost compared to calculating them from scratch.

Due to computational overhead considerations and storage requirements, tuning practitioners are normally not interested in determining effective CPV tuples for *every* OFE budget less

then the maximum OFE budget of interest, $\beta_{\max}$. Normally, a tuning practitioner is only interested in a subset of OFE budgets $\in \{1, 2, \ldots, \beta_{\max}\}$, such as OFE budgets logarithmically spaced between the minimum OFE budget of interest and $\beta_{\max}$. Given this consideration, MOTA calculates the following utility indicator values

$$u_i(\beta) \forall \beta \in B : \beta \leq \beta^+ \tag{3.3}$$

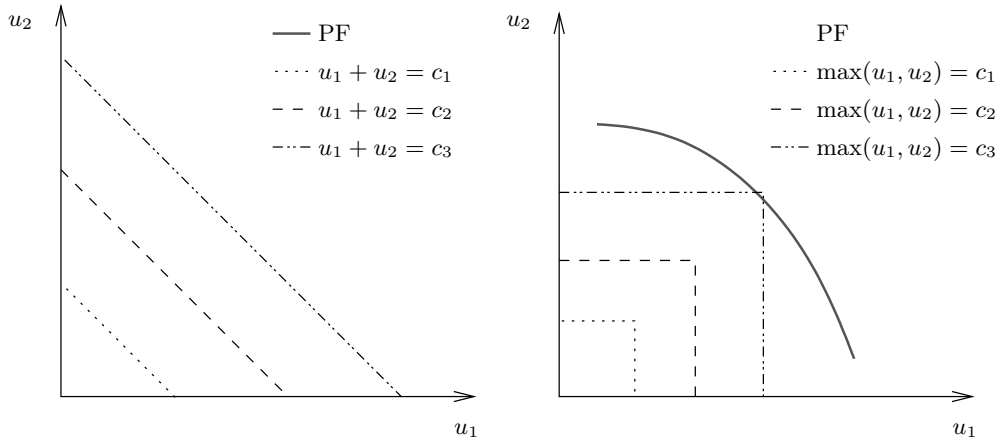
where B is the target OFE budgets selected by the tuning practitioner, and the overshoot budget β^+ specifies the maximum OFE budget for which the utility indicator values are to be calculated from the sampling runs. For MOTA, β^+ is calculated according to a user specified function of β_a, for example $\beta^+ = 1.6\beta_a + 100$. The optimal function for determining β^+ is expected to be problem specific, but is also not expected to drastically alter performance, due to the noise handling strategy used by MOTA, a strategy whose description is postponed until later in this Section.

By making use of the additional utility indicator values from the sample runs, MOTA breaks from traditional multi-objective optimization where one decision vector evaluation results in one objective vector. Instead, one decision vector evaluation by MOTA results in multiple objective vectors. In particular, each CPV tuple assessment results in a multi-dimensional line of objective function values, where the OFE budget objective can be considered as the independent variable. Given this one-to-many relation, a 2-D decomposition strategy is used by MOTA.

3.2.3 Bi-objective Decomposition

Solving a problem via decomposition, entails expressing the original problem as a series of subproblems, which when solved give the solution to the original problem. In the context of multi-objective optimization, a variety of approaches exists for decomposing a multi-objective problem into single-objective subproblems (Zhang and Li, 2007). Decomposition however need not be limited to breaking a problem down into single-objective subproblems. Zhang and Li (2007) argued that it is beneficial to decompose problems to single-objective subproblems, since a significant amount of work has been done for evolutionary computation in single objective optimization, and therefore decomposition into single-objective subproblems is favorable since it allows for all this previous work to be utilized. Following this same argument, it is viable to decompose a problem into bi- or tri-objective subproblems, because a substantial amount of work and success has been achieved when optimizing problems with only two or three objectives (Deb et al., 2002; Zitzler et al., 2001).

In the case of MOTA, bi-objective decomposition is used as it is well suited to processing the additional utility indicator values generated from the sampling runs. The generalized objective function for the bi-objective decompositions which are to minimized, is comprised of one objective consisting of a scalarization of the $u_1, u_2, \ldots, u_{n_u}$ utility indicator values, and the other objective is the OFE budget used to calculate those utility indicator values. Two commonly used scalarization approaches are the aggregated or weighted sum approach and the Tchebycheff approach (Zhang and Li, 2007). Both of these approaches make use of a weights vector $\mathbf{w}$ in



(a) Weighted sum approach (b) Tchebycheff approach

Figure 3.1: Illustration of the weighted sum and the Tchebycheff scalarization approaches. In the illustration, contour lines of equal value according to the scalarization approach are shown for the values c_1, c_2 and c_3, where $c_1 < c_2 < c_3$.

the scalarization process. Specifically, the weighted sum scalarized value $\hat{u}_w$ is determined as:

$$\hat{u}_w = \sum_{i=1}^{n_u} w_i u_i, \tag{3.4}$$

and the Tchebycheff scalarized value $\hat{u}_T$ for minimization problems is determined as:

$$\hat{u}_T = \max\{w_i(u_i - z_i) \quad \forall\, i \in 1, 2, \ldots, n_u\} \tag{3.5}$$

where the z_i values correspond to a chosen reference point. In order to make the process of selecting $\mathbf{w}$ easier for tuning practitioners, MOTA by default normalizes the objective values passed to the scalarize functions. Specifically, the u_i values are normalized between the Ideal and Nadir points of MOTA's current PFA. For the case when the objective values are normalized, zero values are used for all the z_i reference values.

The choice of scalarization approach for the utility indicator values should be made according to the PF characteristics, and according to preference articulations of interest. The Tchebycheff approach is able to handle both convex and concave PFs, while the weighted sum approach can only handle concave PFs, as illustrated in Figure 3.1. However, the Tchebycheff approach is more prone than the weighted sum approach to being controlled by a single objective only. If the user selects the Tchebycheff approach for the j'th subproblem, the corresponding bi-objective minimization function which needs to be minimized $\mathbf{u}_j^{2D}$ is defined as:

$$\mathbf{u}_j^{2D} = \begin{bmatrix} \max\{w_i\, \tilde{u}_i \quad \forall\, i \in (1, 2, \ldots, n_u)\} \\ \beta_a \end{bmatrix}, \tag{3.6}$$

where $\tilde{u}_i$ are the normalized utility indicator values.

Another key aspect to decomposition based approaches is neighborhoods. Neighborhoods are vital since they are used to share information between subproblems, thereby differentiating

decomposition based approaches from approaches were subproblems are optimized in isolation from each other. Here, neighborhoods are split into two categories,

- candidate generation neighborhoods: when generating a candidate decision vector for a target subproblem, this neighborhood specifies the additional subproblems from which information is used for operations such as crossover and mutation, and
- update neighborhoods: after evaluating a decision vector generated for a target subproblem, the resulting objective function values are also used to update the solutions of the subproblems in this neighborhood.

MOTA allows for the use of different neighborhoods for the purposes of generating candidate decision vectors and updating subproblem solutions. Having different neighborhoods for these two operations, allows for a flexibility particularly well suited for tuning optimization algorithms. Consider tuning a single objective optimization algorithm to multiple problems over multiple OFE budgets, with the focus on determining specialist CPVs well suited to each problem on its own. A sensible neighborhood configuration for this tuning problem would be to have a large neighborhood for candidate generation, together with zero-sized update neighborhoods. The large candidate generation neighborhood should be beneficial, since it allows for the CPV candidate generation process to exploit trends observed from other subproblems, while the zero-sized update neighborhoods save computational resources since only one of n_u objectives needs to be evaluated.

The next feature of MOTA which is discussed is the noise handling strategy. In tuning, a noise handling strategy deals with the uncertainty resulting when utility indicator values need to be approximated, as is typically the case when tuning stochastic algorithms.

3.2.4 Handling the Noise Resulting from Tuning a Stochastic Algorithm

When tuning optimization algorithms with stochastic elements, standard utility indicator values become probabilistic distributions. In the context of parameter tuning under multiple OFE budgets, the characteristics of the probability density functions (PDFs) of these utility indicator distributions vary depending upon the location in the objective space, as illustrated in Figure 2.2. The differing PDF characteristics throughout the objective space, rule out many specialized noise (Bui et al., 2009) or uncertainty (Xi et al., 2012) handling strategies, which assume a uniform uncertainty distribution throughout the objective space. Based on tMOPSO (Dymond et al., 2014), MOTA uses a preemptively terminating resampling strategy whereby the sampling gathering processes is interrupted if Mann-Whitney U tests (MWUTs; Conover, 1999) indicate that the decision vector being assessed is unlikely to result in an improvement on the current approximation of the PF.

MOTA's noise handling procedure starts with a group of decision vectors X, where each $\mathbf{x} \in X$ has an associated CPV tuple, assessment OFE budget, and target subproblem. Initially a small number of samples are generated for each $\mathbf{x}$, for the OFE budgets $\mathbf{x}_B$. As outlined in Subsection 3.2.2, $\mathbf{x}_B$ is controlled by the user specified target OFE budgets B and the overshoot function β^+ as

$$\mathbf{x}_B = \{\beta \forall \beta \in B : \beta \leq \beta^+\}. \tag{3.7}$$

Re-sampling interruption checks are then conducted against $\mathbf{x}$'s subproblem and $\mathbf{x}$'s update neighborhood. Specifically, if j is the subproblem index, and T_u is the set of indexes of the subproblems in $\mathbf{x}$'s neighborhood together with j, then the approximation of the j'th subproblem's bi-objective decomposition $\mathbf{u}_j'^{2D}$ is discarded if

$$P_k \leq_\alpha \mathbf{u}_j'^{2D} \quad \forall\, k \in T_u \tag{3.8}$$

where $\leq_\alpha$ denotes likely to be dominated given the confidence level of α, and P_k is the k'th subproblem's PFA.

Two different options are available for conducting Pareto non-dominance likelihood checks:

1. removing the largest OFE budget in $\mathbf{x}_B$ until an OFE budget is reached for which $\mathbf{x}$ is not likely to be dominated by all P_k for $k \in T_u$, and
2. checking all OFE budgets in $\mathbf{x}_B$, and eliminating the OFE budgets for which the $\mathbf{u}_j'^{2D}$ decompositions are likely to be dominated by P_k for all $k \in T_u$.

The choice of approach should be made according to the relative computational cost of calculating utility indicator values from a sample run and that of performing a sampling run. For the case where a cheap utility indicator value is used, such as the objective solution error achieved by a single objective algorithm, the option of reducing the maximum OFE is sensible. Alternatively, when an expensive utility indicator such as HV is used, then the second approach is more appropriate. Re-sampling interruption checks continue until either $\mathbf{x}_B$ is empty or the desired re-sampling size n_s is reached. If the desired re-sampling size n_s is reached, the approximated utility values are used to update the T_u subproblems' PFAs. MOTA users control the aggressiveness of the resampling interruption through two control parameters, namely the number of sample increments between resampling interruption checks, Δ_{n_s}, and the interruption confidence level used, α.

The bi-objective nature of the subproblems is exploited as in Section 2.2.3, in order to efficiently perform PFA dominance and dominance likelihood checks. A flow chart for the CPV tuple assessment procedure for the *check all* noise handling approach is shown in Figure 3.2.

3.2.5 Algorithm Overview

MOTA tunes an optimization algorithm to multiple criteria over a range of OFE budgets, using the aforementioned concepts. The decision space MOTA searches is set up according to the tuning formulation presented in Subsection 3.2.1, where each decision vector is of the following form:

$$\mathbf{x} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ \ln \beta_a \end{bmatrix}, \tag{3.9}$$

where $v_1, v_2, \ldots, v_n$ are the real CPVs optimized, and β_a is the OFE budget at which the $v_1, v_2, \ldots, v_n$ CPV tuple is to be assessed. The natural logarithm of β_a is optimized in place

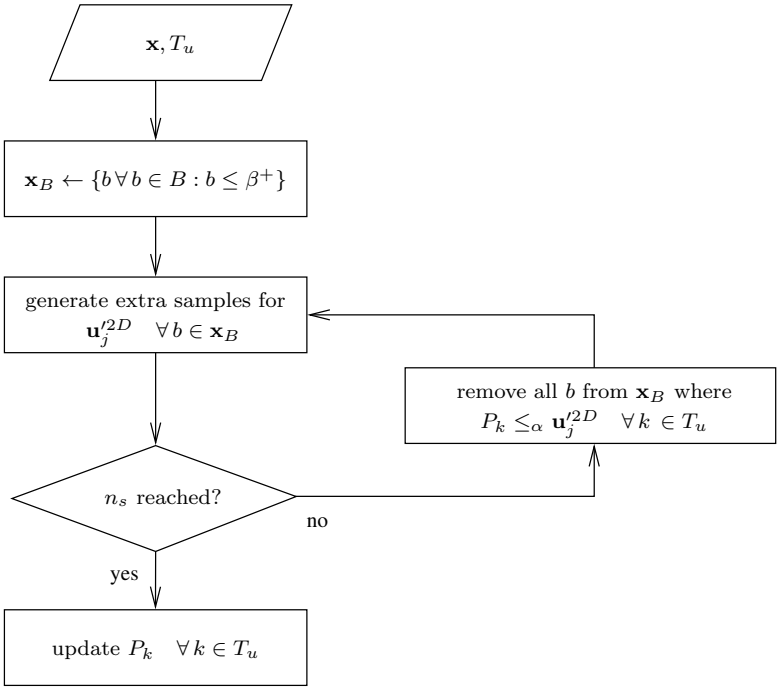


Figure 3.2: Flow chart of MOTA's CPV tuple assessment procedure when computationally expensive utility indicator values are calculated.

of β_a directly, due to logarithmic CPV trends typically observed when tuning algorithms under multiple OFE budgets (Dymond et al., 2014).

Tuning begins with MOTA randomly generating decision vectors throughout the CPV decision space for each of the tuning subproblems. Each initial decision vector $\mathbf{x}_0$ is generated as:

$$\mathbf{x}_0 = \mathbf{I}_l + \mathbf{r}() \circ (\mathbf{I}_u - \mathbf{I}_l), \tag{3.10}$$

where $\mathbf{I}_l$ is the lower initialization bound, $\mathbf{I}_u$ is the upper initialization bound, $\circ$ is the Hadamard product operator, and $\mathbf{r}()$ is a function returning a vector whose individual element values are randomly generated independently of each other between 0 and 1 with a uniform probability density. These initial decision vectors are evaluated as outlined in the Subsection 3.2.4 as to generate the initial subproblem PFAs. A limitation of the initialization approach in (3.10) is the high computational cost resulting when tackling overly contrained problems. Such overly constrainted problems are however not typical for tuning where cheap CPV inequality constraints are the norm.

After initialization, MOTA uses operators based upon differential evolution (DE; Storn and Price, 1997) to generate new candidate decision vectors. DE operators are used given the algorithm's past successes on a vast range of problems (Das and Suganthan, 2010). Before the manner in which the DE operators are extended into the context of the MOTA tuning formulation is explained, the traditional single objective DE operators are first described. DE has numerous strategies available, such as *rand/1/bin* and *best/1/bin*. The DE *rand/1/bin* candidate decision vector generation process for the i'th member of the population begins with

generating a mutant vector $\mathbf{m}_i$ as follows

$$\mathbf{m}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} + \mathbf{x}_{r_3}), \tag{3.11}$$

where $\mathbf{x}_k$ is the decision vector of the k'th member of the population, the indexes r_1, r_2 and r_3 are randomly selected, and F is the user-specified scaling factor. The population indexes r_1, r_2 and r_3 are randomly selected with each member from the population having an equal likelihood of selection, subject to all the indexes being different and none being equal to i. After mutation, crossover takes place to generate the candidate decision vector for the i'th member of the population $\mathbf{x}_i^c$, as follows

$$x_{i,k}^c = \begin{cases} m_{i,k} & \text{if } r() < C_r \text{ or } k = k_f \\ x_{i,k} & \text{otherwise,} \end{cases} \tag{3.12}$$

where k is the dimension index, $r()$ is a function which returns a number randomly between 0 and 1 with a uniform probability density, C_r is the user-specified crossover rate, k_f is the dimension which is forced to crossover, and $\mathbf{x}_i$ is the i'th population member's current decision vector. Further information about DE and its strategies can be found in Das and Suganthan (2010).

MOTA's candidate decision vector generation process for the i'th subproblem, begins by randomly selecting three subproblems indexes s_1, s_2 and s_3, for the purposes of mutation. The pool from which s_1, s_2 and s_3 are randomly selected with equal likelihood, consists of the indexes of the subproblems in the i'th subproblem's candidate generation neighborhood, together with i'th subproblem's index, i. Contrary to the *rand/1/bin* strategy, the constraints are omitted that s_1, s_2 and s_3 all be unique and not equal to i. These constraints are omitted since the Pareto Set of a subproblem consists of multiple decision vectors which can be used for the generation of a mutant vector. Three decision vectors, $\mathbf{x}_{s_1}$, $\mathbf{x}_{s_2}$ and $\mathbf{x}_{s_3}$ are then selected from the PFAs of the s_1, s_2 and s_3 subproblems respectively, according to a target improvement OFE budget, β_t. β_t is selected randomly from the target OFE budgets B with each element in B having equal likelihood of selection. $\mathbf{x}_{s_1}$ is selected as the decision vector from the PFA of the s_1 subproblem which performs best for an OFE budget of β_r, where β_r is perturbed about β_t as follows

$$\ln \beta_r = \ln \beta_t + r_g() \cdot \beta_\delta \cdot (\ln B_{\max} - \ln B_{\min}), \tag{3.13}$$

where β_δ is the user-specified perturbation factor with $\beta_\delta \in [0, 1]$, r_g is a function which returns a value randomly generated using a Gaussian distribution with standard deviation of 0.25 and a mean of 0.0, $B_{\min}$ is the minimum OFE budget in B, and $B_{\max}$ is the maximum OFE budget in B. $\mathbf{x}_{s_2}$ and $\mathbf{x}_{s_3}$ are selected in the same manner, using the same β_t but different β_r values as to exploit any CPV versus OFE budget trends which may be present. Based upon the DE *best/1/bin* strategy, MOTA's mutant vector is generated as

$$\mathbf{m}_i = \mathbf{x}_{s_1} + \mathbf{r}() \circ F(\mathbf{x}_{s_2} + \mathbf{x}_{s_3}), \tag{3.14}$$

where the $\mathbf{r}()$ term is added to promote search diversity (Salehinejad et al., 2014). DE Crossover

is then conducted between the resulting mutant vector, and the decision vector from the i'th subproblem which is optimal for an OFE budget of β_t, as in (3.12). Lastly, the assessment OFE budget of the generated candidate decision vector's is set to β_t.

Constraint handling is achieved by regenerating candidate vectors until all constraints are satisfied. Although this approach is not viable for applications consisting of computationally expensive constraints, it is acceptable for tuning applications since tuning constraints are normally computationally cheap. MOTA does not enforce nor have a specialized strategy for handling bound constraints, since for many tuning problems sensible CPV bounds are difficult to determine *a priori*. Additionally, MOTA has an internal constraint where it is required that the candidate decision vector be different from the $\mathbf{x}_i$ vector it is trying to improve upon. This internal constraint is necessary for the beginning stages of MOTA's tuning optimization, for which the subproblems PFAs consists of a low number of unique decision vectors. If for a given subproblem, candidate generation using the DE operations in (3.14) and (3.12) fails to satisfy the constraints 10 times in a row during a single iteration, then a randomly generated valid decision vector is used, as outlined in (3.10). Once the candidate decision vectors are generated for all of the subproblems, these candidate decision vectors are assessed to update the subproblem PFAs as outlined in Subsection 3.2.4.

Generation of the candidate decision vectors continues until all of the subproblems become inactive. A subproblem becomes inactive when one of its termination criteria is satisfied, signaling that no more candidate vectors should be generated for that subproblem. If a subproblem is inactive but is in the update neighborhood of an active subproblem the inactive subproblem's PFA is still updated, when the active subproblem's candidate vector is assessed. Per subproblem termination or inactivity criteria are appealing since it allows for greater control compared to making all subproblems inactive at the same time. For example, per subproblem inactivity allows for allocation of differing amounts of computational resources for each subproblem. Another example is adding termination criteria whereby a subproblem becomes inactive if no substantial improvement is made to the PFA over the last couple of iterations.

Our implementation of MOTA is available in the optTune Python package[1], and the pseudocode is given in Figure 3.3.

3.3 Numerical Setup

Numerical experiments are conducted to gauge the effectiveness of MOTA. Experiments are chosen based upon the potential benefits of many objective tuning, benefits which motivated MOTA's development. In the introduction to this chapter it was proposed that a many objective tuning algorithm could

- be more efficient at tuning an optimization algorithm to each problem of a test suite compared to tuning an algorithm to each problem instance in isolation,
- be better suited to determining generalist CPV tuples which perform well over an entire problem test suite, and would

[1] http://code.google.com/p/opt-tune-python-package/.

```
procedure MOTA
    for s_p ∈ subproblems do
        repeat
            x_0                                                    ▷ (3.10)
        until x_0 valid
    end for
    evaluate all the generated x_0                                 ▷ Subsection 3.2.4
    i ← 1
    while a subproblem is active do                                ▷ main loop
        for s_p ∈ active subproblems do
            repeat
                β_r                                                ▷ (3.13)
                m                                                  ▷ (3.14)
                x_c                                                ▷ (3.12)
            until x_c valid
        end for
        evaluate all the generated x_c                             ▷ Subsection 3.2.4
        i ← i + 1
    end while
end procedure
```

Figure 3.3: MOTA pseudocode

- be able to tune MOEAs more holistically by tuning them according to multiple unary performance metrics simultaneously.

MOTA is benchmarked with regards to the first two statements, namely determining specialist CPVs which are well suited to a single problem, and with regard to determining generalist CPVs which perform well over an entire test suite. For brevity, the tuning of multi-objective algorithms to multiple unary performance metrics is left for future work.

3.3.1 Tuning Problems Used

The tuning problems used are built around the commonly used ZDT (Zitzler et al., 2000), DTLZ (Deb et al., 2005), and WFG (Huband et al., 2006) multi-objective test problem suites. Since the numerical experiments entail tuning algorithms to these problem suites, lower-than-normal dimensional versions of WFG and DTLZ problems are used, to ensure that the computational costs are manageable. For the bi-objective ZDT problems, algorithms are tuned to ZDT problems $1, 2, 3, 4$ and 6, where the standard setup of 30 decision variables are used for problem 1,2 and 3, while 10 decision variables are used for problem 4 and 6. The 5^{th} ZDT problem is omitted as this is commonly done in practice. Regarding the WFG problems, 2 position decision variables, 10 distance decision variables, and 2 objectives are used for all the problems. For the DTLZ problems the number of decision variables is kept at the commonly used values of 7, 12, 12, 12, 12, 12 and 22 for problems 1 through 7 respectively, while the number of objectives is reduced from the standard 3 to only 2.

Selected multi-objective optimization algorithms are tuned to these problem suites, according to inverted generational distance (IGD; Zhang et al., 2008), which is able to measure both convergence to and spread across the true PF. Initially tuning according to HV was considered,

but was later disregarded due to the problem specific effort of selecting HV reference points sensitive to the performance at low OFE budgets. The tuning objectives are to minimize the IGD for each problem in the test suite the algorithm is being tuned to, while minimizing the OFEs used. Therefore, tuning an algorithm to the 9 WFG problems entails solving a tuning problem with the 10 objectives:

$$F = \begin{bmatrix} IGD_{WFG1} \\ IGD_{WFG2} \\ \dots \\ IGD_{WFG9} \\ \beta_a \end{bmatrix}, \tag{3.15}$$

where IGD_{WFG1} is the IGD achieved on the first WFG problem given the CPV tuples being assessed and an OFE budget of β_a, similarity for IGD_{WFG2} to IGD_{WFG9}. Regarding the OFE budgets which algorithms are to be tuned under, 51 OFE budgets logarithmically spaced between 10 to 10 000 are used.

Separate problems are constructed for each algorithm tuned, as to have a tuning problem which entails determining specialist CPVs only, and to have tuning problems which entail determining generalist and specialist CPVs together. For the specialist tuning problems, the bi-objective PF sections of interest correspond to each problem in the test suite in isolation together with the OFE budget objective. Continuing to use the WFG test suite example, the nine utility weight vectors of $[1, 0, \dots, 0]$, $[0, 1, 0, \dots, 0]$, $\dots$, $[0, \dots, 0, 1]$ would be used for the specialist problem. For the generalist tuning problems, additional preference articulations are added to determined CPV tuples which perform well for all utility objectives, i.e. $[1, 1, \dots, 1]$, and for all leave-one-out combinations, i.e. $[0, 1, \dots, 1]$, $[1, 0, 1, \dots, 1]$, $\dots$, $[1, \dots, 1, 0]$. All of the generalist subproblems make use of weighted sum scalarization in the normalized objective space for their bi-objective decomposition. The leave-one-out preference articulations are added as to allow for scrutinization of the $[1, 1, \dots, 1]$ articulation, as to determine if one objective has a disproportionate effect in spite of scalarization occurring using a normalized objective space.

3.3.2 Algorithms Tuned

The algorithms tuned to the ZDT, DTLZ and the WFG problems are the commonly used second version of non-dominated sorting genetic algorithm (NSGA-II; Deb et al., 2002) and the multi-objective evolutionary algorithm based on decomposition (MOEA/D; Zhang and Li, 2007). The NSGA-II and MOEA/D implementations from version 4.3 of the jMetal software package (Durillo and Nebro, 2011)[2] are used in these experiments, with slight modifications. The default NSGA-II and MOEA/D implementations in jMetal 4.3 handle the OFE budget termination criteria in different manners. The NSGA-II implementation checks against the OFE budget constraint after every pair of offspring are created, i.e. every 2 OFEs. In contrast the MOEA/D implementation checks against the OFE budget constraint at the end of each iteration after all the subproblem candidates have been evaluated. The NSGA-II implementation is

[2] http://jmetal.sourceforge.net/

modified so that the same behavior as MOEA/D is achieved, so that the tuning results can be compared against previous studies (Dymond et al., 2013).

Selected real and integer CPVs are tuned for NSGA-II and MOEA/D, while options based control parameters are left on their jMetal defaults. For NSGA-II the tuned control parameters are the population size N, the crossover probability c_p and the mutation probability m_p. The selection, mutation and cross-over operators are fixed to binary tournament selection, simulated binary crossover (Deb and Agrawal, 1994) and polynomial mutation, respectively. The tuning initialization bounds for NSGA-II are $N \in [10, 200]$, $c_p \in [0, 1]$ and $m_p \in [0, 1]$. The tuning constraints for NSGA-II are

$$5 \leq N \leq 500 \tag{3.16}$$
$$0 \leq c_p \leq 1 \tag{3.17}$$
$$0 \leq m_p \leq 1. \tag{3.18}$$

Concerning MOEA/D, the control parameters tuned are the number of subproblems N_s, the neighborhood size fraction T_f, the DE crossover probability C_r and DE scaling factor F. The neighborhood size fraction controls the size of MOEA/D subproblem neighborhoods, with the neighborhood size being equal to N_s multiplied by T_f. MOEA/D's initialization bounds are $N_s \in [10, 200]$, $T_f \in [0, 1]$, $C_r \in [0, 1]$ and $F \in [0, 2]$. The tuning constraints of MOEA/D's are

$$5 \leq N_s \leq 500 \tag{3.19}$$
$$0 \leq T_f \leq 1 \tag{3.20}$$
$$2 \leq N_s\, T_f \tag{3.21}$$
$$0 \leq C_r \leq 1 \tag{3.22}$$
$$0 \leq F \leq 2. \tag{3.23}$$

For pragmatic reasons both NSGA-II's N and MOEA/D's N_s are restricted to a maximum of 500, since the computational overhead of NSGA-II and MOEA/D increase proportionately with N and N_s, respectively.

The computational budget allocated to the tuning problems corresponds to the upper limit of what is deemed to be the standard use-case scenario. This upper limit is chosen as the computational work produced by a high-end desktop or laptop computer left to tune over-night. Specifically, 12 hours fully utilizing a 4^{th} Generation Intel® Core™ i7-4700MQ Processor. For the tuning problems described, the WFG generalist tuning problems are the most computationally expensive. Given this limiting factor, a computational budget is assigned to each bi-objective decomposition which is equivalent to assessing 1000 CPV tuples up to the maximum OFE budget of 10 000 without resampling, giving a γ of 10^7. Since evaluating a CPV tuple on a generalist bi-objective decomposition entails assessing the CPV tuple on all problems in the test suite being tuned to, the specified γ budgets effectively allows for n_u times less CPV tuple evaluations compared to the specialist decompositions.

3.3.3 Tuning Algorithms Compared

The tuning algorithms compared in the numerical experiments are MOTA, tMOPSO, and a base-line heuristic RAND^M. tMOPSO is compared against MOTA since it has been shown to be effective at tuning optimization algorithms under multiple OFE budgets (Dymond et al., 2014). RAND^M, which is a basic random search heuristic equipped with MOTA's CPV assessment procedure, is added to the numerical experiments to gauge the benefits of MOTA's DE-based process for generating candidate decision vectors. RAND^M generates a candidate decision vector for the i'th subproblem, $\mathbf{x}_i^c$ using a uniform random distribution as follows

$$x_{i,k}^c = \begin{cases} \ln \beta_t & \text{if } k = 1 \\ x_{i,k}^{b_t} + \big(r() - 0.5\big)\big(I_{u,k} - I_{l,k}\big) & \forall\, k \in 2, n_x, \end{cases} \tag{3.24}$$

where β_t is the target improvement OFE budget selected randomly from B with each OFE budget having the same likelihood of being selected, $\mathbf{x}^{b_t}$ is the decision vector from the i'th subproblem's PFA which performs the best at β_t, $\mathbf{I}_u$ and $\mathbf{I}_l$ are the tuning problem's initialization bounds, and $r()$ is random function returning a value between 0 and 1 using a uniform distribution. M-FETA (Smit et al., 2010) is not compared against MOTA on account of the M-FETA algorithm not being designed to tune under multiple OFE budgets.

In order to apply the bi-objective tMOPSO tuning algorithm to the many objective tuning problems, each tuning problem is broken up into uncoupled bi-objective subproblems. The tMOPSO runs to determine the generalist CPV tuples happen last, since those runs require information on the Nadir and Ideal point which are taken from the tMOPSO runs targeted on specialist CPVs. Regarding the generalist tuning problems, MOTA and RAND^M have an advantage over tMOPSO, in that these algorithms share information between the solution approximations of the different subproblems. tMOPSO as a bi-objective tuning algorithm has no mechanics for sharing this many objective information. MOTA and RAND^M use the same neighborhood topology for the tuning problems. In particular, the specialist articulations, and the overall generalist articulations have a zero-sized update neighborhood, while each of the leave-one-out generalist articulations have an update neighborhood of size 3, consisting of two other leave-one-out generalist articulations as well as the overall generalist articulation. For all cases, MOTA and RAND^M use a candidate generation neighborhood consisting of all of the subproblems.

The same re-sampling interruption procedure is used by the compared tuning algorithms. This commonality should ensure that any performance difference observed is not influenced by the use of difference noise handling procedures. For MOTA, tMOPSO, and RAND^M a total sampling size of 25 is used, with resampling interruption checks occurring after sampling increments of $1, 2, 3, 4, 5$ and 10 using MWUTs given an interruption confidence of 60%. Since the IGD calculations are of moderate computational cost, resampling interruption checks are conducted for each OFE budget constraint under which a CPV tuple is being assessed. A common constraint handling approach is also used by the compared tuning algorithms. Candidate process generation repeats until a valid decision vector is found, subject to a threshold of 10 attempts. After 10 attempts, random values are generated inside the initialization bounds until

a valid candidate is generated. Finally all algorithms use an OFE budget overshoot function of $\beta^+ = 1.6\beta_a + 100$.

The comparison of MOTA and tMOPSO is complicated by the fact that the performance of these algorithms is sensitive to their CPVs. These algorithms are therefore first tuned before being compared, as to ensure MOTA, and tMOPSO use CPVs suitable for the problems on which they are going to be compared. In contrast, the base-line RAND^M does not require pre-comparison tuning on account of not having any control parameters. The pre-comparison tuning of MOTA and tMOPSO is done using the tuning particle swarm optimization (tPSO) algorithm (Dymond et al., 2014)). tPSO is set up to search for CPVs well suited for the NSGA-II ZDT specialist problem, and for the NSGA-II DTLZ specialist problem. The results from these tPSO runs are to be compared against each other as to check consistency. MOTA and tMOPSO are not tuned to the NSGA-II WFG specialist tuning problem since the WFG tuning problems are computationally expensive in this numerical setup, being more then 10 times more expensive than their ZDT and DTLZ counterparts. tPSO tunes each respective algorithm according to the utility measure u_{tPSO}. u_{tPSO} is a weighted sum performance aggregation over all of the respective subproblems for the tuning problem. For the NSGA-II DTLZ specialist problem which has 7 subproblems or preference articulations, tPSO minimizes

$$u_{tPSO} = -\sum_{i=1}^{7} \tau_i, \tag{3.25}$$

where τ_i is the bi-objective HV of the i'th IGD objective and the β_a or speed objective, calculated on the PF normalized between a commonly used Nadir and Ideal point. The same setup is used for tuning to the NSGA-II ZDT specialist problem. To account for stochastic effects, resampling over 20 independent runs is conducted. Based on Dymond et al. (2014), the tPSO settings are a swarm size of 10 and an inertia factor of 0.5. The tPSO tuning budget is 400 CPV tuple evaluations, which is equivalent to assessing 20 CPV tuples using a resampling size of 20. tPSO performs pre-emptive resampling termination checks after every resampling iteration, using MWUTs and an interruption confidence of 66%. Regarding handling tuning constraints, if an invalid CPV tuple is generated, tPSO continues to regenerate that tuple until all constraints are satisfied.

The MOTA CPVs which are tuned by tPSO are the DE scaling factor F, the DE crossover rate C_r, and the OFE perturbation factor β_δ. tPSO initialization bounds are $F \in [0, 4]$, $C_r \in [0, 1]$ and $\beta_\delta \in [0.1, 0.5]$ The tPSO tuning constraints for MOTA are $F > 0$, $C_r \in [0, 1]$ and $\beta_\delta \in [0, 1]$. For tMOPSO, the swarm size N, the inertia factor ω, the personal acceleration constant c_p, the global acceleration constant c_g and tMOPSO's OFE perturbation factor c_β are tuned by tPSO. The initialization bounds for tMOPSO CPVs are $N \in [2, 10]$, $\omega \in [0, 1]$, $c_p \in [0, 3]$, $c_g \in [0, 3]$ and $c_\beta \in [0.1, 0.5]$. After initialization tPSO search constraints are $N \geq 2$, $\omega \in [0, 1]$, $c_p \geq 0$, $c_g \geq 0$ and $c_\beta \in [0, 1]$.

After the tPSO tuning is completed and effective CPVs for MOTA and tMOPSO are determined, MOTA, tMOPSO and RAND^M are applied to the 12 ZDT, DTLZ and WFG tuning problems. To account for stochastic effects, comparison is conducted using a sample of 20 independent runs per tuning problem.

3.4 Numerical Results

The results from the numerical experiments are presented in three parts. First the results are given from the tPSO tuning as to determine effective CPVs for MOTA and tMOPSO. Thereafter, the numerical results from the specialist tuning problems are presented, followed by those of the generalist tuning problems.

When analyzing the results from the tuning problems, the objective function values of respective PFAs are normalized using a common objective normalization function. Use of a common objective normalization function is required, since there is expected to some variance in the Ideal and Nadir points approximations made by MOTA, tMOPSO, and RAND^M during the course of the tuning optimization. Therefore the objective value results are renormalized between a common Nadir and Ideal point, for purposes of fair comparison. The comparison Ideal and Nadir points were calculated after MOTA, tMOPSO, and RAND^M were applied to the specialist tuning problems, by combining the tuning results. Specifically, a PFA was constructed for each tuning problem by combining all of the results for that problem. After which, the Ideal and Nadir points used to compare the tuning algorithms, were taken from these constructed PFAs. This approach could not however be followed for the tPSO tuning of MOTA and tMOPSO since tPSO needs to compare performances during the course of the tuning run, and as such cannot postpone the calculation of the normalization Ideal and Nadir points. Therefore, tMOPSO was applied to the relevant specialist tuning problem to determine normalization Ideal and Nadir points for use in the tPSO tuning runs. In particular, the results from ten independent runs of tMOPSO using the CPV settings from Dymond et al. (2014), were combined into one PFA as to determine the tPSO normalization Ideal and Nadir points, the results of which are shown in Table 3.1.

3.4.1 Selecting the CPVs for the Compared Tuning Algorithms

Three independent runs were conducted for each tPSO tuning of MOTA and tMOPSO respectively, as to check consistency among the tPSO results. Furthermore, the CPV recommendations from the three runs from tuning to the NSGA-II ZDT specialist problem, are compared to those of the three runs from tuning to the NSGA-II DTLZ specialist problem. The aim of these consistency checks is to ensure that tPSO does not return an outlier CPV tuple, which is unlikely to be reproduced by an independent third party conducting the same experiments.

Table 3.2 shows the CPV tuples which tPSO found to be effective for MOTA on the NSGA-II ZDT and DTLZ specialist problems. An acceptable level of consistency is observed among the tPSO CPV recommendations for MOTA, with similar values for F, C_r, β_δ being returned. Regarding tMOPSO, Table 3.3 shows tPSO's CPV recommendations. An acceptable level of consistency in terms of exploitation versus exploration is observed again, if the combined effects of ω, $c_p + c_g$ (Clerc and Kennedy, 2002), and the tMOPSO OFE perturbations c_β factor are considered. For example, the tPSO recommendation which has a far higher ω than the other tPSO ω recommendations, is accompanied by a lower $c_p + c_g$ value relative to the other tPSO runs. As expected, different CPV recommendations were made for the NSGA-II ZDT specialist problem compared to that of the NSGA-II DTLZ specialist problem.

Table 3.1: The Nadir and Ideal points of the respective NSGA-II tuning problems, which were used by tPSO to compare performances. For these problems NSGA-II is tuned to enhance performance according to the IGD performance metric under multiple OFE budgets. The IGD values tabulated are multiplied by 10^3 for readability.

	ZDT_1	ZDT_2	ZDT_3	ZDT_4	ZDT_6
max.	78.830	121.652	85.624	3197.750	279.499
min.	0.387	0.422	0.473	1.387	0.661

	$DTLZ_1$	$DTLZ_2$	$DTLZ_3$	$DTLZ_4$	$DTLZ_5$	$DTLZ_6$	$DTLZ_7$
max.	18222.560	47.663	52108.309	13.593	46.092	677.292	182.925
min.	17.100	0.243	460.708	0.108	0.234	7.668	0.336

Table 3.2: CPVs recommended by the tPSO pre-tuning of MOTA

specialist problem	run	F	Cr	β_δ
NSGA-II ZDT	best	2.40	0.55	0.37
	second best	2.55	0.83	0.44
	worst	3.02	0.80	0.40
NSGA-II DTLZ	best	1.94	0.73	0.17
	second best	1.99	0.85	0.24
	worst	1.54	0.79	0.18

Table 3.3: CPVs recommended by the tPSO pre-tuning of tMOPSO

specialist problem	run	N	ω	c_p	c_g	c_β
NSGA-II ZDT	best	4	0.44	0.73	2.76	0.02
	second best	7	0.31	1.62	2.10	0.23
	worst	6	0.30	1.82	1.83	0.29
NSGA-II DTLZ	best	5	0.28	1.35	2.34	0.02
	second best	7	0.73	0.89	1.59	0.15
	worst	6	0.26	3.10	1.31	0.30

Given that no outliers were observed, the CPVs used in the remainder of the experiments for MOTA and tMOPSO, are taken from the best tPSO run for tuning to the NSGA-II DTLZ specialist problem. The tPSO results from the NSGA-II DTLZ specialist problem are chosen in place of the results from NSGA-II ZDT specialist problem, since the DTLZ specialist problem is considered more representative in terms of number of tuning objectives. Specifically, the DTLZ problem has 8 tuning objectives if the speed objective is included, while the ZDT problem has 6 tuning objectives and the WFG problem has 10 tuning objectives.

3.4.2 Specialist Tuning Results

MOTA, tMOPSO and RAND^M were applied to the specialist and generalist tuning problems as to generate 20 independent runs for each problem. Analysis of the results from these tuning problems is conducted according to the τ performance metric, where τ measures the HV achieved for a given bi-objective decomposition in the normalized objective space as outlined in (3.25). The normalization Ideal and Nadir points used in the analysis of the results for the NSGA-II tuning problems are shown in Table 3.4, and in Table 3.5 for the MOEA/D tuning problems.

Comparison according to performance on the NSGA-II specialist tuning problems is conducted quantitatively according to MWUTs as summarized in Table 3.6, and qualitatively according to box plots as shown in Figure 3.4. Two sample means are considered statistically similar if a MWUT indicates that the difference in these two sample means is not statistically significant given a confidence level of 90%. On the NSGA-II ZDT specialist problem, tMOPSO outperforms MOTA on 2/5 subproblems, is outperformed by MOTA on 2/5 subproblems, and performs statistically similar on the remaining subproblem. For the NSGA-II DTLZ specialist problem, tMOPSO beats MOTA on 2/7 subproblems, is outperformed by MOTA on one subproblem, and is statistically similar on the other 4/7 subproblems. Regarding the NSGA-II WFG specialist problem, tMOPSO outperformed MOTA on 2/9 subproblems, it outperformed by MOTA on 2/9 subproblems, while being statistically similar to MOTA on the remaining 5/9 subproblems. RAND^M is outperformed by tMOPSO and MOTA on all the subproblems for the NSGA-II specialist tuning problems, with exception to the DTLZ$_4$ subproblem where the difference in sample means is not statistically significant. The sample mean comparisons are in agreement with a box plot comparisons of the algorithms performance, shown in Figure 3.4.

For the MOEA/D specialist tuning problems, comparison is conducted in the same manner, with Table 3.7 summarizing the MWUT comparisons and Figure 3.5 showing box plots of the sample distributions. On the MOEA/D ZDT specialist problem, MOTA and tMOPSO produce similar performances on 4/5 subproblems, and MOTA outperformed tMOPSO on the other ZDT subproblem. For the MOEA/D DTLZ specialist problem, tMOPSO outperformed MOTA on 5/7 subproblems, while MOTA performed better on the remaining 2/7 subproblems. Regarding the MOEA/D WFG specialist problem, tMOPSO outperformed MOTA on 3/9 subproblems, while for the other 6/9 subproblems MOTA and tMOPSO offer statistically similar performances. RAND^M is outperformed by MOTA and tMOPSO on all of the subproblems for all of the specialist tuning problems, with exception of the subproblem focused on determing CPVs for WFG$_9$, for which statistically similar performance was recorded.

Taking the specialist results into account as a whole, it is concluded that MOTA and

Table 3.4: The Nadir and Ideal points used to compare the NSGA-II based tuning problems. For these problems NSGA-II is tuned to enhance performance according to the IGD performance metric under multiple OFE budgets. The IGD values tabulated are multiplied by 10^3 for readability.

	ZDT_1	ZDT_2	ZDT_3	ZDT_4	ZDT_6
max.	78.746	121.304	85.624	3140.462	275.842
min.	0.364	0.402	0.471	1.204	0.620

	$DTLZ_1$	$DTLZ_2$	$DTLZ_3$	$DTLZ_4$	$DTLZ_5$	$DTLZ_6$	$DTLZ_7$
max.	17557.518	45.638	51911.085	13.593	45.951	673.232	174.187
min.	12.777	0.232	363.496	0.086	0.231	6.189	0.335

	WFG_1	WFG_2	WFG_3	WFG_4	WFG_5	WFG_6	WFG_7	WFG_8	WFG_9
max.	12.397	4.845	1.655	3.474	5.766	6.889	4.640	3.620	6.516
min.	3.483	0.171	0.714	0.092	0.605	0.679	0.081	0.745	0.180

Table 3.5: The Nadir and Ideal points used to compare the MOEA/D based tuning problems. For these problems MOEA/D is tuned to enhance performance according to the IGD performance metric under multiple OFE budgets. The IGD values tabulated are multiplied by 10^3 for readability.

	ZDT_1	ZDT_2	ZDT_3	ZDT_4	ZDT_6
max.	81.068	123.746	88.729	3236.527	280.509
min.	0.129	0.112	0.402	0.797	0.056

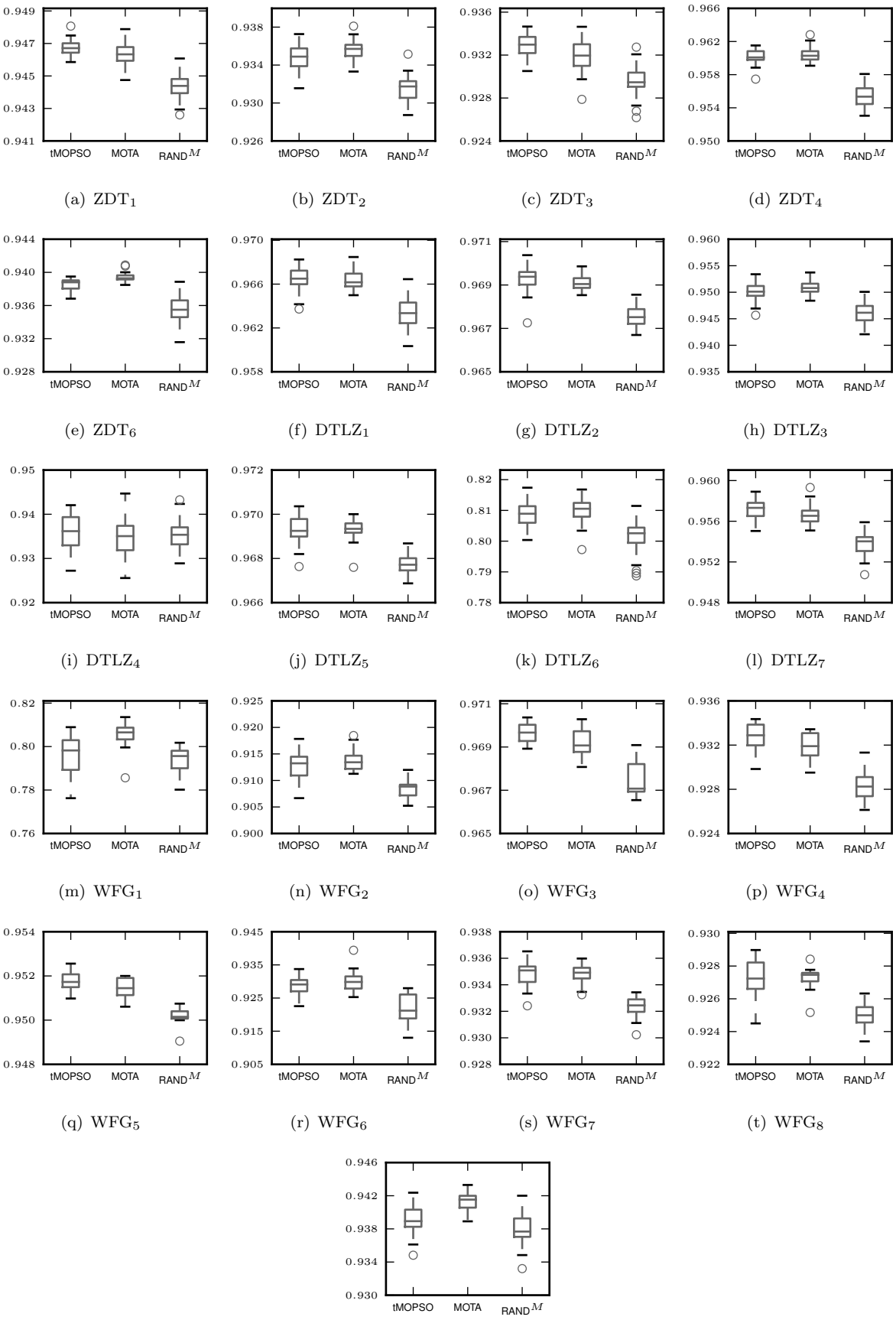
	$DTLZ_1$	$DTLZ_2$	$DTLZ_3$	$DTLZ_4$	$DTLZ_5$	$DTLZ_6$	$DTLZ_7$
max.	18895.532	47.975	54699.749	13.922	47.198	680.422	182.987
min.	2.101	0.174	228.864	0.063	0.177	0.120	0.194

	WFG_1	WFG_2	WFG_3	WFG_4	WFG_5	WFG_6	WFG_7	WFG_8	WFG_9
max.	12.434	5.066	1.699	3.740	6.059	7.189	4.864	3.743	6.863
min.	1.820	0.285	0.707	0.172	0.557	0.251	0.118	0.714	0.209

Table 3.6: Performances on the NSGA-II specialist tuning problems.

		$\tau(\times 10^3)$		
suite	problem	tMOPSO	MOTA	RAND^M
ZDT	1	**946.743**	946.265	944.384
	2	934.845	**935.562**	931.651
	3	**932.770**	931.857	929.537
	4	*960.120*	**960.496**	955.470
	6	938.548	**939.447**	935.492
DTLZ	1	**966.481**	*966.385*	963.242
	2	**969.295**	969.101	967.557
	3	950.034	**950.824**	946.185
	4	**936.064**	*934.882*	*935.222*
	5	*969.339*	**969.345**	967.734
	6	*808.841*	**810.097**	801.350
	7	**957.173**	956.581	953.861
WFG	1	795.679	**805.681**	793.968
	2	*912.616*	**913.655**	908.411
	3	**969.683**	969.162	967.491
	4	**932.710**	931.937	928.284
	5	**951.742**	*951.456*	950.191
	6	*928.772*	**930.064**	921.503
	7	**934.793**	*934.787*	932.267
	8	**927.275**	*927.262*	924.992
	9	939.018	**941.294**	938.045

† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 90% confidence level.



(a) ZDT_1 (b) ZDT_2 (c) ZDT_3 (d) ZDT_4

(e) ZDT_6 (f) $DTLZ_1$ (g) $DTLZ_2$ (h) $DTLZ_3$

(i) $DTLZ_4$ (j) $DTLZ_5$ (k) $DTLZ_6$ (l) $DTLZ_7$

(m) WFG_1 (n) WFG_2 (o) WFG_3 (p) WFG_4

(q) WFG_5 (r) WFG_6 (s) WFG_7 (t) WFG_8

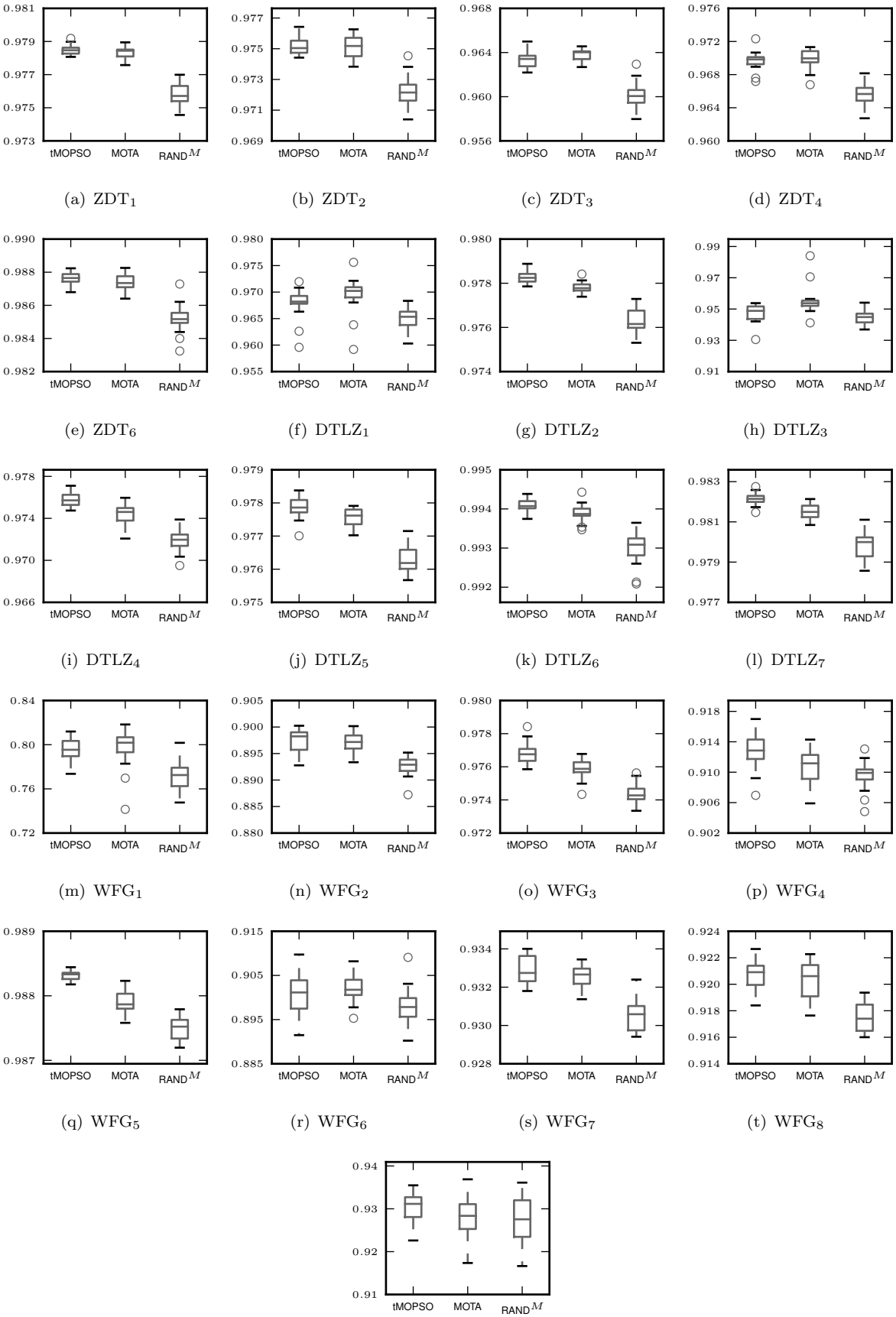
(u) WFG_9

Figure 3.4: τ distributions for the NSGA-II specialist problems

Table 3.7: Performances on the MOEA/D specialist tuning problems.

		$\tau(\times 10^3)$		
suite	problem	tMOPSO	MOTA	RAND^M
ZDT	1	**978.496**	*978.324*	975.777
	2	*975.150*	**975.154**	972.183
	3	963.370	**963.768**	960.094
	4	*969.652*	**969.875**	965.638
	6	**987.639**	*987.392*	985.198
DTLZ	1	967.943	**969.541**	965.065
	2	**978.276**	977.818	976.321
	3	947.300	**955.056**	944.539
	4	**975.730**	974.372	971.954
	5	**977.851**	977.553	976.276
	6	**994.090**	993.891	993.032
	7	**982.132**	981.473	979.793
WFG	1	*795.654*	**796.439**	773.524
	2	**897.245**	*897.224*	892.643
	3	**976.787**	975.889	974.403
	4	**912.752**	910.685	909.565
	5	**988.313**	987.905	987.498
	6	*900.856*	**902.272**	898.140
	7	**932.921**	*932.576*	930.612
	8	**920.745**	*920.254*	917.489
	9	**930.300**	*928.280*	*927.276*

† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 90% confidence level.



(a) ZDT_1 (b) ZDT_2 (c) ZDT_3 (d) ZDT_4

(e) ZDT_6 (f) $DTLZ_1$ (g) $DTLZ_2$ (h) $DTLZ_3$

(i) $DTLZ_4$ (j) $DTLZ_5$ (k) $DTLZ_6$ (l) $DTLZ_7$

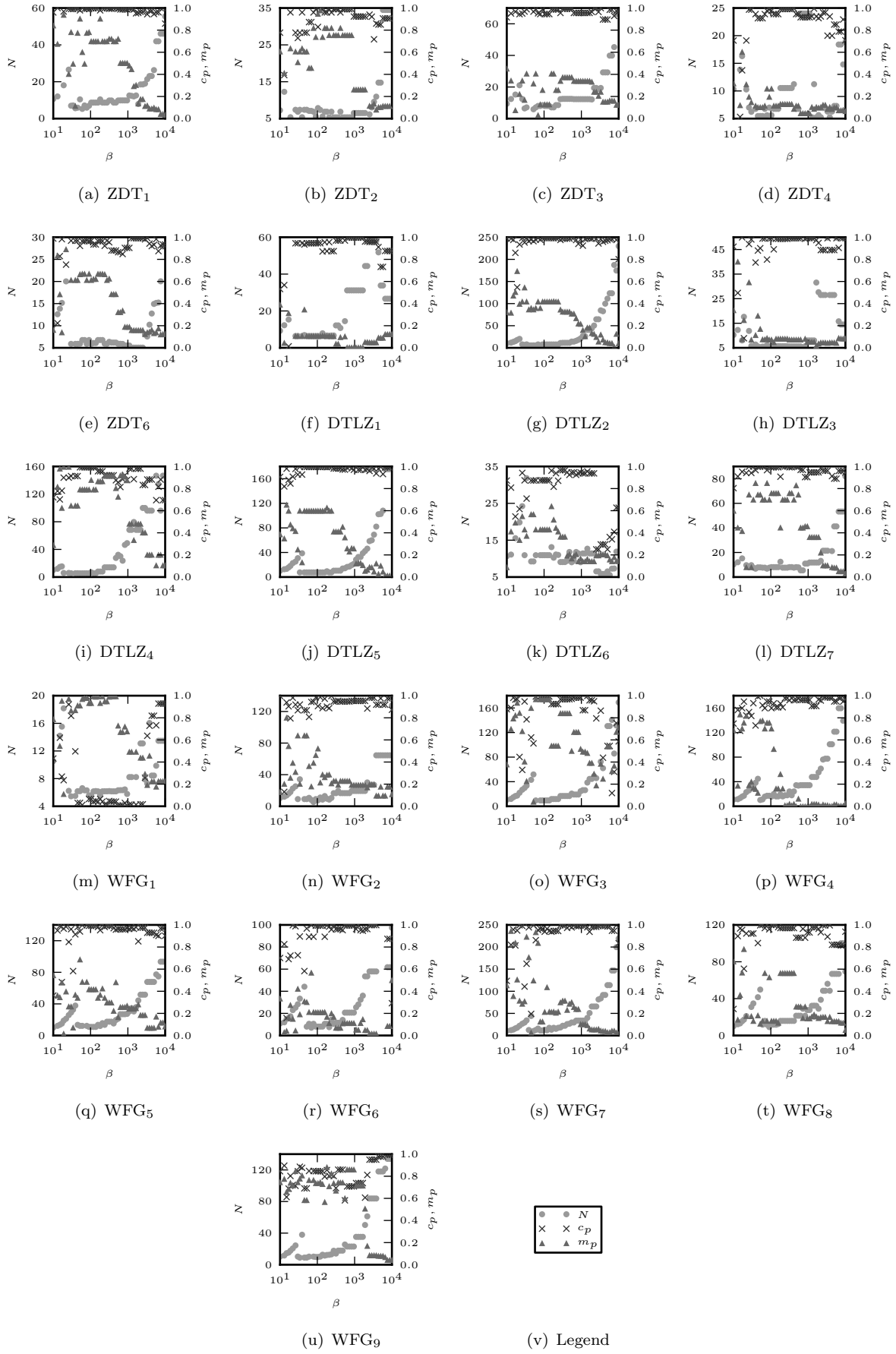
(m) WFG_1 (n) WFG_2 (o) WFG_3 (p) WFG_4

(q) WFG_5 (r) WFG_6 (s) WFG_7 (t) WFG_8

(u) WFG_9

Figure 3.5: τ distributions for the MOEA/D specialist problems

tMOPSO offer similar performance, while both algorithms outperform the base-line RAND^M. It was postulated in the introduction that MOTA's ability to share information among sub-problems may be of benefit even for the case when an algorithm is to be tuned under multiple OFE budgets to each instance of a problem suite on an individual basis only. In particular, information sharing could aid tuning by exploiting common trends among the tuning solutions to these problem instances. For these experiments, MOTA's information sharing strategy via candidate generation neighborhoods and DE operators, is not able to outperform tMOPSO, even though CPV trends are as shown in Figure 3.6 and in Figure 3.7. If these results are limited to MOTA mechanics, or are reflective of the validity of the idea of sharing information as to aid to determining specialist CPVs over multiple OFE budgets as a whole, is left as a question for future research.



(a) ZDT_1 (b) ZDT_2 (c) ZDT_3 (d) ZDT_4

(e) ZDT_6 (f) $DTLZ_1$ (g) $DTLZ_2$ (h) $DTLZ_3$

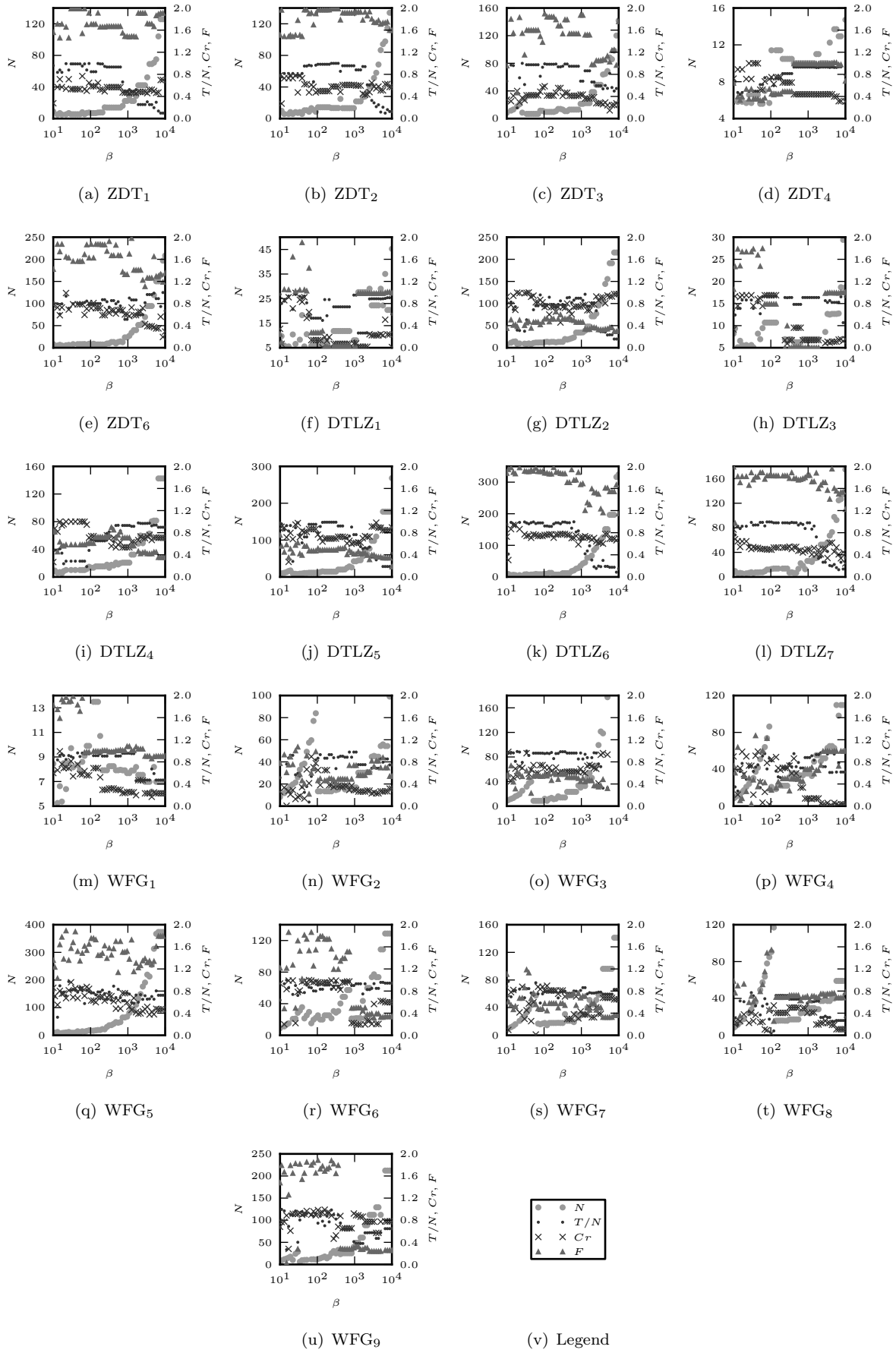
(i) $DTLZ_4$ (j) $DTLZ_5$ (k) $DTLZ_6$ (l) $DTLZ_7$

(m) WFG_1 (n) WFG_2 (o) WFG_3 (p) WFG_4

(q) WFG_5 (r) WFG_6 (s) WFG_7 (t) WFG_8

(u) WFG_9 (v) Legend

Figure 3.6: The best tMOPSO results for the NSGA-II specialist tuning problems. The recommended CPVs are shown for differing OFE budgets, β.



(a) ZDT_1 (b) ZDT_2 (c) ZDT_3 (d) ZDT_4

(e) ZDT_6 (f) $DTLZ_1$ (g) $DTLZ_2$ (h) $DTLZ_3$

(i) $DTLZ_4$ (j) $DTLZ_5$ (k) $DTLZ_6$ (l) $DTLZ_7$

(m) WFG_1 (n) WFG_2 (o) WFG_3 (p) WFG_4

(q) WFG_5 (r) WFG_6 (s) WFG_7 (t) WFG_8

(u) WFG_9 (v) Legend

Figure 3.7: The best tMOPSO results for the MOEA/D specialist tuning problems. The recommended CPVs are shown for differing OFE budgets, β.

3.4.3 Generalist Tuning Results

On the generalist tuning problems, MOTA's CPV tuple assessment procedure which utilizes update neighborhoods had a significant effect. On the NSGA-II generalist problems MOTA outperformed tMOPSO on all of the ZDT, DTLZ and WFG subproblems as shown in Table 3.8 and Figure 3.10. RAND^M is more competitive against MOTA on the NSGA-II generalist problems compared to the specialist problems. Specifically, on 6/24 subproblems the difference in sample τ means between RAND^M and MOTA is not statistically significant according to MWUTs given a 90% confidence level. MOTA outperformed RAND^M on the remaining 18/24 NSGA-II generalist subproblems. Regarding the MOEA/D generalist problems MOTA outperformed tMOPSO on all but 1/24 subproblems, as Table 3.9 and Figure 3.11 show. RAND^M outperformed MOTA on 1/24 subproblems, while performing worse than MOTA on the remaining 23/24 subproblems.

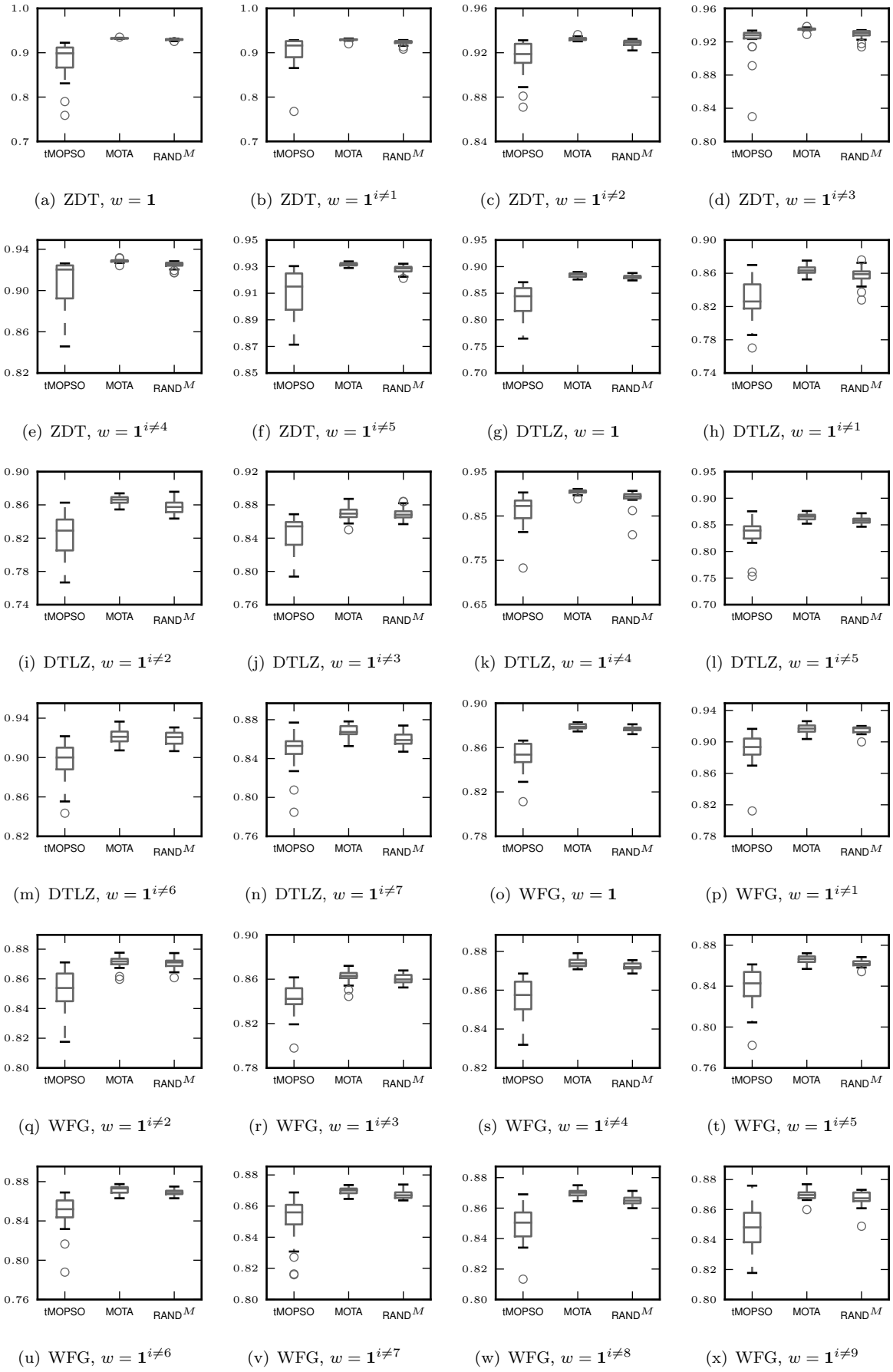
The superior performance of MOTA on the generalist tuning problem is expected given MOTA's design. Unlike tMOPSO, MOTA is designed to be a many objective tuning algorithm. Therefore MOTA is designed to share information between subproblems, whereas tMOPSO is a bi-objective optimization algorithm and therefore has no mechanics to propagate information among the different subproblems it solves. The outperformance of tMOPSO by RAND^M on the generalist tuning problems, compared to tMOPSO completely outperforming RAND^M on the specialist tuning problems, further emphasizes the importance of information sharing.

MOTA's results were scrutinized in regard to producing sensible CPV recommendations. Subproblems were added in the generalist tuning problem formulations for all the leave-one-out generalist combinations. Comparing the results from these subproblems against the $\mathbf{w} = [1, 1, \ldots, 1]$ subproblem, gives an indication of the validity of MOTA's results. Graphical comparison of the results of these leave-one-objective-out generalist and the full generalist is shown for the NSGA-II generalist problems in Figure 3.10, and for the MOEA/D generalist problems in Figure 3.11. For the NSGA-II problems, a high level of consistency among the generalist subproblems in terms of crossover and mutation probabilities for similar OFE budgets is observed, while the majority of population size CPV recommendations followed a similar trend. In particular, for very low OFE budgets the optimal population size is equal to the OFE budget. After which there is a drop in the optimal population size, at the OFE budget where the NSGA-II operators become more effective than random search. Thereafter, the optimal population size continues to increase as the available OFE budget increases. For the MOEA/D problems, an acceptable level of consistency is also observed, with similar DE scaling factors, crossover probability and neighborhood factions being recommended among similar OFE budgets, while similar population size trends as for the NSGA-II generalist problem were observed. Given the consistency with previous work in Dymond et al. (2013) MOTA's results are deemed acceptable.

Table 3.8: Performances on the NSGA-II generalist tuning problems. The $\mathbf{1}^{i \neq j}$ notation indicates a vector whose elements are all equal to 1, with exception to the j'th element which is equal to 0.

		$\tau \times 10^3$		
suite	**w**	tMOPSO	MOTA	RAND^M
ZDT	$\mathbf{1}$	882.125	**932.601**	929.575
	$\mathbf{1}^{i \neq 1}$	902.718	**929.089**	923.368
	$\mathbf{1}^{i \neq 2}$	913.863	**932.617**	928.565
	$\mathbf{1}^{i \neq 3}$	920.730	**935.314**	929.664
	$\mathbf{1}^{i \neq 4}$	907.071	**928.597**	925.018
	$\mathbf{1}^{i \neq 5}$	908.475	**931.625**	927.993
DTLZ	$\mathbf{1}$	834.185	**884.049**	880.356
	$\mathbf{1}^{i \neq 1}$	826.849	**864.007**	857.315
	$\mathbf{1}^{i \neq 2}$	823.796	**865.996**	857.257
	$\mathbf{1}^{i \neq 3}$	844.050	**869.802**	*869.114*
	$\mathbf{1}^{i \neq 4}$	858.144	**904.283**	891.732
	$\mathbf{1}^{i \neq 5}$	832.369	**864.767**	857.945
	$\mathbf{1}^{i \neq 6}$	894.699	**921.736**	*920.040*
	$\mathbf{1}^{i \neq 7}$	847.747	**868.590**	859.925
WFG	$\mathbf{1}$	851.983	**879.122**	876.701
	$\mathbf{1}^{i \neq 1}$	890.162	**916.305**	*915.295*
	$\mathbf{1}^{i \neq 2}$	852.251	**870.980**	*870.457*
	$\mathbf{1}^{i \neq 3}$	840.971	**861.910**	*860.170*
	$\mathbf{1}^{i \neq 4}$	856.030	**874.207**	872.268
	$\mathbf{1}^{i \neq 5}$	839.367	**866.008**	862.065
	$\mathbf{1}^{i \neq 6}$	848.161	**871.661**	868.776
	$\mathbf{1}^{i \neq 7}$	850.533	**869.538**	867.419
	$\mathbf{1}^{i \neq 8}$	848.610	**869.767**	865.051
	$\mathbf{1}^{i \neq 9}$	847.614	**869.408**	*866.949*

† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 90% confidence level.



(a) ZDT, $w = \mathbf{1}$ (b) ZDT, $w = \mathbf{1}^{i\neq1}$ (c) ZDT, $w = \mathbf{1}^{i\neq2}$ (d) ZDT, $w = \mathbf{1}^{i\neq3}$

(e) ZDT, $w = \mathbf{1}^{i\neq4}$ (f) ZDT, $w = \mathbf{1}^{i\neq5}$ (g) DTLZ, $w = \mathbf{1}$ (h) DTLZ, $w = \mathbf{1}^{i\neq1}$

(i) DTLZ, $w = \mathbf{1}^{i\neq2}$ (j) DTLZ, $w = \mathbf{1}^{i\neq3}$ (k) DTLZ, $w = \mathbf{1}^{i\neq4}$ (l) DTLZ, $w = \mathbf{1}^{i\neq5}$

(m) DTLZ, $w = \mathbf{1}^{i\neq6}$ (n) DTLZ, $w = \mathbf{1}^{i\neq7}$ (o) WFG, $w = \mathbf{1}$ (p) WFG, $w = \mathbf{1}^{i\neq1}$

(q) WFG, $w = \mathbf{1}^{i\neq2}$ (r) WFG, $w = \mathbf{1}^{i\neq3}$ (s) WFG, $w = \mathbf{1}^{i\neq4}$ (t) WFG, $w = \mathbf{1}^{i\neq5}$

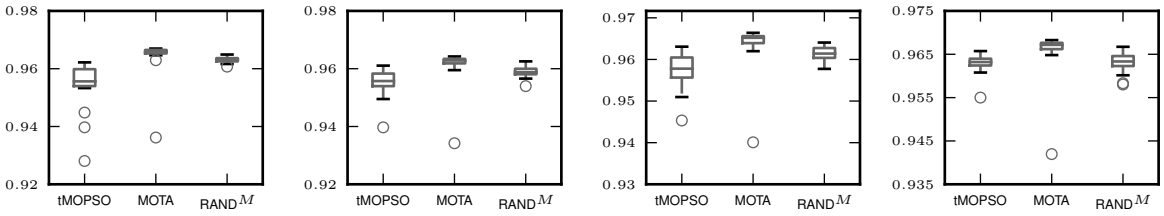
(u) WFG, $w = \mathbf{1}^{i\neq6}$ (v) WFG, $w = \mathbf{1}^{i\neq7}$ (w) WFG, $w = \mathbf{1}^{i\neq8}$ (x) WFG, $w = \mathbf{1}^{i\neq9}$

Figure 3.8: τ distributions for the NSGA-II generalist problems.

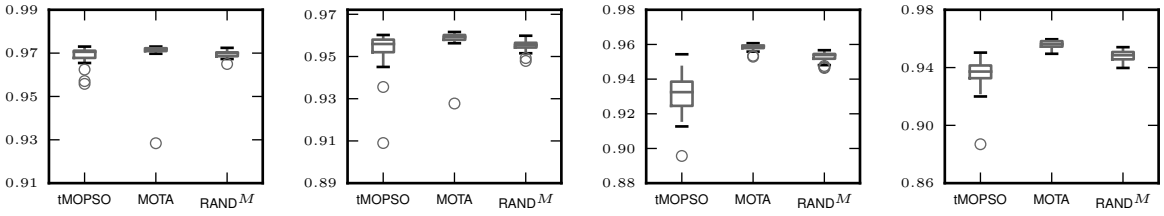
Table 3.9: Performances on the MOEA/D generalist tuning problems. The $\mathbf{1}^{i\neq j}$ notation indicates a vector whose elements are all equal to 1, with exception to the j'th element which is equal to 0.

		$\tau \times 10^3$		
suite	**w**	tMOPSO	MOTA	RAND^M
ZDT	$\mathbf{1}$	954.444	**964.325**	963.128
	$\mathbf{1}^{i\neq 1}$	955.393	**961.119**	958.943
	$\mathbf{1}^{i\neq 2}$	957.256	**963.529**	961.398
	$\mathbf{1}^{i\neq 3}$	962.893	**965.680**	963.138
	$\mathbf{1}^{i\neq 4}$	*968.530*	969.579	**969.613**
	$\mathbf{1}^{i\neq 5}$	951.967	**957.607**	955.255
DTLZ	$\mathbf{1}$	930.843	**958.360**	953.098
	$\mathbf{1}^{i\neq 1}$	934.621	**955.802**	948.353
	$\mathbf{1}^{i\neq 2}$	929.119	**952.000**	944.280
	$\mathbf{1}^{i\neq 3}$	939.228	**962.001**	956.258
	$\mathbf{1}^{i\neq 4}$	930.771	**953.190**	944.010
	$\mathbf{1}^{i\neq 5}$	927.883	**954.376**	944.924
	$\mathbf{1}^{i\neq 6}$	923.124	**950.179**	941.941
	$\mathbf{1}^{i\neq 7}$	929.391	**954.263**	944.859
WFG	$\mathbf{1}$	847.868	**882.793**	880.510
	$\mathbf{1}^{i\neq 1}$	881.356	**904.044**	898.567
	$\mathbf{1}^{i\neq 2}$	850.824	**880.573**	877.219
	$\mathbf{1}^{i\neq 3}$	831.903	**866.881**	864.155
	$\mathbf{1}^{i\neq 4}$	836.309	**878.954**	876.121
	$\mathbf{1}^{i\neq 5}$	836.376	**866.981**	863.358
	$\mathbf{1}^{i\neq 6}$	854.657	**879.328**	876.132
	$\mathbf{1}^{i\neq 7}$	845.944	**872.224**	868.259
	$\mathbf{1}^{i\neq 8}$	850.214	**874.633**	870.659
	$\mathbf{1}^{i\neq 9}$	849.236	**881.582**	878.529

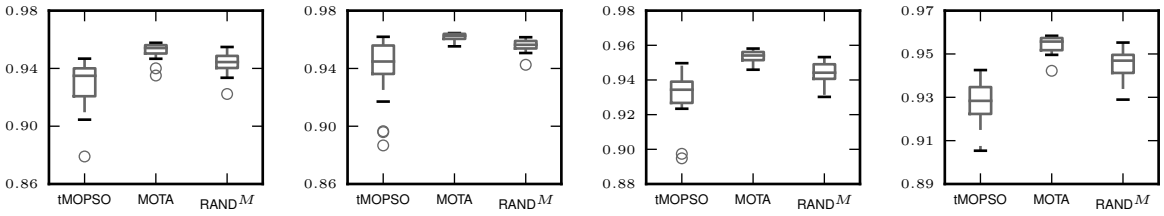
† *Italic entries* indicate samples whose difference in mean relative to the sample with the best mean is not statistically significant according to Mann-Whitney U-test with a 90% confidence level.



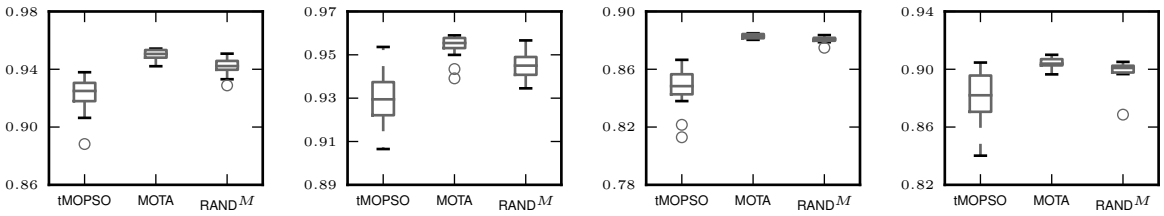
(a) ZDT, $w = \mathbf{1}$ (b) ZDT, $w = \mathbf{1}^{i\neq1}$ (c) ZDT, $w = \mathbf{1}^{i\neq2}$ (d) ZDT, $w = \mathbf{1}^{i\neq3}$



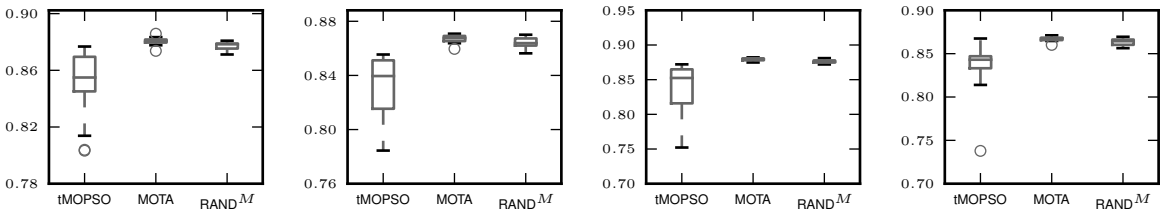
(e) ZDT, $w = \mathbf{1}^{i\neq4}$ (f) ZDT, $w = \mathbf{1}^{i\neq5}$ (g) DTLZ, $w = \mathbf{1}$ (h) DTLZ, $w = \mathbf{1}^{i\neq1}$



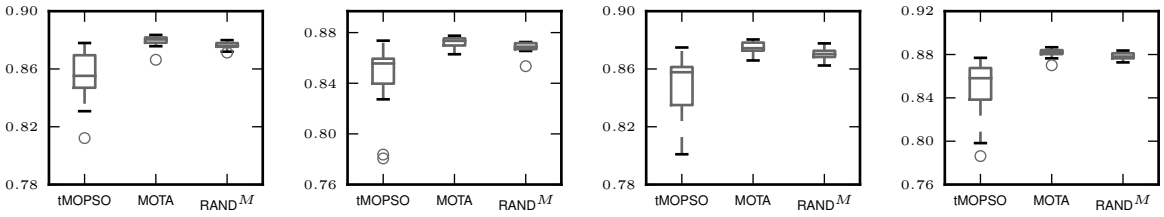
(i) DTLZ, $w = \mathbf{1}^{i\neq2}$ (j) DTLZ, $w = \mathbf{1}^{i\neq3}$ (k) DTLZ, $w = \mathbf{1}^{i\neq4}$ (l) DTLZ, $w = \mathbf{1}^{i\neq5}$



(m) DTLZ, $w = \mathbf{1}^{i\neq6}$ (n) DTLZ, $w = \mathbf{1}^{i\neq7}$ (o) WFG, $w = \mathbf{1}$ (p) WFG, $w = \mathbf{1}^{i\neq1}$

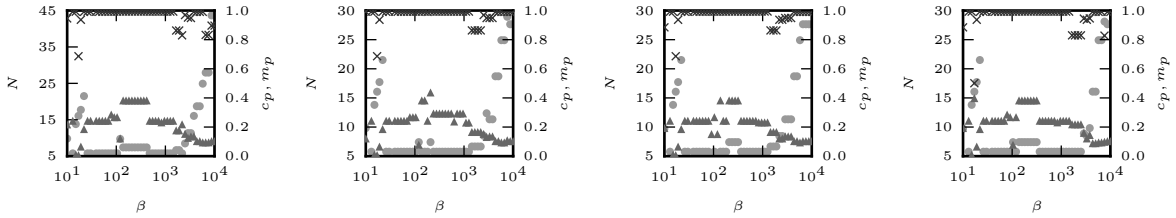


(q) WFG, $w = \mathbf{1}^{i\neq2}$ (r) WFG, $w = \mathbf{1}^{i\neq3}$ (s) WFG, $w = \mathbf{1}^{i\neq4}$ (t) WFG, $w = \mathbf{1}^{i\neq5}$

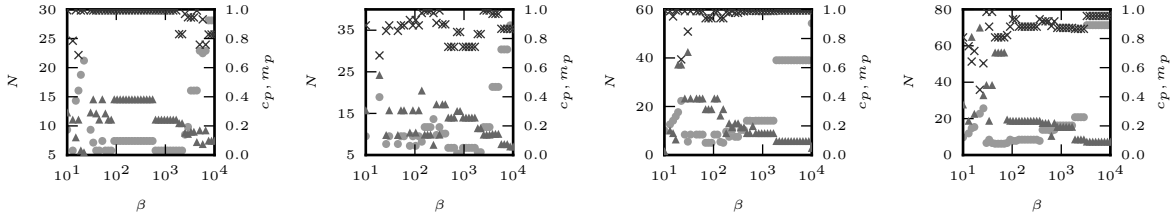


(u) WFG, $w = \mathbf{1}^{i\neq6}$ (v) WFG, $w = \mathbf{1}^{i\neq7}$ (w) WFG, $w = \mathbf{1}^{i\neq8}$ (x) WFG, $w = \mathbf{1}^{i\neq9}$

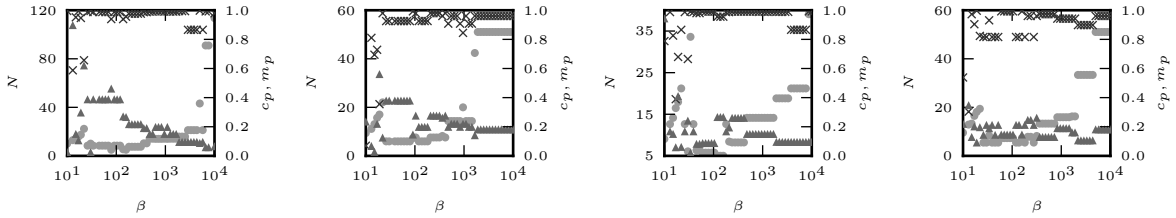
Figure 3.9: τ distributions for the MOEAD generalist problems.



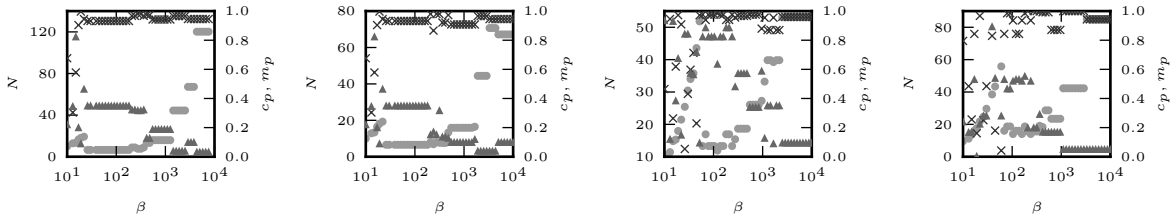
(a) ZDT, $w = \mathbf{1}$ (b) ZDT, $w = \mathbf{1}^{i\neq 1}$ (c) ZDT, $w = \mathbf{1}^{i\neq 2}$ (d) ZDT, $w = \mathbf{1}^{i\neq 3}$



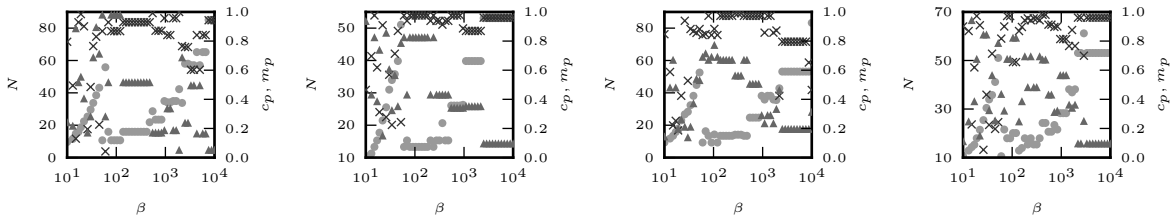
(e) ZDT, $w = \mathbf{1}^{i\neq 4}$ (f) ZDT, $w = \mathbf{1}^{i\neq 5}$ (g) DTLZ, $w = \mathbf{1}$ (h) DTLZ, $w = \mathbf{1}^{i\neq 1}$



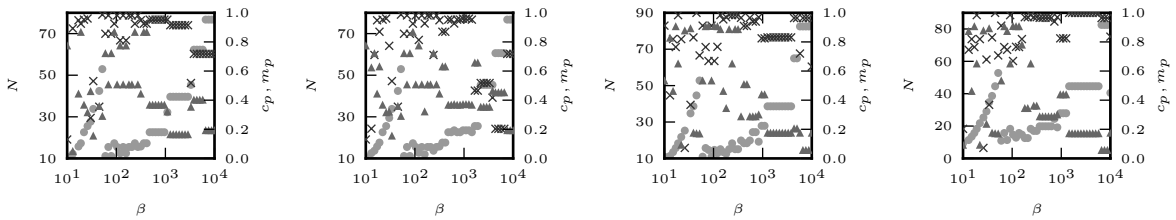
(i) DTLZ, $w = \mathbf{1}^{i\neq 2}$ (j) DTLZ, $w = \mathbf{1}^{i\neq 3}$ (k) DTLZ, $w = \mathbf{1}^{i\neq 4}$ (l) DTLZ, $w = \mathbf{1}^{i\neq 5}$



(m) DTLZ, $w = \mathbf{1}^{i\neq 6}$ (n) DTLZ, $w = \mathbf{1}^{i\neq 7}$ (o) WFG, $w = \mathbf{1}$ (p) WFG, $w = \mathbf{1}^{i\neq 1}$

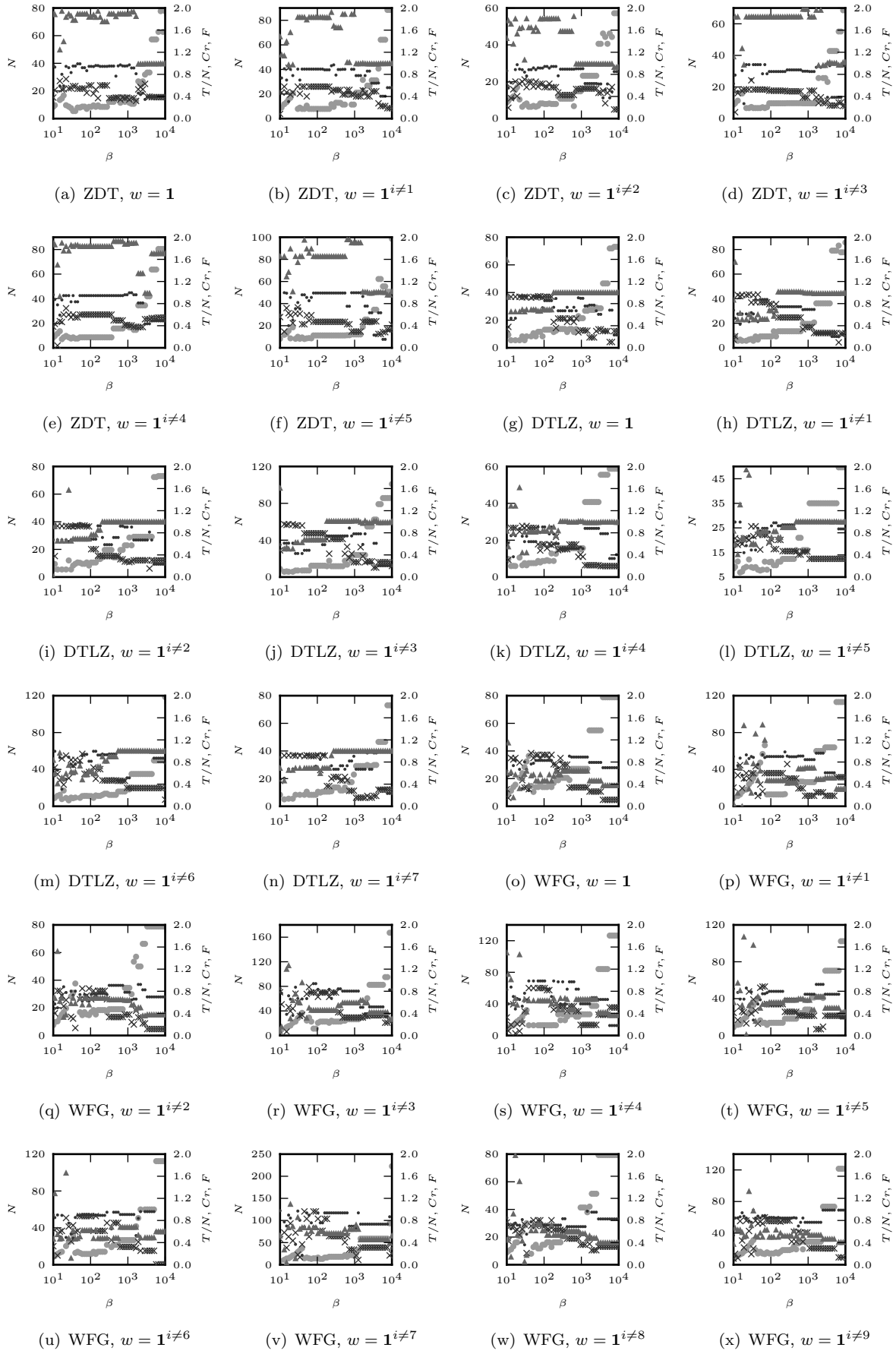


(q) WFG, $w = \mathbf{1}^{i\neq 2}$ (r) WFG, $w = \mathbf{1}^{i\neq 3}$ (s) WFG, $w = \mathbf{1}^{i\neq 4}$ (t) WFG, $w = \mathbf{1}^{i\neq 5}$



(u) WFG, $w = \mathbf{1}^{i\neq 6}$ (v) WFG, $w = \mathbf{1}^{i\neq 7}$ (w) WFG, $w = \mathbf{1}^{i\neq 8}$ (x) WFG, $w = \mathbf{1}^{i\neq 9}$

Figure 3.10: The best MOTA results for the NSGA-II generalist tuning problems. The recommended CPVs are shown for differing OFE budgets, β. The legend for the above subfigures is the same as in Figure 3.6.



(a) ZDT, $w = \mathbf{1}$ (b) ZDT, $w = \mathbf{1}^{i\neq1}$ (c) ZDT, $w = \mathbf{1}^{i\neq2}$ (d) ZDT, $w = \mathbf{1}^{i\neq3}$

(e) ZDT, $w = \mathbf{1}^{i\neq4}$ (f) ZDT, $w = \mathbf{1}^{i\neq5}$ (g) DTLZ, $w = \mathbf{1}$ (h) DTLZ, $w = \mathbf{1}^{i\neq1}$

(i) DTLZ, $w = \mathbf{1}^{i\neq2}$ (j) DTLZ, $w = \mathbf{1}^{i\neq3}$ (k) DTLZ, $w = \mathbf{1}^{i\neq4}$ (l) DTLZ, $w = \mathbf{1}^{i\neq5}$

(m) DTLZ, $w = \mathbf{1}^{i\neq6}$ (n) DTLZ, $w = \mathbf{1}^{i\neq7}$ (o) WFG, $w = \mathbf{1}$ (p) WFG, $w = \mathbf{1}^{i\neq1}$

(q) WFG, $w = \mathbf{1}^{i\neq2}$ (r) WFG, $w = \mathbf{1}^{i\neq3}$ (s) WFG, $w = \mathbf{1}^{i\neq4}$ (t) WFG, $w = \mathbf{1}^{i\neq5}$

(u) WFG, $w = \mathbf{1}^{i\neq6}$ (v) WFG, $w = \mathbf{1}^{i\neq7}$ (w) WFG, $w = \mathbf{1}^{i\neq8}$ (x) WFG, $w = \mathbf{1}^{i\neq9}$

Figure 3.11: The best MOTA results for the MOEA/D generalist tuning problems. The recommended CPVs are shown for differing OFE budgets, β. The legend for the above subfigures is the same as in Figure 3.7.

Before concluding this chapter, a disclaimer is given pertaining to the CPV tuples found to be optimal in these numerical experiments. NSGA-II and MOEA/D practitioners are reminded that these CPV tuples are only guaranteed of producing favorable results for optimization problems similar to those used in these experiments, i.e. the ZDT, DTLZ and WFG problems. Therefore if an optimization problem is tackled which is different from these problems, practitioners are advised to use MOTA or another tuning algorithm, as to determine CPV tuples which are effective on testing problems more representative of the problem being tackled.

With tMOPSO and MOTA presented, the next chapter demonstrates how these CPV study tools can be used in optimization algorithm benchmarking.

CHAPTER 4

BENCHMARKING OPTIMIZATION ALGORITHMS ACCORDING TO THEIR TUNABILITY

Benchmarking optimization algorithms is significant to both researchers and practitioners. Researchers use benchmarking in order to assess new optimization algorithms and algorithm modifications. For practitioners, benchmarking results provide useful information on which optimization algorithm to use for the optimization problem they are engaged with. The benchmarking of optimization algorithms is however complicated by many factors. The performance of optimization algorithms is sensitive to characteristics of the problem to be tackled (Wolpert and Macready, 1997), and the termination criteria imposed. Therefore, if a practitioner uses an algorithm well suited to a certain objective functions and constraints, there is no guarantee of favorable performance on another problem with different characteristics.

Another important benchmarking consideration is that of the effect of CPVs. Here the philosophy is followed that an optimization algorithm is an idea or an approach to optimization, and the CPVs specify how exactly that idea is executed. Consider a genetic algorithm (GA) which uses the nature-inspired processes of selection, mutation and crossover to solve an optimization problem. A GA's CPVs specify the finer details of the search, by controlling factors such as the population size, the mutation rate, and the manner in which crossover is conducted. Benchmarking an optimization algorithm according to one CPV tuple can therefore be misleading, since that algorithm's idea is not being benchmarked but rather only one instance of that idea. Moreover, is that performance measured representative of an optimization algorithm's suitability to a given optimization problem and termination criteria as a whole, or is the performance measured representative of those CPVs only. Similarly, out of all the CPVs and options available, how challenging is it to get favorable performance out of the algorithm being benchmarked for specific problem characteristics and termination criteria? Questions of this nature are particularly significant to optimization practioners who employ automated al-

gorithm configuration (López-Ibáñez and Stützle, 2012) techniques. In particular information regarding the tunability of an optimization algorithm to certain problem types, should aid automated algorithm configuration practitioners in selecting which algorithm to tune to the selected training problems.

This chapter's contribution is to investigate algorithm benchmarking through tuning, with the suitability of an algorithm to an optimization problem being gauged, according to that algorithm's tunability to that problem's search space characteristics and termination criteria. Benchmarking via tunability is motivated by the notion that if an optimization algorithm is well suited to a given problem, then not only will there be CPVs which result in good performance, but those CPVs will be easy to find. Benchmarking via tunability is therefore different to comparing tuned algorithms, since the proposed approach does not only entail the comparison of algorithms under even tuning, but also incorporates the difficulty of computational effort to obtain those tuned performances. Incorporation of tuning effort is a fundamental difference, as is elaborated on further in Section 4.3.

The outline of this chapter is as follows; Preliminary definitions and benchmarking consideration are discussed in Section 4.1. Current benchmarking practices and related work are then presented in Section 4.2. Benchmarking via tunability is then discussed in Section 4.3. Numerical experiments to investigate the effectiveness of benchmarking via tunability then follow in Section 4.4.

4.1 Preliminaries

Before related benchmarking practice is discussed, the prerequisite definitions and concepts are presented. Among these definitions is what constitutes an optimization algorithm with regard to both stochastic and deterministic algorithms. After this, benchmarking is discussed in the context of these definitions.

4.1.1 Definitions

The focus of this chapter is on benchmarking optimization algorithms for solving real-valued single objective optimization problems. For these problems an optimization algorithm needs to minimize or maximize an objective function f, where $f : \Re^{n_x} \mapsto \Re$ and n_x is the dimensionality of the search space. A problem can either be unconstrained or constrained. For the constrained case an optimization algorithm needs to search a subset of $\Re^{n_x}$.

Central to our definition of an optimization algorithm is the concept of a deterministic search process. A deterministic search process is a procedure which, when applied to an optimization problem with given starting conditions and termination criteria, produces the same result every time it is run. Readers familiar with the no free lunch (NLF) theorems for optimization (Wolpert and Macready, 1997), can think of this deterministic search process as equivalent to the parameterless deterministic algorithm described in the NFL theorems. The performance of a deterministic search process depends upon both the optimization problem at hand, and the specified termination criteria (Wolpert and Macready, 1997), since a search process consists of search mechanics which, although beneficial for certain problems, are detrimental on others.

Consider a hill climbing and a hill descending search process applied to a minimization problem. For the general case it is safe to assume that if the minimization problem is unimodal, then the hill climbing algorithm will be out-performed by the hill descending algorithm, while the opposite will hold true if the algorithms are applied to a multi-modal problem. Further complicating the comparison of deterministic search processes, is sensitivity to termination criteria such as OFE budgets.

Here, an optimization algorithm is viewed as a set of deterministic search processes unified according to a central philosophy or idea. When a deterministic optimization algorithm is applied to an optimization problem, the algorithm's control parameter values (CPVs) specify which of the deterministic search processes comprising the optimization algorithm is executed. Given the sensitivities to both optimization problem characteristics and termination criteria, CPVs are desirable since they allow practitioners to easily access different deterministic search processes. Each of these deterministic search processes is better suited for different problems, therefore allowing the practitioner to tackle a large variety of optimization problems using one optimization algorithm. CPVs are therefore convenient, as the practitioner does not need to download or implement a new deterministic search process for each new problem and termination criteria tackled.

Stochastic algorithms can also be represented through a set of deterministic search processes. In particular, the stochastic search process used is specified by the algorithm CPVs. This stochastic search process can be represented by a set of deterministic search processes, where each deterministic search process has a certain likelihood of being selected. Consequently, a stochastic optimization algorithm also describes a set of deterministic search processes, where the CPVs and the random state specify which deterministic search process is applied to an optimization problem. Therefore, a deterministic and a stochastic optimization algorithm can both be represented through a set of deterministic search processes, sets which are often infinite in size.

The infinite set of deterministic search processes representing an optimization algorithm, is not the same as the set of all deterministic search processes, which is also infinite in size. To demonstrate this principle, consider the set of real numbers between 2 and 3. This set between 2 and 3, although of infinite size, does not contain all real numbers. In the same way, an optimization algorithm which is an infinite set of deterministic search processes unified according to a central idea, is not the same as the set of all deterministic search processes.

4.1.2 Benchmarking in the Context of the No Free Lunch Theorems

The NFL theorems show that comparing search processes over the set of all problems is unnecessary, since all search processes are equivalent with one out-performing the other on exactly half of the set of all problems. This is provided that the search processes in question do not revisit the same points in the decision space. Similarly, the NFL theorems prove that two stochastic search processes are equivalent to each other when compared over all problems. Differentiation between search processes is possible however if a subset of all possible problems or a class of problems is considered.

The NFL theorems therefore indicate that if a practitioner does not incorporate any problem

specific knowledge when selecting a search process, the success achieved or lack there-off, is purely up to chance. Practitioners should therefore, when possible, incorporate knowledge of the problem at hand when selecting an optimization search process. For example, if the objective function is based on an engineering simulation, then knowledge of physics behind this simulation could be used. Alternatively, if the practitioner has in the past tackled optimization problems with similar characteristics and termination criteria, knowledge of which search processes worked well previously may also prove useful. This intuition may allow a practitioner to discern which problem class the problem being tackled belongs to. However, since these factors rely on the intuition and experience of the practitioner, selecting an appropriate search process for the problem at hand is still a science and an art.

For the purpose of this chapter, it is assumed that a practitioner is able to incorporate problem specific knowledge as to identify which class of problems the problem being tackled belongs to. Once a problem class is identified, an algorithm with search processes which work well on that class of problems can be used. Accordingly, the task falls to algorithm developers to determine optimization algorithms well suited for a specified class of problems, not only with regard to problem characteristics but also in terms of termination criteria.

4.2 Related Work

The suitability of an optimization algorithm is normally gauged by running numerical experiments on a set of testing problems representative of the problem class of interest. The numerical results from these experiments are scrutinized to determine if certain algorithms are better suited than others to the problem class in question. Standard benchmarking practice is to gauge performance using fixed CPVs, where each algorithm being compared is run on all of the benchmarking problems using the same CPVs. The COmparing Continuous Optimizers (COCO; Hansen et al., 2010; Pošík et al., 2012) platform, which was used at the GECCO 2009 to 2013 conferences, is an example of a problem suite designed for benchmarking algorithms according to fixed CPVs.

The COCO platform assesses the performance of an optimization algorithm over a vast range of objective function characteristics and termination criteria. Specifically, the COCO problems are classified into five groups according to the characteristics of their objective functions. These groups of objective functions being separable, low or moderate conditioning, high conditioning and unimodal, multi-modal with adequate global structure, and multi-modal with weak global structure. Each problem group consists of four to five base objective functions, all of which are of generalizable search space dimension. An optimization algorithm is benchmarked by running that algorithm over multiple variants or instances of each of the base problems. These problem instances are transformed versions of the base problem with modifications made such as shifting the location of the optimum and rotating the search space. For each problem instance, the COCO framework records the number of OFEs required to reach different solution accuracy levels which are logarithmically spaced. Performance measures similar to data profiles (Moré and Wild, 2009) and performance profiles (Dolan and Moré, 2002) are then used to compare the algorithms.

Data profiles measure the ratio of additional OFEs required by an algorithm to obtain a specified accuracy level, compared to that of the best algorithm for that problem and accuracy level, for various accuracy levels. Data profiles are different from performance profiles which focus on one solution accuracy level only. Performance profiles are generated by determining the fraction of benchmarking problems an algorithm is able to solve given τ times the OFE used by the best algorithm for that problem, for various τ. These performance measures all require that the algorithms compared make use of the same CPVs throughout the experiments. In order to gauge performance for the various CPV choices available for optimization algorithms, we investigate an approach which makes use of control parameter tuning.

4.3 Benchmarking via Tunability

The philosophy behind benchmarking via tunability is that if an algorithm is well-suited to a problem, then it should be easy to determine CPVs which result in favorable performance. Therefore according to benchmarking via tunability, the suitability of an optimization algorithm to an optimization problem class and termination criteria, is equivalent to that algorithm's tunability to that optimization problem class. Formally, benchmarking via tunability entails the comparison of algorithms according to their tuned performances for various tuning budgets. Subsequently, the outcome of benchmarking via tunability depends upon the following factors:

- the tuning formulations (the CPVs to tune, the tuning bounds, ...) of each of the benchmarked algorithms,
- the tuning algorithm selected,
- the performance measure to tune to, and
- the tuning budgets selected.

The concept of a *normal* automated algorithm configuration practitioner could be used as a guideline for selecting these four factors. A *normal* automated algorithm configuration practitioner makes use of the standard implementations of an optimization algorithm and tunes the control parameters associated with that implementation, while having limited amount of computing resources at his/her disposal for tuning to the representative training problems. As such, using tuning budgets for benchmarking via tunability which range up to the computational effort equivalent of leaving a modern PC running overnight, would make sense according to the concept of a *normal* automated algorithm configuration practitioner. This choice is admittedly rather arbitrary, but reasonable, and other benchmarking practitioners may elect to choose a different upper limit for the tuning budget.

Incorporation of the tuning effort is a crucial part of benchmarking via tunability. Consider a completely generic optimization algorithm with control parameters which allow it to manifest all possible search processes. For example, a Turing complete programming language whose control parameters are the code characters. Given enough tuning this generic algorithm would be able to produce the optimal search process for the problem class being considered. Therefore, the solution to the best optimization algorithm according to optimal CPVs is any completely

generic algorithm. This solution is of little use to practitioners and algorithm developers, because of the intractable amount of computational resources required for the tuning of such a generic algorithm. On the other end of the scale, if zero tuning effort is allocated, benchmarking according to default CPVs results. Given these considerations, tuning effort needs to be incorporated.

The concepts of generalist CPVs (Smit and Eiben, 2010b) can be used to gauge an optimization algorithm's tunability to a problem class, and reduce the risk of over-tuning. Contrary to specialist CPVs which are tuned to a single problem, generalist CPVs are tuned to multiple problems as to get favorable overall performance. Tuning according to generalist CPVs therefore helps to reduce the risk of over-tuning, where CPVs are found which are over-specialized for specific problem characteristics. The drawback of determining generalist CPVs is that an extra layer of abstraction is added between the tuning results and the problem the algorithm is being tuned to. If a testing problem is of special interest and/or particularly well understood by the optimization practitioner, then information regarding the tunability of specialist CPVs is more insightful, than the tunability of generalist CPVs.

The generalist versus specialist discussion can also be addressed from a multi-objective perspective. If a search process is applied to n problems using k performance metrics, then $n \times k$ performance objectives result. The performance resulting from CPV tuples can then be compared according to Pareto dominance, with one CPV tuple dominating another, if it results in better performance for all of the $n \times k$ performance objectives. Since these objectives are typically conflicting, CPVs will often not dominate each other, with them being relatively non-dominated. If the NFL theorems hold, then the larger the number of problems tackled and the number of performance metrics, the more likely that CPV tuples are relatively non-dominated. Generalist CPVs can be viewed as a comparison method whereby CPV tuples are compared by aggregating the performance objectives into a scalar value and then comparing them. As such generalist CPVs allow overall comparison, but results in information being discarded. It is therefore important that both specialist and generalist CPV performance be considered.

Case studies to demonstrate benchmarking via tunability follow. These case studies are designed to demonstrate different advantages and disadvantages of benchmarking via tunability, compared to benchmarking using fixed CPVs.

4.4 Case Studies

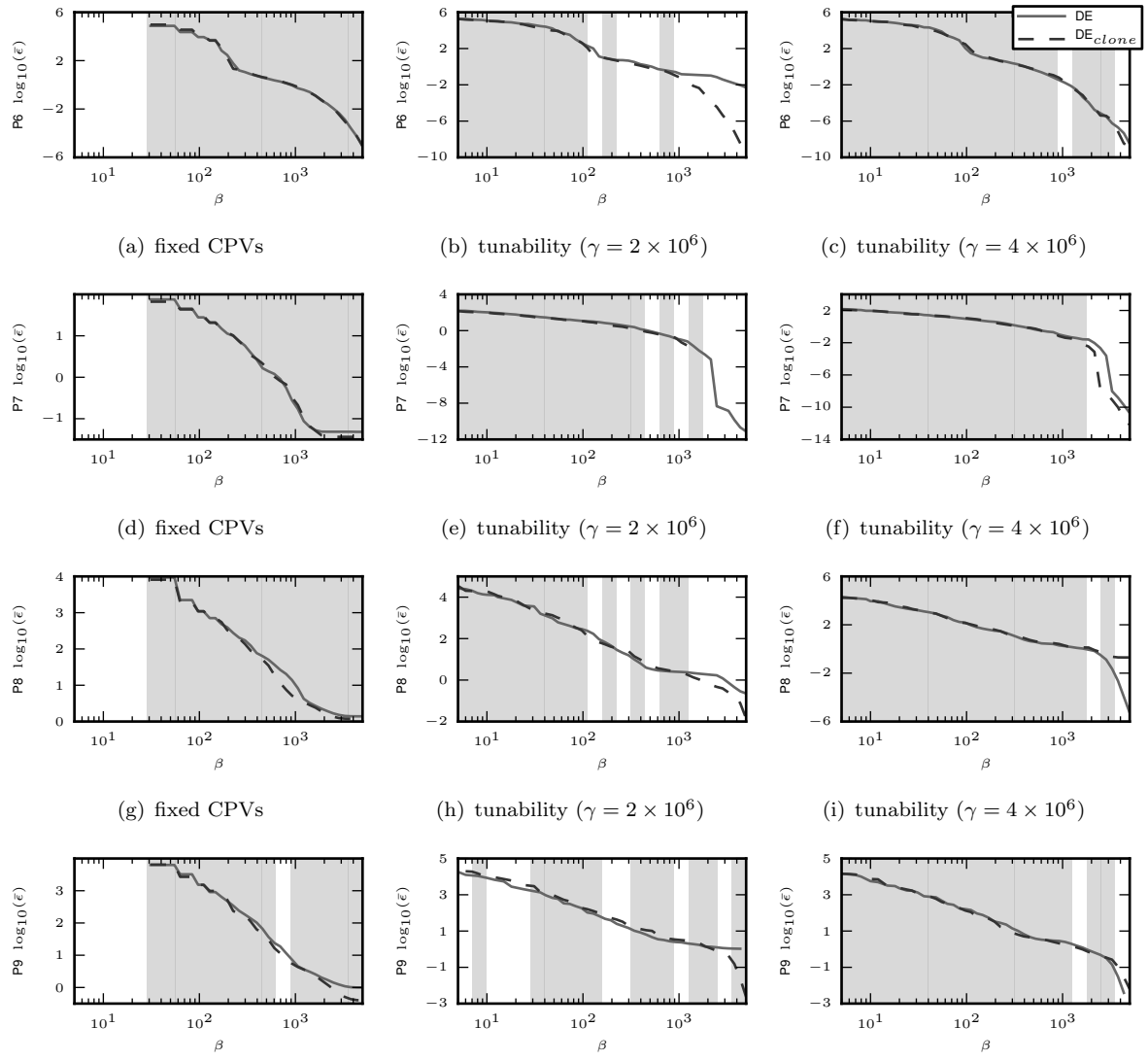
All of the case studies conducted make use of problems from the COCO platform, and entail comparison over multiple OFE budgets. Specifically, the first instance of selected COCO problems are used. Furthermore, as to keep the numerical experiments computationally light and therefore easily reproducible, the 6 dimensional versions of these COCO problems are used. Using a low dimensionality reduces the computational cost of evaluating the objective functions, and also reduces the range of OFE budgets which need to be considered. A lower OFE budget range can be considered because it should take less OFEs to accurately approximate the solution to the lower dimensional version of the problem compared to the higher dimensional version.

4.4.1 Comparing an Optimization Algorithm against Itself

The first case study entails the benchmarking of an optimization algorithm against itself. Conventional fixed CPV benchmarking and benchmarking via tunability should indicate that this algorithm is equal to itself. The problems used are the low or moderate conditioning COCO problems, namely: the attraction sector COCO problem (P6), the step ellipsoidal problem (P7), the Rosenbrock problem (P8), and the rotated Rosenbrock problem (P9). The algorithm benchmarked against itself is Differential Evolution (DE; Storn and Price, 1997). DE is a population based algorithm designed to optimize problems with non-differentiable, non-linear objective functions, and has numerous control parameters which influence search behavior. Among these control parameters are the population size N, the scaling factor F, and the crossover rate C_r. The implemented version of DE uses the *rand/1/bin* (Storn and Price, 1997) scheme. Regarding the search bound constraints for the COCO problems of $[-5, 5]^6$, the implemented DE algorithm uses a strategy whereby the candidate decision vector generation process repeats until a valid design is determined. If the candidate decision vector process fails 10 times, then that individual is re-initialized.

DE is compared against itself DE_{clone}, for OFE budgets ranging between 5 and 5000. For statistical considerations a resampling size of 20 is used for comparing DE to DE_{clone} according to mean solution accuracy in terms of objective function values. A sample size of 20 is deemed acceptable given that MWUTs are to be conducted to check the statistical significance of the results. The DE parameters used for fixed CPV benchmarking are a N of 30, a C_r of 0.9 and an F of 0.5. The tMOPSO tuning algorithm is used for the benchmarking via tunability study. Based upon the work in Chapter 2, tMOPSO parameters are a population size of 10, an inertia factor 0.2, personal and global acceleration constants of 2.0, an OFE assessment overshoot factor of 2.0 and an OFE perturbation factor of 0.1. For noise handling a resampling interruption confidence of 0.9 is used. The choice of tMOPSO CPVs although important, is not critical, since both DE and DE_{clone} are to be tuned using the same settings. The application layer evaluation (γ) budget, which is the budget for the number of calls made by the algorithm being tuned on the objective function it is being tuned to, is set to 2×10^6 and 4×10^6. A γ of 2×10^6 is equivalent to assessing 20 CPV tuples up to 5000 OFEs using a resampling size of 20. Two different γ budgets are used to investigate if there is any significant difference in the results depending upon the tuning budget used. For the tuning of DE and DE_{clone}, the tuning initialization bounds are $N \in [5, 50]$, $C_r \in [0, 1]$ and $F \in [0, 1]$. After initialization the tMOPSO algorithm is allowed to explore outside these bounds, subject to $N \geq 5$, $C_r \in [0, 1]$ and $F > 0$.

Figure 4.1 shows that both benchmarking using fixed CPVs and via tunability indicate that DE and DE_{clone} are equal. For comparison according to fixed CPVs, MWUTs show that DE and DE_{clone} are statistically indistinguishable in terms of mean values given a confidence level of 95% for the null-hypothesis, for all the OFE budgets considered. Regarding comparison according to tunability, the results are more noisy with false differences being observed at some OFE budgets. These false differences are small and scarce enough that it can still be discerned from the tunability results that DE and DE_{clone} offer very similar optimization performance. For the comparison of DE versus DE_{clone}, benchmarking using fixed CPVs produced a less noisy result at a lower computational cost. Specifically, benchmarking using fixed CPVs used one 20^{th}



(a) fixed CPVs (b) tunability ($\gamma = 2 \times 10^6$) (c) tunability ($\gamma = 4 \times 10^6$)

(d) fixed CPVs (e) tunability ($\gamma = 2 \times 10^6$) (f) tunability ($\gamma = 4 \times 10^6$)

(g) fixed CPVs (h) tunability ($\gamma = 2 \times 10^6$) (i) tunability ($\gamma = 4 \times 10^6$)

(j) fixed CPVs (k) tunability ($\gamma = 2 \times 10^6$) (l) tunability ($\gamma = 4 \times 10^6$)

Figure 4.1: Comparison of DE against DE_{clone} according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95% .

of the γ of that used by the benchmarking via tunability at a γ budget of 2×10^6. If tMOPSO's γ budget is increased to 4×10^6 then the noise level on the tunability results is reduced, as is also shown in Figure 4.1.

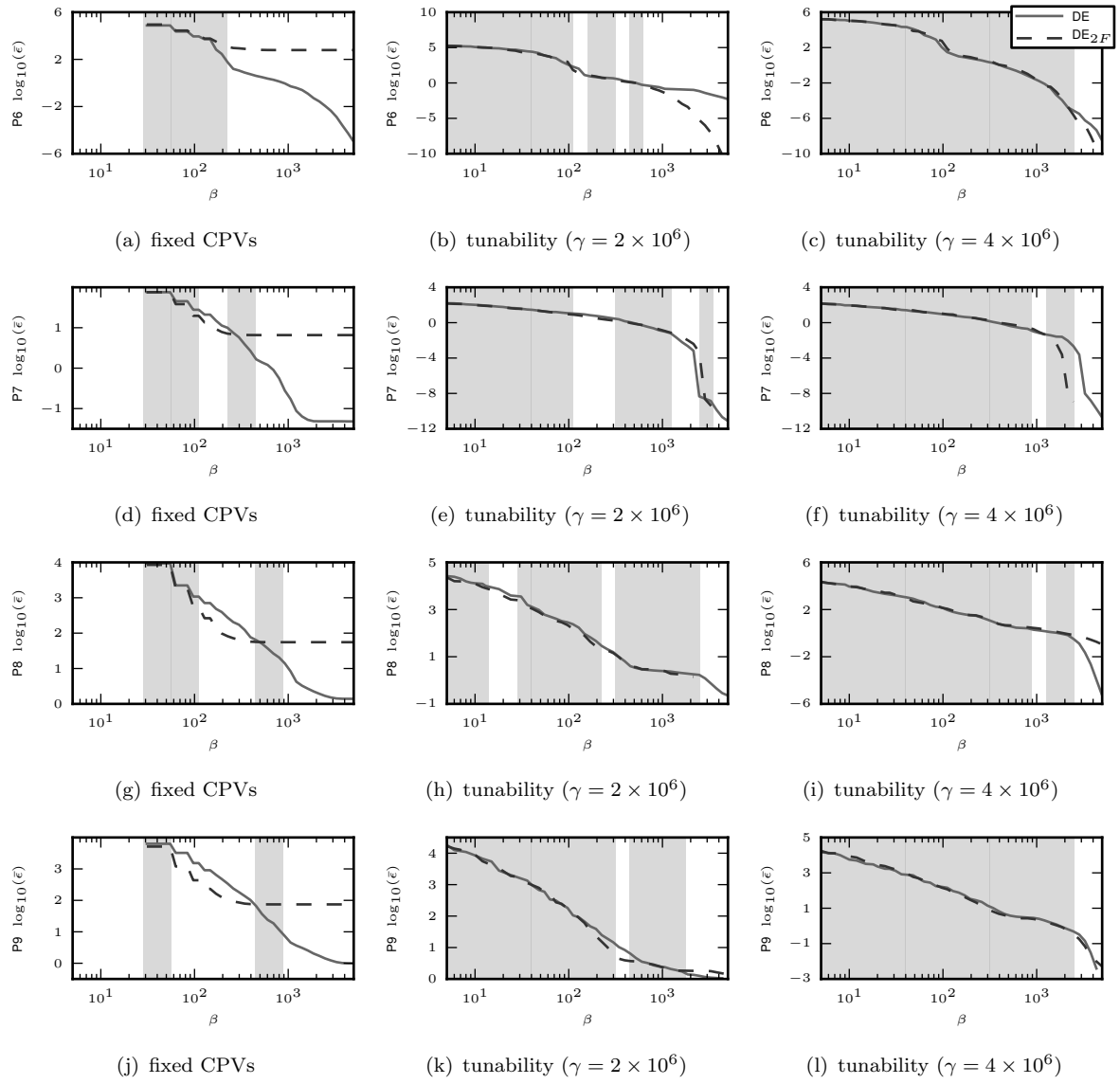
Next, the scenario is considered where DE is compared against itself with different CPVs and different tuning bounds. This scenario is added to illustrate the sensitivity of benchmarking via fixed CPVs to the CPVs chosen for the algorithm being benchmarked, and benchmarking via tunability's sensitivity to the tuning bounds of the algorithms being compared. DE is benchmarked against a variant of DE, DE_{2F}, where DE_{2F} is identical to DE except in one regard. For DE_{2F} the scaling factor's effect is halved to promote exploitation while reducing exploration. The parameters passed to DE_{2F} and the tuning problem formulation are the same as that for DE. Figure 4.2 shows the comparison of DE versus DE_{2F}, according to performance using fixed CPVs, and according to tunability.

Comparison based on fixed CPVs, shows that DE_{2F} does perform better at lower OFE budgets and worse at higher OFE budgets on all four of the problems considered. However, since we know that DE and DE_{2F} are the same algorithm, we know that this result is an artifact of the CPVs chosen for the fixed CPV comparison, and not a result of any conceptual difference between DE and DE_{2F}. Comparison based on tunability was less susceptible to the error in tuning initialization bounds. Since DE and DE_{2F} only differ with regard to F, the CPVs determined by tMOPSO for DE and DE_{2F} should primarily only differ in terms of F, with N and C_r being comparable. However, as shown in Figure 4.3, different N and C_r values were determined, in addition to different F values. Part of this difference is attributed to noise from the resampling process used to approximate the mean solution error, and the stochastic elements of tMOPSO. The rest of the difference is attributed to DE_{2F}'s initialization bounds which were $F \in [0, 1]$, instead of $F \in [0, 2]$. Even with this constructed error in the tuning bounds, benchmarking via tunability was still able to correctly gauge DE and DE_{2F} as equivalent.

The tunability results from this case study are useful to a practitioner tackling a problem similar to P6, P7, P8, or P9, in an individual sense. If performance in an overall sense is of interest, an alternative tunability benchmarking approach is required.

4.4.2 Benchmarking over a Group of Problems

In order to gauge algorithm performance on a group of problems according to tunability, the concept of Generalist CPVs is used. Generalist CPVs were introduced in the context of tuning to multiple problems each at a single OFE budget. The concept of anytime parameters (Radulescu et al., 2013), which perform well over a range of OFE budgets, could be used in conjunction with generalist CPVs, to tune for anytime-generalist CPVs. However, acting under the assumption that the practitioner knows which OFE budget is going to be used, the tuning in this case study aims to determine multiple generalist CPV tuples each of which is well suited to a different OFE budget. Tuning according to multiple problems over multiple OFE budgets can be efficiently achieved using many objective tuning. In particular, the tuning problem can be formulated with an objective for each problem being tuned to, together with a speed objective. As such, tuning an algorithm to COCO problems P6 to P9 over multiple OFE budgets, can be achieved



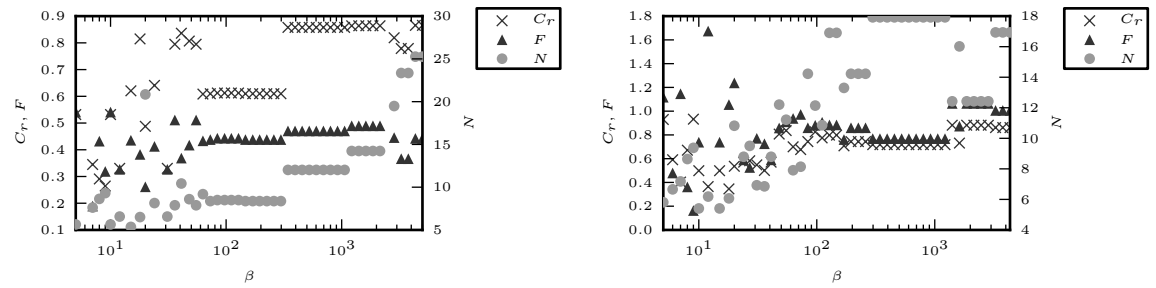
(a) fixed CPVs (b) tunability ($\gamma = 2 \times 10^6$) (c) tunability ($\gamma = 4 \times 10^6$)

(d) fixed CPVs (e) tunability ($\gamma = 2 \times 10^6$) (f) tunability ($\gamma = 4 \times 10^6$)

(g) fixed CPVs (h) tunability ($\gamma = 2 \times 10^6$) (i) tunability ($\gamma = 4 \times 10^6$)

(j) fixed CPVs (k) tunability ($\gamma = 2 \times 10^6$) (l) tunability ($\gamma = 4 \times 10^6$)

Figure 4.2: Comparison of DE and DE_{2F} according to mean solution error ($\bar{\epsilon}$) achieved at different OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%



(a) DE (b) DE_{2F}

Figure 4.3: Control Parameters found by tMOPSO for a γ of 4×10^6 on P6 for DE and DE_{2F}

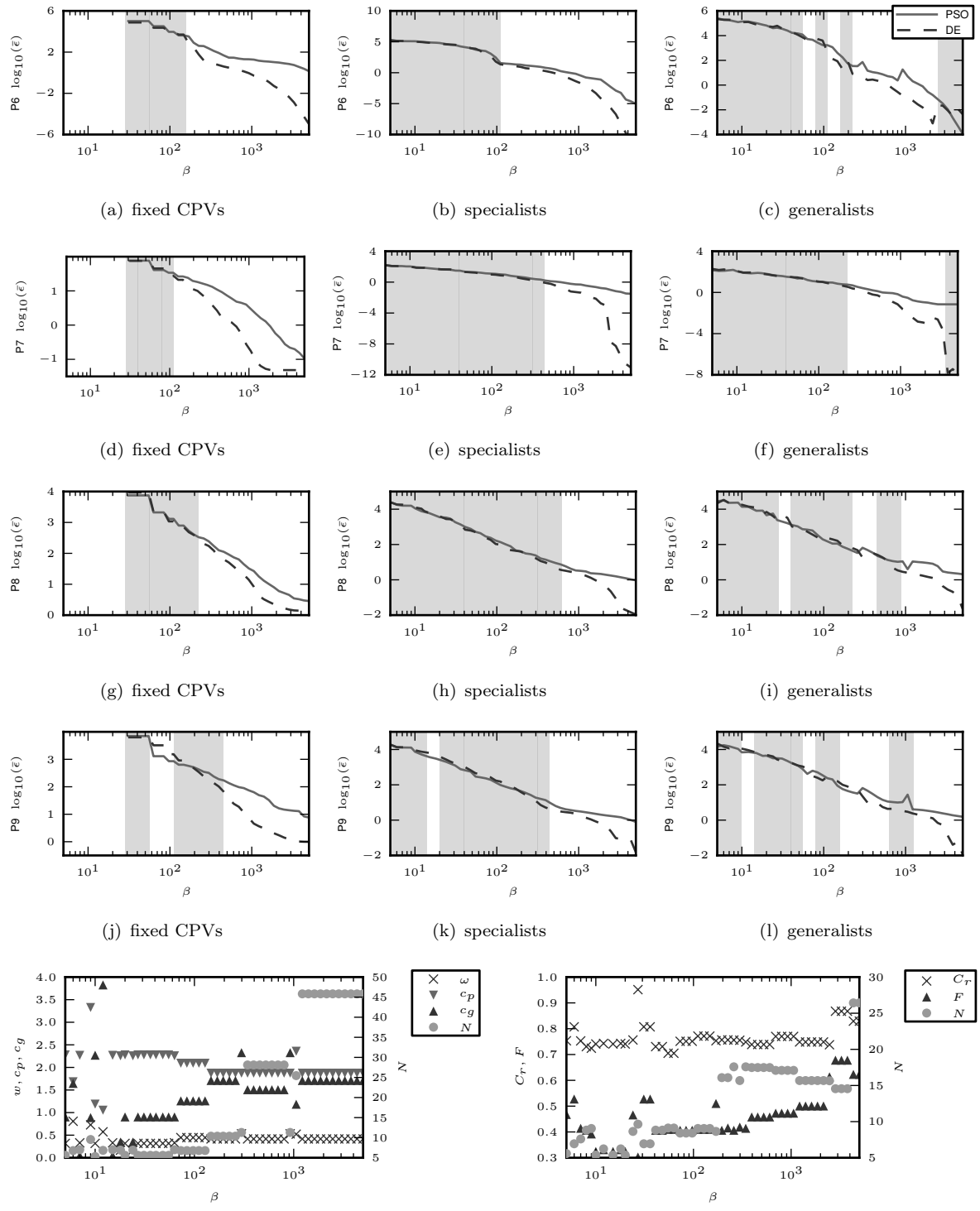
by solving a 5 objective many objective problem. The MOTA algorithm which was specifically developed for these scenarios is used in this case study.

To demonstrate benchmarking via tunability according to generalist CPVs, the DE algorithm from the previous case study is compared to a particle swarm optimization (PSO; Eberhart and Kennedy, 1995) algorithm, over different COCO problem groups. The three groups of COCO problems that are used in this case study, are the low or moderate conditioning group (P6-P9), the unimodal high conditioning group (P10-P14) and the multi-modal with adequate global structure group (P15-P19). The PSO algorithm benchmarked uses global neighborhood topology, zero initial velocities, a fixed inertia factor, and the same constraint handling strategy as DE. The PSO CPVs varied are the swarm size N, the inertia factor ω, and the personal and global acceleration factors, c_p and c_g respectively. The initialization bounds used for these CPVs are $N \in [5, 50]$, $\omega \in [0, 1]$, $c_p \in [0, 3]$ and $c_g \in [0, 3]$. PSO tuning constraints are $N \geq 5$, $\omega \in [0, 1]$, $c_p \geq 0$, and $c_g \geq 0$. Both the initialization bounds and tuning constraints are chosen based upon the studies presented in Shi and Eberhart (1998) and Clerc and Kennedy (2002). The CPVs used to benchmark PSO according to fixed CPVs are chosen based on Clerc and Kennedy (2002), as $N = 30$, $\omega = 0.7$, $c_p = 2$ and $c_g = 2$. The fixed CPVs and tuning formulation for DE are the same as in the previous case study.

MOTA is set up to determine generalist CPVs which perform well according to a weighted sum scalarization of objective solution errors in the normalized objective space. MOTA's subproblem are set up as defined as, one bi-objective decomposition for determining the generalist CPVs for various OFE budgets, while the other bi-objective decompositions focus on specialist CPVs and all leave-one-out combinations. The MOTA parameter values used for the tunability tests are a DE scaling factor of 2, a crossover rate of 0.7, an OFE perturbation factor of 0.2, and an OFE overshoot factor of 2. For statistical considerations a resampling size of 20, and resampling interruption confidence level of 0.9 are used. MOTA's γ budget for each bi-objective decomposition is 4×10^6 which corresponds to evaluating 40 CPV tuples up to 5 000 OFEs given a resampling size of 20.

Benchmarking according to fixed CPVs and according to tunability produced different results, as shown in Figures 4.4 to 4.6. Comparison of DE and PSO according to the fixed CPVs indicates that on certain problems PSO performs better at low OFE budgets, while DE performs better in all other cases. Comparison according to specialist CPV performance indicates that up until around 10^3 OFEs, both PSO and DE offer comparable performance when using CPVs well-suited to the problem at hand, while for OFE budgets higher than this DE begins to out-perform PSO, except on P15 and P19 where PSO outperforms DE at higher OFE budgets. Similarly, comparison of the generalist CPVs indicate that DE and PSO are comparable up to 10^3 OFE budgets. Analysis of the generalist CPVs determined by MOTA, which are also shown in Figures 4.4 to 4.6, is done to scrutinize the tunability results. For OFE budgets higher than 20 OFEs, CPV trends such as an increasing optimal population size are observed. This observation is consistent with a previous study (Dymond et al., 2014).

Both DE and PSO have CPVs recommended for OFE budgets up to 10 and 20 respectively, which result in the algorithms only performing one iteration. Since both algorithms have the same initialization procedure, where the population or swarm is assigned decision vectors



(a) fixed CPVs (b) specialists (c) generalists

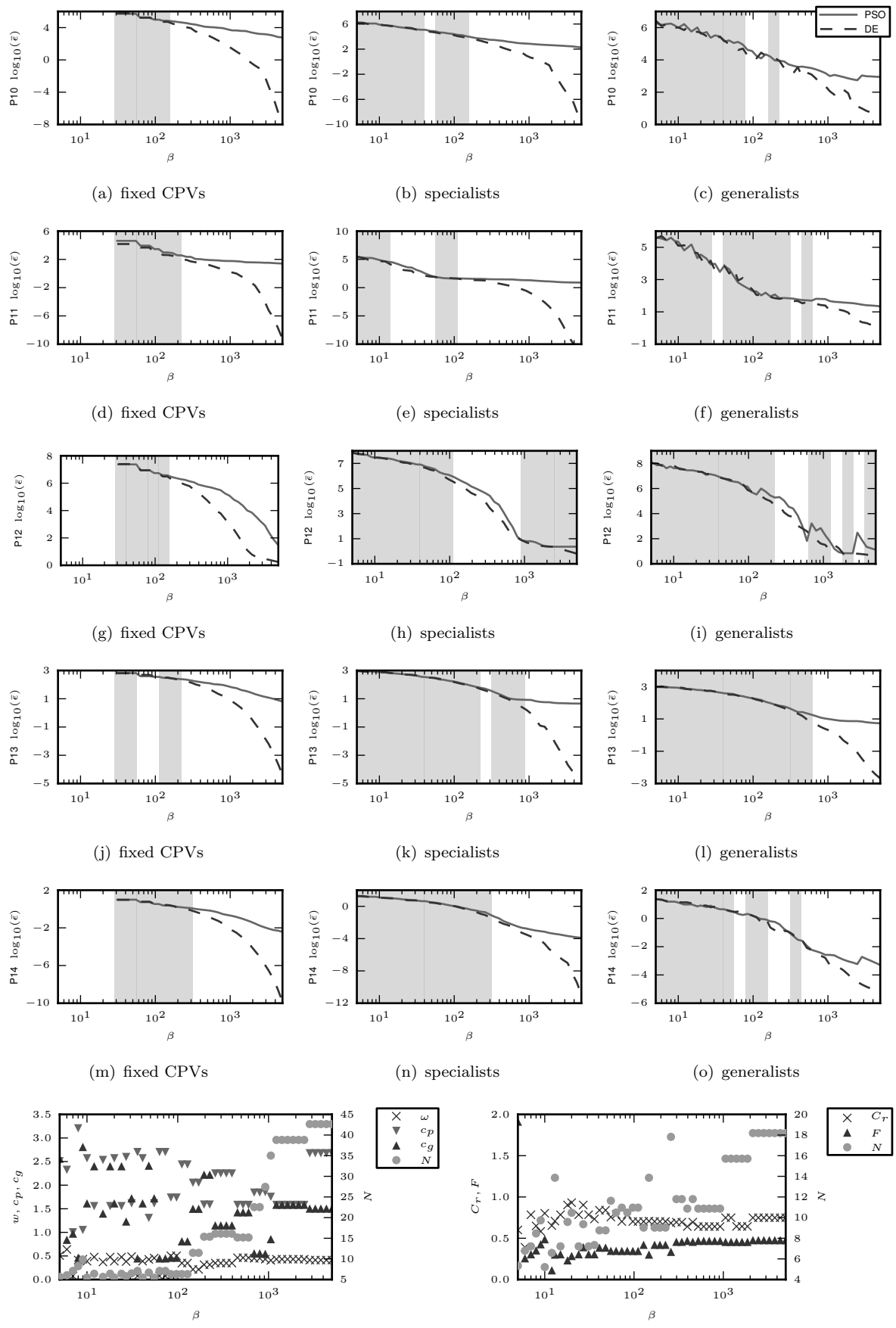
(d) fixed CPVs (e) specialists (f) generalists

(g) fixed CPVs (h) specialists (i) generalists

(j) fixed CPVs (k) specialists (l) generalists

(m) PSO generalist CPVs (n) DE generalist CPVs

Figure 4.4: Comparison of PSO and DE on COCO problems P6 to P9 according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.



(a) fixed CPVs (b) specialists (c) generalists

(d) fixed CPVs (e) specialists (f) generalists

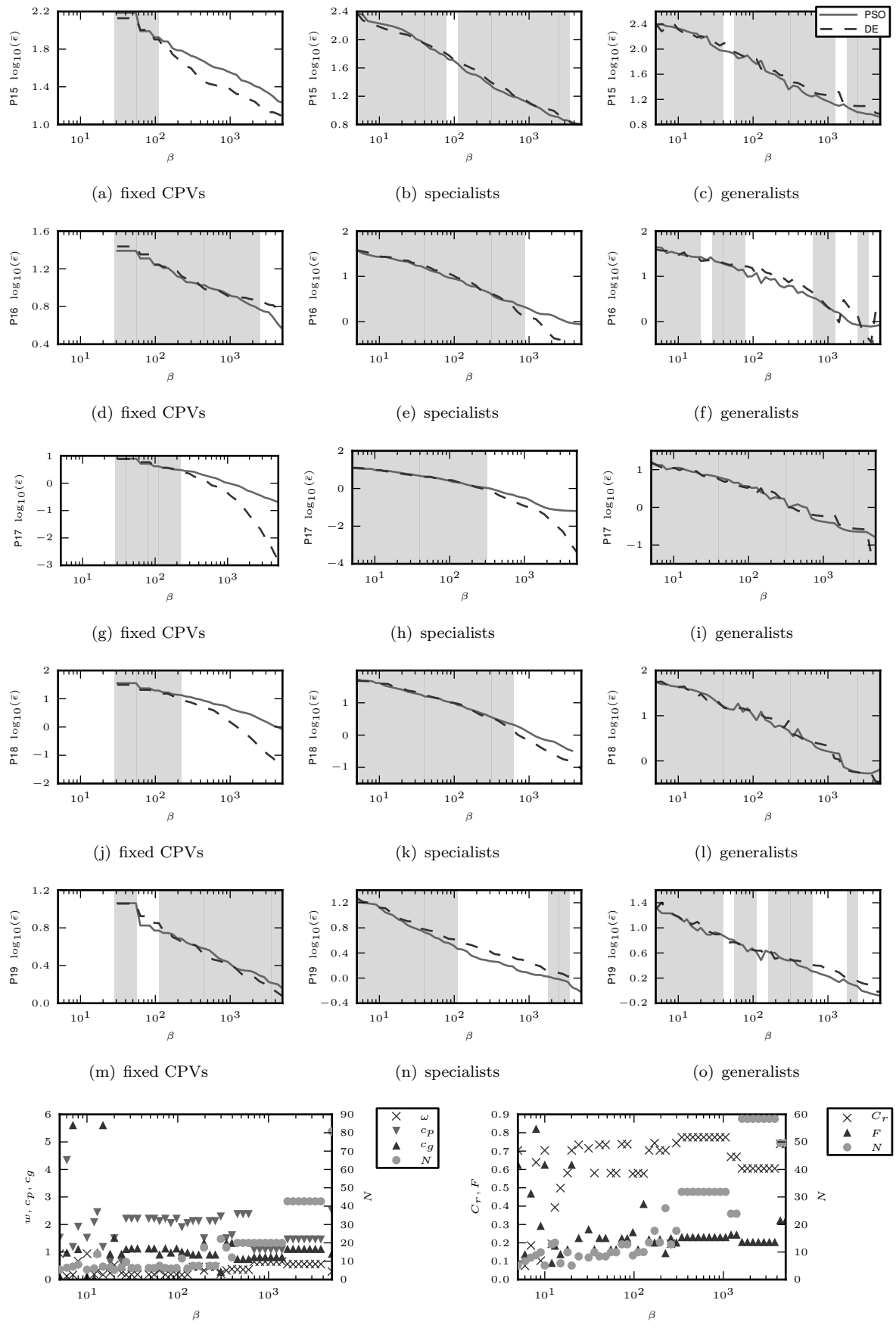
(g) fixed CPVs (h) specialists (i) generalists

(j) fixed CPVs (k) specialists (l) generalists

(m) fixed CPVs (n) specialists (o) generalists

(p) PSO generalist CPVs (q) DE generalist CPVs

Figure 4.5: Comparison of PSO and DE according on COCO problems P10 to P14 to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.



(a) fixed CPVs (b) specialists (c) generalists

(d) fixed CPVs (e) specialists (f) generalists

(g) fixed CPVs (h) specialists (i) generalists

(j) fixed CPVs (k) specialists (l) generalists

(m) fixed CPVs (n) specialists (o) generalists

(p) PSO generalist CPVs (q) DE generalist CPVs

Figure 4.6: Comparison of PSO and DE according on COCO problems P15 to P19 to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.

throughout the search space using a uniform random distribution, these tuning results indicate that at a very low OFE budget, a random search may be equivalent to DE or PSO, given the constraint that $N > 5$.

4.4.3 The Equivalence of Algorithms at Very Low OFE Budgets

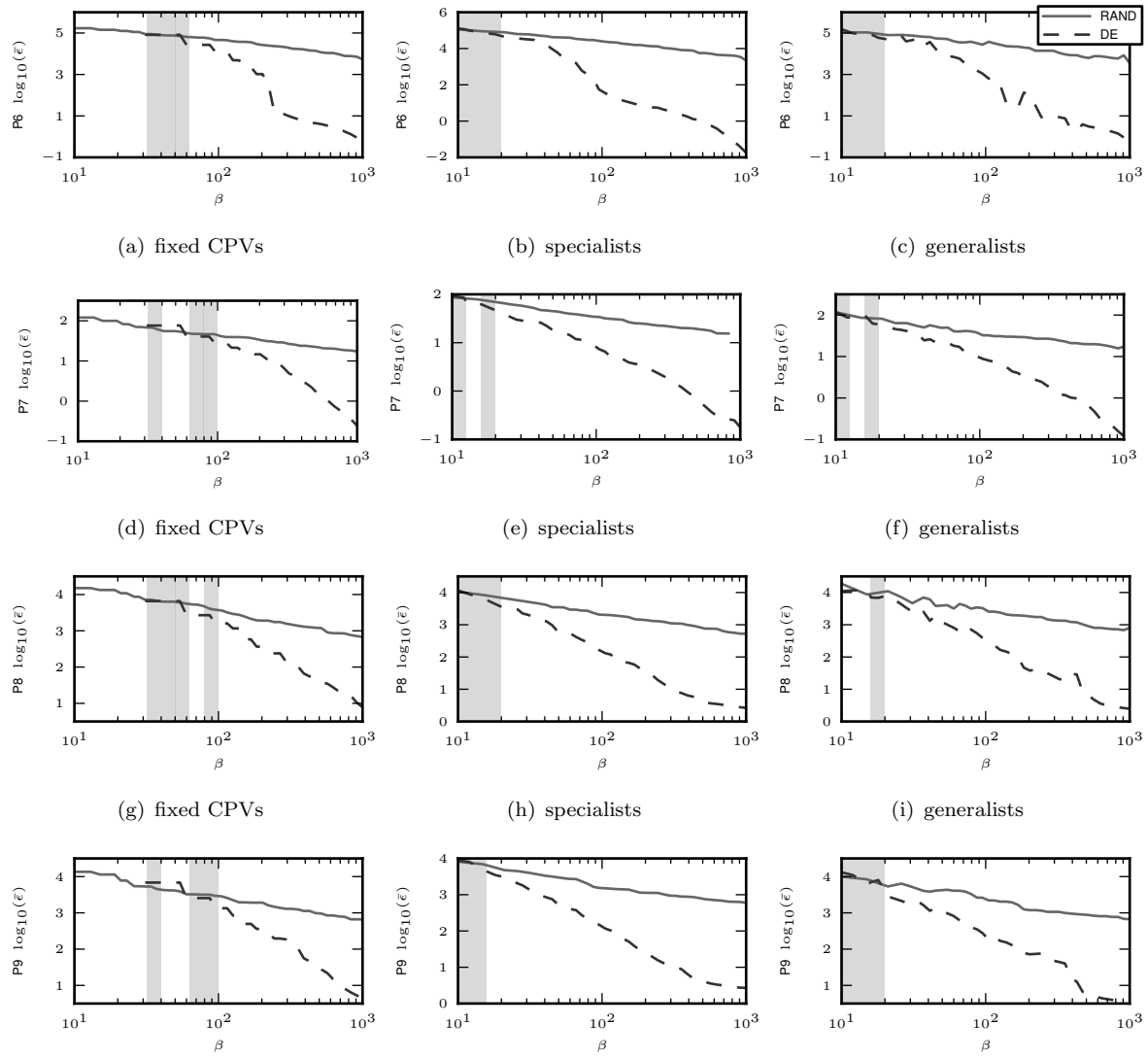
To investigate the equal performance of DE and PSO at very low OFE budgets, DE is compared against a random search algorithm. The hypothesis supporting this investigation is that during the first generation of DE and the first iteration of PSO, both algorithms generate solutions inside the search space bounds using a uniform random distribution, with no opportunity yet for the use of either evolutionary- or swarm operators. By comparing DE against a random search algorithm, this case study aims to determine at which OFE budgets for COCO problems P6 to P9, DE begins to outperform a random search algorithm. The random search algorithm, RAND, searches for the optimum using a uniform random distribution to generate new candidate decision vectors. The generation of candidate decision vectors continues until the OFE budget is exhausted.

The OFE budgets investigated range between 10 and 10^3. For benchmarking via tunability, MOTA is used with the same setting as in the DE versus PSO case study, with the exception that the γ budget is reduced to 10^6. This γ budget is equivalent to assessing 50 CPVs up to 10^3 OFEs using a resampling size of 20. Since RAND does not have any CPVs, MOTA tuning essentially entails running RAND until the γ budget has been exhausted and recording the best resampling values. Tuning of RAND is necessary as the performance of RAND and DE is approximated using 20 sample runs, and is therefore noisy. RAND is therefore tuned using the same number of γ, as to reduce the influence of the resampling noise on the tunability comparison.

Comparison using fixed CPVs shows that RAND is equivalent to DE up to 90 OFEs for problems P6 to P9. The question arising from this observation is if DE itself is equivalent to random search up to 90 OFEs, or is this threshold of 90 OFEs an artifact of CPVs used in the comparison. Benchmarking via tunability naturally addresses this question, showing that the value of 90 OFEs is indeed an artifact of CPVs chosen and that DE can outperform random search for OFE budgets lower this provided that appropriate CPVs are chosen. Both the specialist and generalist tunability comparison are also shown in Figure 4.7.

The RAND versus DE and the DE versus PSO case studies have shown that the algorithms investigated are equivalent up to certain OFE budgets, provided that each algorithm has effective CPVs specified. For DE versus PSO this threshold OFE budget was about 10^3 OFEs, and for DE versus RAND the threshold was less then 10^2 OFEs. This equivalence originates due to all algorithms compared starting their optimization in the same manner, by randomly generating initialized points over the entire search space.

Continuing with the theme of comparing optimization over multiple OFE budgets, the next case compares algorithms developed for different orders of OFE budgets.



(a) fixed CPVs (b) specialists (c) generalists

(d) fixed CPVs (e) specialists (f) generalists

(g) fixed CPVs (h) specialists (i) generalists

(j) fixed CPVs (k) specialists (l) generalists

Figure 4.7: RAND versus DE according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.

4.4.4 Comparing of Optimization Algorithms Developed for Different OFE Budget Ranges

DE is compared to the efficient global optimization (EGO, Jones et al., 1998) algorithm, where EGO is an algorithm specifically designed for low OFE budget applications. Benchmarking via tunability is particularly well-suited for this comparison since the standard CPV recommendations for these algorithms are for different orders of magnitude of OFE budgets, with DE developed for high OFE budget applications and EGO developed for low OFE budget applications. Comparison based upon DE and EGO's standard CPVs is therefore unable to answer questions in the line of:

- if EGO's parameters are changed for better performance at higher OFE budgets can it outperform DE at high OFE budgets, or alternatively
- if DE uses CPVs which are effective at low OFE budgets can it compete against EGO at low OFE budgets?

EGO uses surrogate or meta-modeling together with the concept of maximum expected improvement (EI) in order to search for a problem optimum. The rationale behind EGO is to explore the search space according to where the fitted meta-model indicates the minimum is, while also trying to improve the fit of the meta-model. EGO uses the EI from a fitted Kriging model to achieve this goal. The EGO implementation from the DiceOptim R package (Roustant et al., 2012)[1] is used for this case study. DiceOptim has a variety of Kriging parameters available for the user to specify, including type of covariance functions used and the trend basis functions. For the EGO versus DE comparison ordinary Kriging is used, which consists of a constant trend basis together with Gaussian co-variance kernels. Furthermore, EGO is set up to use Latin hypercube sampling to generate the N_0 decision vectors used for fitting the first Kriging model. The next candidate decision vector is chosen to maximize the EI of the fitted model, after which the model is refit. This candidate decision vector process repeats until the OFE budget has been exhausted.

EGO and DE are compared for OFE budgets ranging from 10 to 300 OFEs, using both fixed CPV and tunability benchmarking. An upper limit of 300 OFEs is chosen due to computational considerations. For each OFE after the initial N_0 points have been generated, EGO needs to solve an optimization problem as to determine where the point of maximum EI is, and then refit the Kriging model. For DiceOptim this EI optimization problem is solved through a GA which uses gradients from the Kriging model as to speed convergence. Even so, the EGO implementation benchmarked is too computationally expensive[2] to compare at OFE budgets higher then 300. To be fair it should be noted that although the computational overhead of EGO is problematic from a benchmarking perspective, EGO's computational overhead is inconsequential compared to the computational cost of evaluating the expensive objective and constraint functions for which EGO was designed.

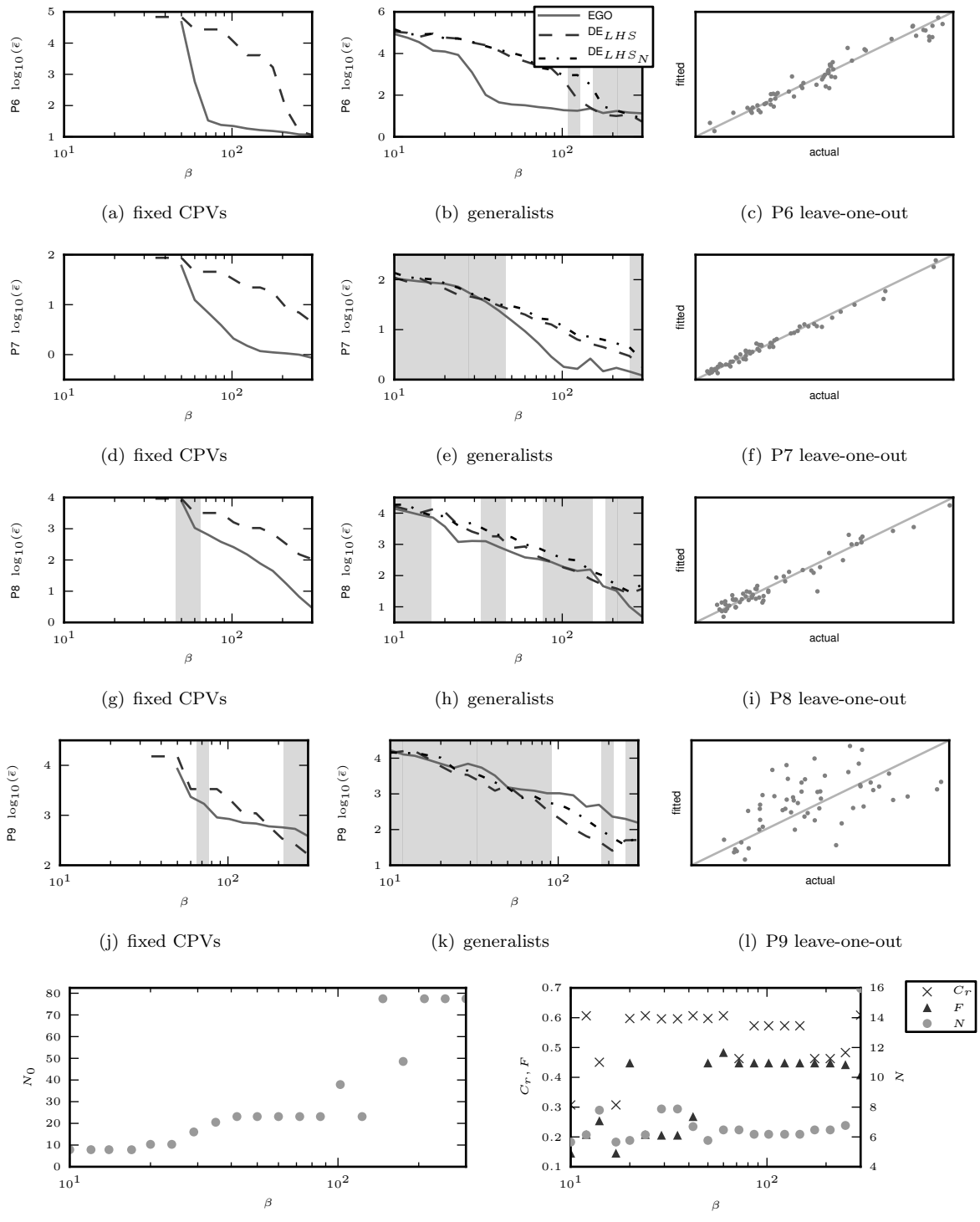
[1] DiceOptimum version 1.4 is used in these experiments

[2] Running the DiceOptim EGO up to 1000 OFEs using a resampling size of 20 takes just less than two days using a single processor on an Intel© Core™ i7-4700MQ processor

To nullify the effects of starting conditions, EGO is compared to DE_{LHS} where DE_{LHS} is the same as the DE algorithm used in the case studies so far, with the exception of Latin hypercube sampling being used to generate the initial population. Therefore neither EGO nor DE_{LHS} has an initial advantage with both algorithms starting their search in the same manner. For benchmarking according to fixed CPVs, DE_{LHS} uses the same CPVs as in the PSO versus DE study. EGO uses a N_0 of 50 points, a value which is used in the DiceOptim documentation for solving a 6 dimensional example problem. For benchmarking via tunability, a generalist and specialist comparison is conducted using MOTA as the tuning algorithm. Due to EGO's high computational overhead, MOTA's γ budget is set to 10^5 which is approximately equivalent to running 17 CPV tuple assessments up to 300 OFEs using a resampling size of 20. The number of CPV tuples assessed will be greater than 17 since MOTA does not assess every CPV tuple up to the maximum OFE budget, and because of MOTA's use of preemptively terminating resampling. The only CPV of EGO which is tuned is N_0 which control the number of initial points generated using Latin hyper cube sampling, with the initialization and search bound being $N_0 \in [7, 100]$, where 7 is the minimum number of points required to fit an ordinary Kriging model to a 6 dimensional problem. DE_{LHS} is tuned using the same bounds and tuning variables as the DE algorithm from the DE versus PSO case study. Additionally a DE_{LHS} variant, DE_{LHS_N}, is added which is the same as DE_{LHS} except that only the N control parameter is tuned. DE_{LHS_N} is added to see if DE_{LHS}'s tuning formulation of 3 variables compared to the 1 of EGO affects the obtained results. Similarly, to ensure a fair comparison, the DE_{LHS_N} bounds for N are the same as that of EGO's N_0.

In order to gauge which of the benchmarking approaches yields more accurate results, it is assumed that the benchmarked EGO is better suited than the benchmarked DE to problems which can be accurately modeled using ordinary Kriging. To determine which COCO problems can be accurately modeled using ordinary Kriging, leave-one-out validation tests are performed. A leave-one-out modeling validation check entails generating n_c decision vectors throughout the search space, typically using a process such as Latin hypercube sampling. Then for each of the n_c decision vectors, the model being validated is fitted to the other $n_c - 1$ decision vectors, after which the model objective function value predictions are compared to the actual objective function value for the decision vector left out of the fitting. For the leave-one-out checks a n_c of 60 and Latin hypercube sampling with $\mathbf{x} \in [-5, 5]^6$, are used.

The results for benchmarking via fixed CPVs, benchmarking via tunability, and the leave-one-out validation checks, are shown in Figure 4.8 to Figure 4.10. According to the fixed CPVs comparison EGO is better than DE_{LHS} for OFE budgets less then 200 OFEs for 11/14 COCO problems used, is equivalent to DE_{LHS} on 2/14 problems, and is worse than DE_{LHS} for one problem. For the benchmarking according to tunability comparison, EGO performs better than DE_{LHS} for OFE budgets less then 200 OFEs for 5/14 problems, worse than DE_{LHS} on 1/14 of the problems, and is statistically equivalent on the other 8/14 problems. The leave-one-out modeling validity checks show that P9, P12, P14, and P16 to P19 are poorly suited to ordinary Kriging modeling. Over these problems, benchmarking via tunability found EGO and DE_{LHS} to be equivalent, with one exception of P9 where DE_{LHS} performed better at certain OFE budgets. For benchmarking via fixed CPVs, EGO was found to be better than DE_{LHS} on the



(a) fixed CPVs (b) generalists (c) P6 leave-one-out

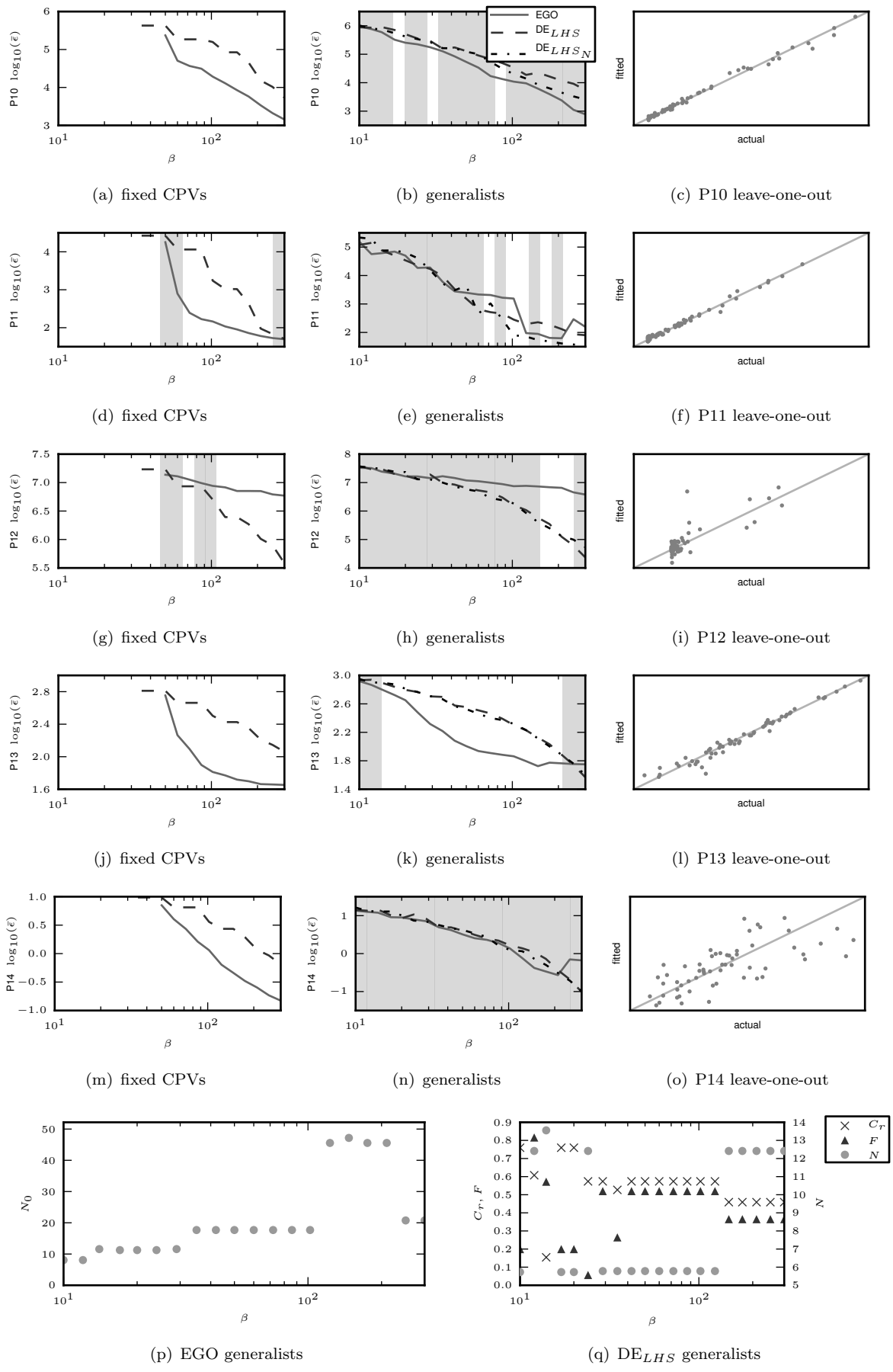
(d) fixed CPVs (e) generalists (f) P7 leave-one-out

(g) fixed CPVs (h) generalists (i) P8 leave-one-out

(j) fixed CPVs (k) generalists (l) P9 leave-one-out

(m) EGO generalists (n) DE_{LHS} generalists

Figure 4.8: Comparison of EGO and DE on COCO problems P6 to P9 according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.



(a) fixed CPVs (b) generalists (c) P10 leave-one-out

(d) fixed CPVs (e) generalists (f) P11 leave-one-out

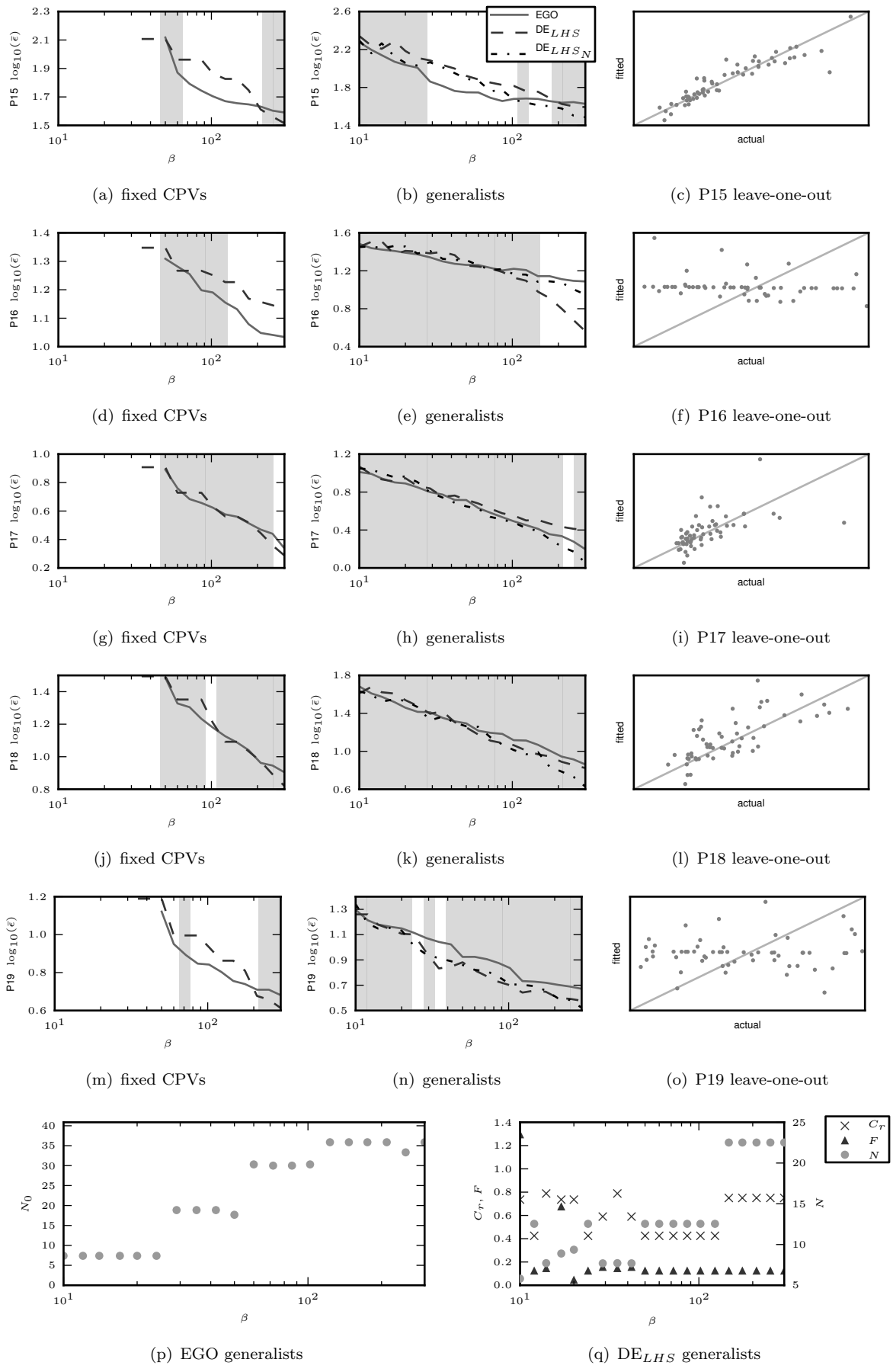
(g) fixed CPVs (h) generalists (i) P12 leave-one-out

(j) fixed CPVs (k) generalists (l) P13 leave-one-out

(m) fixed CPVs (n) generalists (o) P14 leave-one-out

(p) EGO generalists (q) DE_{LHS} generalists

Figure 4.9: Comparison of EGO and DE on COCO problems P11 to P14 according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.



(a) fixed CPVs (b) generalists (c) P15 leave-one-out

(d) fixed CPVs (e) generalists (f) P16 leave-one-out

(g) fixed CPVs (h) generalists (i) P17 leave-one-out

(j) fixed CPVs (k) generalists (l) P18 leave-one-out

(m) fixed CPVs (n) generalists (o) P19 leave-one-out

(p) EGO generalists (q) DE_{LHS} generalists

Figure 4.10: Comparison of EGO and DE on COCO problems P15 to P19 according to mean solution error ($\bar{\epsilon}$) achieved at various OFE budgets (β). Shaded regions indicate β where Mann-Whitney U-tests failed to show that the difference in means is statistically significant given a confidence level of 95%.

majority of the COCO problems which were poorly suited to ordinary Kriging modeling, with an exception for P12 were DE_{LHS} performed better for OFE budgets higher then 100. Given that benchmarking via tunability produced better agreement with the leave-one-out tests compared to benchmarking via fixed CPVs, benchmarking via tunability is judged to be more accurate.

Regarding the comparison of EGO and DE_{LHS}, the benchmarking via tunability results are viewed as accurate, albeit with a low level of confidence. For a higher confidence level, more comprehensive computational experiments could be conducted. These computational experiments should consider the effects of varying other EGO Kriging parameters, in addition to the EGO N_0 parameter. This case study also brings to the fore a significant grey area when it comes to benchmarking via tunability. Specifically, what CPVs of each algorithm should be tuned? Benchmarkers need to use their discretion in this regard, as tuning insignificant CPVs while ignoring significant CPVs, is likely to produce misleading results. For this reason, depending on the level of certainty desired for the numerical experiments, multiple tuning formulations should be considered. In this case study, DE_{LHS} was tuned according to N, C_r and F, while DE_{LHS_N} was only tuned according to N while C_r and F were fixed. Given the low γ budget for tunability, DE_{LHS} and DE_{LHS_N} produced mostly equivalent results, adding confidence to the conclusion that tuning EGO according to only N_0 is sufficient. However for higher γ budgets, DE_{LHS} is expected to begin to outperform DE_{LHS_N}, in which case additional EGO CPVs should also be tuned.

4.5 Discussion

The case studies conducted illustrate many of the advantages and disadvantages of benchmarking via tunability compared to that of benchmarking using fixed CPVs. Benchmarking via tunability entails tuning each algorithm being compared, and is therefore more computationally expensive compared to benchmarking via fixed CPVs on problem per problem basis. Although the degree of extra computational expense can be reduced through the use of efficient tuning algorithms, as was done in the case studies, the difference is still notable with the lowest degree of extra computational expense in the conducted case studies, being a 20 fold increase. This computational disadvantage argument holds provided that no computational resources were used in control parameter studies as to determine the CPVs used in the fixed CPV comparison.

Benchmarking via fixed CPVs has a disadvantage in the lack of incorporation of the sensitivity of an algorithm's performance to its CPVs. Therefore the benchmarker needs to demonstrate that any performance difference observed is not an artifact of the choice of CPVs used, and is representative of the algorithms themselves. An argument often used to bypass the question of the appropriateness of CPVs used, is to compare algorithms according to default CPVs, since it can be argued that these are the CPVs which will be used in practice. However, given the increasing availability and use of automated algorithm configuration techniques, more practitioners are using CPVs shown to work well on testing problems, which they believe to be representative of the problem at hand. Comparison according to the performance achieved using default CPV recommendations is problematic for another reason. Common practice for presenting an new optimization algorithm, is to gauge that algorithm's performance using the

default CPVs on a selected problem testbed according to selected performance criteria. Problematically however, the test suite and performance criteria which are used, often differ. Finally, even if algorithms are compared whose default CPVs have been shown to be effective on the same testing problems using the same performance metrics, those default CPVs are not necessarily appropriate for the problems in a different benchmark testbed. For these reasons, the choice of CPVs to use in fixed CPV benchmarking is of major significance, and using default CPVs is deemed inappropriate if algorithms are to be compared in a holistic manner.

Benchmarking via tunability by incorporating the effects of CPVs into the benchmarking process mitigates these issues, provided certain pitfalls are avoided. Firstly, care should be taken that the tuning formulations for each algorithm compared are appropriately constructed. CPVs which are identified as influential in an algorithm's documentation should be tuned, as those are the CPVs the *normal* automated algorithm configuration practitioner would adjust. Secondly, the effect of the tuning algorithm on the tunability results needs to be considered, as the tuning algorithm used may be better suited for one algorithm's tuning formulation than another algorithm's tuning formulation. Thirdly, when stochastic tuning algorithms are used, checks need to be made that the tunability results are consistent from one tuning run to another. Ultimately however, benchmarking via tunability is a numerical process, and therefore the results obtained should be scrutinized accordingly.

Conclusions from the benchmarking via tunability's chapter together with those from the tMOPSO and MOTA chapters, follow next, in the conclusion to this thesis.

CHAPTER 5

CONCLUSION

The tMOPSO and MOTA tuning algorithms, together with an investigation into benchmarking via tunability have been presented. Each of these contributions was presented in its own chapter, together with supporting numerical experiments. Conclusions for each chapter, a general conclusion and recommendations for future research, follow.

In Chapter 2, the tMOPSO algorithm is proposed for tuning stochastic optimization algorithms under multiple OFE budgets. Central to the proposed algorithm is the direct incorporation of CPV sensitivity to OFE budgets into the tuning problem formulation through the of use of a multi-objective optimization. tMOPSO is specialized for tuning stochastic algorithms through the use of a noise-handling strategy which uses MWUTs to pre-emptively terminate the resampling process and thereby boost tuning efficiency. Furthermore, tMOPSO utilizes the historical information from optimization runs used to assess the performance resulting from a specified CPV tuple for a given OFE budget, as to quantify that CPV tuple's performance at OFE budgets lower than the specified OFE budget. To efficiently process this information, fast Pareto dominance checking and Pareto dominance likelihood checking procedures are used. Conducted numerical experiments verify that tMOPSO is effective at tuning optimization algorithms. Specifically, for the tuning problems used and when compared under even tuning, tMOPSO was found to be better than or at least comparable to existing multiple OFE budget tuning algorithms. Furthermore, the numerical experiments conducted also indicate that tuning an optimization algorithm under multiple OFE budgets using tMOPSO should be more effective compared to setting up multiple uncoupled tuning problems each of which is focused on a different single OFE budget.

In Chapter 3, the MOTA algorithm is proposed for tuning stochastic optimization algorithms according to multiple utility measures under multiple OFE budgets. MOTA uses many objective optimization to achieve this end, with an objective for each utility measure and an extra speed objective as to tune under multiple OFE budgets. Decomposition is used to solve the resulting many objective optimization problem, with the original problem being broken up into multiple bi-objective subproblems. Similarly to tMOPSO, MOTA utilizes the history information from CPV assessment runs, so that one CPV tuple assessment run is used to gauge

utility at multiple OFE budgets. The bi-objective decomposition scheme used by MOTA is well-aligned to making use of this history information. Numerical experiments were conducted to gauge MOTA's performance. For the specialist tuning problems, MOTA is effective at determining specialist CPV tuples over a range of OFE budgets, having a comparable performance to the tMOPSO algorithm. For the many objective generalist tuning problems, MOTA outperformed the tMOPSO and RAND^M tuning algorithms. This superior performance is attributed to MOTA being designed from the ground up as a many objective tuning algorithm.

In Chapter 4, work regarding benchmarking via tunability is presented. The idea behind benchmarking via tunability is that if a numerical method is well-suited to a problem, then it should be easy to determine parameter settings which result in good performance. Accordingly, benchmarking via tunability compares optimization algorithms according to the difficulty of selecting CPVs which result in favorable performance. In order to compare benchmarking using tunability against the traditional approach of benchmarking via fixed CPVs, case studies were conducted to compare various algorithms over multiple OFE budgets. For the conducted case studies, benchmarking via tunabily produced better results compared to comparison using fixed CPVs. Benchmarking via tunablity is a useful tool for comparing optimization algorithms in a holistic manner, where the different CPV choices available to the algorithm user are included into the benchmarking process. By incorporating the CPV choices available to an optimization practitioner into the benchmarking process, benchmarking via tunability offers a significant paradigm shift compared to benchmarking according to fixed CPVs. This paradigm shift favors the increasing number of practitioners which employ automated algorithm configuration during the process of numerical optimization.

Tuning optimization algorithms under multiple OFE budgets is the unifying theme of this thesis. The tMOPSO algorithm is developed for tuning an optimization algorithm to a single problem under multiple OFE budgets. MOTA expands upon tMOPSO as to conduct many-objective tuning over multiple OFE budgets. Finally, multiple OFE budget tuning is used in the case studies conducted to investigate benchmarking optimization algorithms according to their tunability. Benchmarking via tunability could be done using a single OFE budget tuning algorithm, however the results obtained would be limited to a single OFE budget. Benchmarking via tunability, using a multiple OFE budget tuning algorithm as to gauge algorithm performance over a range of OFE budgets is more useful, since the number of OFEs available to an optimization practioner varies widely.

Avenues for future research are available for both tuning under multiple OFE budgets, and for benchmarking via tunability. In particular, MOTA's effectiveness at tuning multi-objective optimization algorithms to multiple performance measures at multiple OFE budgets needs to be gauged. Further investigation could also be done with regard to utilizing CPV trends to enhance performance when determining specialist CPVs for a problem suite under multiple OFE budgets. Lastly, benchmarking via tunability has been conceptually presented and demonstrated on select case studies. Conducting additional case studies, together with mathematical formalization of the concepts behind benchmarking via tunability, should further contribute to the field of numerical optimization.

For success at numerical optimization, practitioners need to select an optimization algorithm

and CPVs for that algorithm which are appropriate for the problem they are engaged with. Accordingly, an optimization algorithm and CPVs need to be selected which are effective for the numerical characteristics of the objective function, constraints imposed, and termination criteria of the problem at hand. To aid practitioners in this task, two new tuning algorithms named tMOPSO and MOTA are presented, and benchmarking via tunability is discussed. Specifically, tMOPSO and MOTA are tuning algorithms for determining CPVs which are effective on representative testing problems, while benchmarking via tunability aims to inform a practitioner of which optimization algorithm should be tuned to those problems. Consequently, tMOPSO, MOTA, and benchmarking via tunability, can be used to assist the process of numerical optimization, thereby benefiting the many design processes which rely on numerical optimization to create an effective product.

BIBLIOGRAPHY

Auger, A. and Hansen, N. (2005). Performance evaluation of an advanced local search evolutionary algorithm. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1777 – 1784 Vol. 2.

Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. *Hybrid Metaheuristics*, pages 108–122.

Bartz-Beielstein, T., Lasarczyk, C., and Preuss, M. (2005). Sequential parameter optimization. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 773–780. IEEE.

Bartz-Beielstein, T., Parsopoulos, K., and Vrahatis, M. (2004). Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis & Computational Mathematics*, 1(2):413–433.

Berry, A. and Vamplew, P. (2006). An efficient approach to unbounded bi-objective archives: Introducing the mak_tree algorithm. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, page 626. ACM.

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Beyer, H. (2000). Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):239–267.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Citeseer.

Branke, J. and Elomari, J. (2012). Meta-optimization for parameter tuning with a flexible computing budget. In *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference*, pages 1245–1252. ACM.

Brockhoff, D. and Zitzler, E. (2009). Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary Computation*, 17(2):135–166.

Bui, L., Abbass, H., and Essam, D. (2009). Localization for solving noisy multi-objective optimization problems. *Evolutionary Computation*, 17(3):379–409.

Clerc, M. and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

Coello, C., Pulido, G., and Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279.

Conover, W. (1999). *Practical Nonparametric Statistics*. John Wiley & Sons, third edition.

Das, S. and Suganthan, P. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, (99):1–28.

Deb, K. and Agrawal, R. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, 9:1–34.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pages 105–145. Springer London.

den Besten, M. L. (2004). *Simple Metaheuristics for Scheduling: An empirical investigation into the application of iterated local search to deterministic scheduling problems with tardiness penalties*. PhD thesis.

Di Pierro, F., Khu, S.-T., and Savić, D. A. (2007). An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45.

Dolan, E. and Moré, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Dréo, J. (2008). Multi-criteria meta-parameter tuning for mono-objective stochastic metaheuristics. In *2nd International Conference on Metaheuristics and Nature Inspired Computing*.

Dréo, J. (2009). Using performance fronts for parameter setting of stochastic metaheuristics. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation: Late Breaking Papers*, pages 2197–2200. ACM.

Durillo, J. J. and Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771.

Dymond, A.S.D., Engelbrecht, A.P., and Heyns, P.S. (2011). The sensitivity of single objective optimization algorithm control parameter values under different computational constraints. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1412–1419. IEEE.

Dymond, A.S.D., Engelbrecht, A.P., Kok., S., and Heyns, P.S. (2014). Tuning optimization algorithms under multiple objective function evaluation budgets. *IEEE Transactions on Evolutionary Computation*, Volume Still not known(X):Preprint should be available from June 2014.

Dymond, A.S.D., Kok., S., and Heyns, P.S. (2013). The sensitivity of multi-objective optimization algorithm performance to objective function evaluation budgets. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1868–1875. IEEE.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95*, pages 39–43.

Eiben, A., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141.

Engelbrecht, A.P. (2007). *Computational Intelligence: An Introduction*. Wiley.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128.

Handl, J., Lovell, S. C., and Knowles, J. (2008). Multiobjectivization by decomposition of scalar cost functions. In *Parallel Problem Solving from Nature–PPSN X*, pages 31–40. Springer.

Hansen, N., Auger, A., Finck, S., Ros, R., et al. (2010). Real-parameter black-box optimization benchmarking 2010: Experimental setup. *INRIA*.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

Hutter, F., Hoos, H., Leyton-Brown, K., and Murphy, K. (2009a). An experimental investigation of model-based parameter optimisation: SPO and beyond. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 271–278. ACM.

Hutter, F., Hoos, H., Leyton-Brown, K., and Stützle, T. (2009b). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. *IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004*.

López-Ibáñez, M. and Stützle, T. (2011). Automatic configuration of multi-objective ACO algorithms. *Swarm Intelligence*, pages 95–106.

López-Ibáñez, M. and Stützle, T. (2012). The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861 –875.

Malan, K. and Engelbrecht, A. (2009). Quantifying ruggedness of continuous landscapes using entropy. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1440–1447. IEEE.

Maron, O. and Moore, A. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1):193–225.

Moré, J. and Wild, S. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191.

Mostaghim, S. and Teich, J. (2005). Quad-trees: A data structure for storing pareto sets in multiobjective evolutionary algorithms with elitism. *Evolutionary Multiobjective Optimization*, pages 81–104.

Nannen, V. and Eiben, A. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 975–980. Morgan Kaufmann Publishers Inc.

Pedersen, M. (2010). *Tuning & Simplifying Heuristical Optimization*. PhD thesis, University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group.

Pedersen, M. and Chipperfield, A. (2008). Parameter tuning versus adaptation: proof of principle study on differential evolution.

Pošík, P., Huyer, W., and Pál, L. (2012). A comparison of global search algorithms for continuous black box optimization. *Evolutionary Computation*, pages 1–32.

Radulescu, A., López-Ibáñez, M., and Stützle, T. (2013). Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 825–840. Springer Berlin Heidelberg.

Roustant, O., Ginsbourger, D., Deville, Y., et al. (2012). DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51.

Salehinejad, H., Rahnamayan, S., Tizhoosh, H., and Chen, S. (2014). Micro-differential evolution with vectorized random mutation factor. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE.

Saxena, D. K., Duro, J. A., Tiwari, A., Deb, K., and Zhang, Q. (2013). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, pages 77–99.

Shi, Y. and Eberhart, R. (1998). Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600. Springer.

Sierra, M. and Coello, C. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Evolutionary Multi-Criterion Optimization*, pages 505–519. Springer.

Smit, S. K. and Eiben, A. E. (2009). Comparing parameter tuning methods for evolutionary algorithms. In *IEEE Congress on Evolutionary Computation*, pages 399–406. IEEE.

Smit, S. K. and Eiben, A. E. (2010a). Beating the 'world champion' evolutionary algorithm via REVAC tuning. In *2010 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

Smit, S. K. and Eiben, A. E. (2010b). Parameter tuning of evolutionary algorithms: Generalist vs. specialist. *Applications of Evolutionary Computation*, pages 542–551.

Smit, S. K., Eiben, A. E., and Szlávik, Z. (2010). An MOEA-based method to tune EA parameters on multiple objective functions. In *IJCCI (ICEC)*, pages 261–268. Citeseer.

Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Report*.

Wagner, T. and Wessing, S. (2012). On the effect of response transformations in sequential parameter optimization. *Evolutionary Computation*.

Wang, Y., Cai, Z., and Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55 –66.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Xi, Z., Hu, C., and Youn, B. D. (2012). A comparative study of probability estimation methods for reliability analysis. *Structural and Multidisciplinary Optimization*, 45(1):33–52.

Yuan, Z., de Oca, M., Birattari, M., and Stützle, T. (2011). Modern continuous optimization algorithms for tuning real and integer algorithm parameters. *Swarm Intelligence*, pages 203–214.

Zhang, J. and Sanderson, A. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhang, Q., Liu, W., Tsang, E., and Virginas, B. (2010). Expensive multiobjective optimization by MOEA/D with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474.

Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., and Tiwari, S. (2008). Multiobjective optimization test instances for the CEC 2009 special session and competition. *University of Essex and Nanyang Technological University, Technical Report. CES-487*.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195.

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. In *EUROGEN*, pages 95–100. Citeseer.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117 – 132.