Tuning Optimization Algorithms under Multiple Objective Function Evaluation Budgets

Antoine S. Dymond, Andries P. Engelbrecht, Schalk Kok, P. Stephan Heyns

July 2, 2015

Abstract

Most sensitivity analysis studies of optimization algorithm control parameters are restricted to a single objective function evaluation (OFE) budget. This restriction is problematic because the optimality of control parameter values is dependent not only on the problem's fitness landscape, but also on the OFE budget available to explore that landscape. Therefore the OFE budget needs to be taken into consideration when performing control parameter tuning. This article 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. Conducted numerical experiments provide evidence that tMOPSO is effective at tuning under multiple OFE budget constraints.

1 Introduction

Optimization practitioners often refer to parameter tuning studies when selecting an algorithm's control parameter values (CPVs). These studies are usually restricted to a single objective function evaluation (OFE) budget constraint. This is problematic since the performance of optimization algorithms is dependent not only on the problem's fitness landscape [1, 2], 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 by Dymond *et al.* [3]. For the sensitivity investigation by Dymond *et al.* 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 by Dymond *et al.* 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 the sensitivity investigation by Dymond *et al.* 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.

The outline of this paper is as follows: related work is discussed in Section 2, after which the proposed tuning algorithm is described in Section 3. Then the numerical setup used to gauge the effectiveness of tMOPSO is given in Section 4, followed by the numerical results in Section 5.

2 Related work

Control parameter tuning entails finding which algorithm's CPVs are optimal according to a specified utility metric [4]. 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 [5], where:

- the application layer refers to the problem instance(s) used in CPV 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.

One of the first examples of control parameter tuning applied to evolutionary algorithms was using a genetic algorithm to improve another genetic algorithm's performance on five testing problems [6]. Since then, many other control parameter tuning algorithms have been proposed [7, 8, 9, 10, 11, 12, 13], and numerous other tuning studies have been performed [14, 15, 16, 17]. The M-FETA algorithm [13], 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 [18] 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) [9] and the Sequential Parameter Optimization (SPO) [8] 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 [19]. 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 [20] 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 [21]. The Flexible Budget method [21] 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 [22] was proposed in Dréo [23]. Dréo [23] 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 [24]. The Dréo [23] proof-of-concept algorithm, although similar to, is not a multiple OFE budget tuning algorithm. Specifically, the Dréo [23] 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 [23] 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 paper 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 [23] 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 [25] 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:

$$\textit{minimize } \mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_{n_f}(\mathbf{x}) \end{bmatrix} \tag{1}$$

$$\textit{subject to } g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, n_g \tag{2}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \dots, n_h \tag{3}$$

$$x_k \in [b_k^L, b_k^U] \quad k = 1, \dots, n, \tag{4}$$

where $f_1, f_2, \dots, f_{n_f}$ are the conflicting sub-objectives, n 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:

$$f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2), \forall\, k \in 1, 2, \dots, n_f, \tag{5}$$

and

$$\exists k \in 1, 2, \dots, n_f : f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2). \tag{6}$$

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.

3 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:

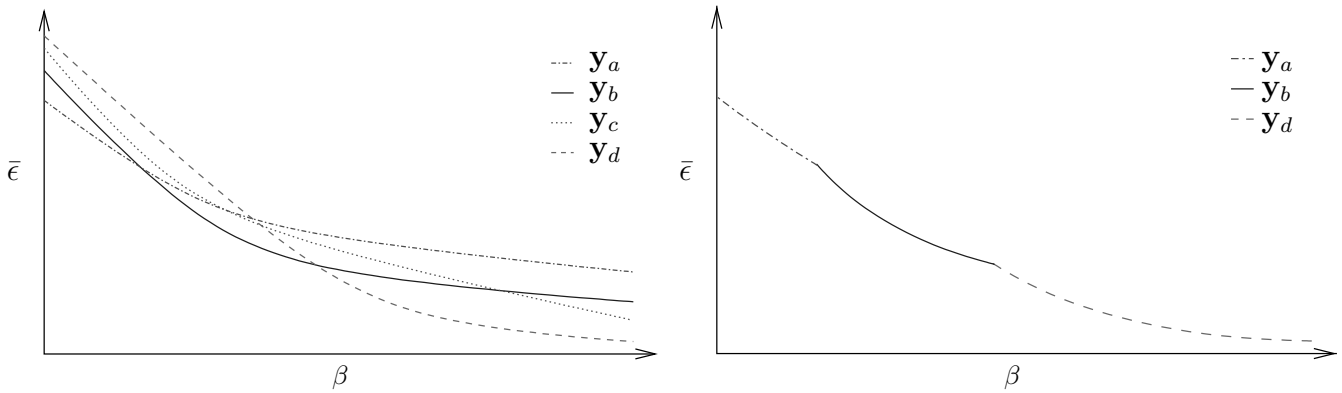
1. the mean solution error or mean cost function value obtained after a fixed number of OFEs, and
2. the mean number of OFEs required to solve a problem within a specified solution error tolerance.

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 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 [22] 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:

$$\mathbf{u}(\mathbf{y}, \beta) = \begin{bmatrix} \bar{\epsilon}(\mathbf{y}, \beta) \\ \beta \end{bmatrix}, \tag{7}$$

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,



(a) The mean solution error obtained $\bar{\epsilon}$ versus the OFE budget available β, for four CPV tuples namely, $\mathbf{y}_a$, $\mathbf{y}_b$, $\mathbf{y}_c$ and $\mathbf{y}_d$.

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

Figure 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.

as illustrated in Figure 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:

$$\begin{aligned} &\text{minimize} && \mathbf{u}(\mathbf{y}, \beta) \\ &\text{subject to} && 0 < \beta \leq \beta_{\max} \\ & && g_i(\mathbf{y}) \leq 0, \quad i = 1, \ldots, n_g, \end{aligned} \tag{8}$$

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 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.

3.1 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 OFE budget increases, as illustrated in Figure 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 [26]. Also, although there are established multi-objective algorithms such as methods based on local models [26], 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 [27] 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

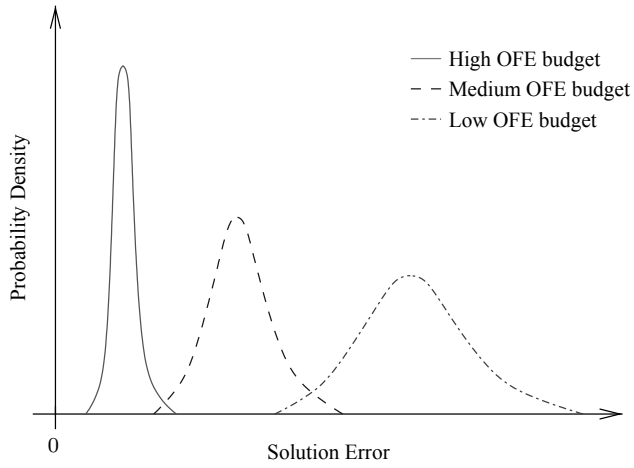


Figure 2: Illustration of the expected decrease in solution error mean and variance as the available OFE budget increases, for CPVs close to the Pareto-optimal front.

the noise strength by a factor of $\sqrt{n_s}$ [26], 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 a racing [28] or racing equivalent [29] method in order to achieve pre-emptively terminating resampling.

tMOPSO employs a pre-emptively terminating resampling strategy which uses the Mann-Whitney U test (MWUT) [20]. 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:

$$\tilde{F}(\mathbf{x}_p)_k \leq_\alpha \tilde{F}(\mathbf{x}_c)_k \qquad \forall k \in 1, 2, \ldots, n_f, \tag{9}$$

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:

$$\beta_2 \leq \beta_1 \tag{10}$$

and

$$\tilde{\epsilon}(\mathbf{y}_2, \beta_2) \leq_\alpha \tilde{\epsilon}(\mathbf{y}_1, \beta_1), \tag{11}$$

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.

3.2 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:

$$\mathbf{u}(\mathbf{y}, N) \rightarrow \mathbf{u}(\mathbf{y}, 2N) \rightarrow \cdots \rightarrow \mathbf{u}(\mathbf{y}, \beta - N). \tag{12}$$

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:

$$\mathbf{u}(\mathbf{y}, b_i) \, \forall \, b_i \in B : b_i \leq \beta' \tag{13}$$

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 (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 (10)-(11). Given a set of candidate groups of CPVs Y, the utility values from (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:

$$\beta' = \min(\lambda \cdot \beta, \, \beta_{max}) \tag{14}$$

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 (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 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.

3.3 Quickly 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 [30], 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,

$$\text{add } \mathbf{x}_c \text{ to } A \text{ if } \nexists\, \mathbf{x} \prec \mathbf{x}_c \,\forall \mathbf{x} \in A, \tag{15}$$

$$\text{discard from } A \text{ all } \mathbf{x} \text{ where } \mathbf{x}_c \prec \mathbf{x}, \tag{16}$$

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 [31]. 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,

$$\mathbf{x}_c \not\prec A \text{ if } (\mathbf{x}_c \not\prec \mathbf{x}_n), \tag{17}$$

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

$$f_2(\mathbf{x}_n) = \max\big\{ f_2(\mathbf{x}_c) - f_2(\mathbf{x}) \,\forall\, \mathbf{x} \in A \\ : f_2(\mathbf{x}_c) - f_2(\mathbf{x}) \leq 0 \big\}. \tag{18}$$

The approach of Berry and Vamplew [31] 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 (17)-(18) due to the noise characteristics of tMOPSO's utility measure. In particular, since the second objective

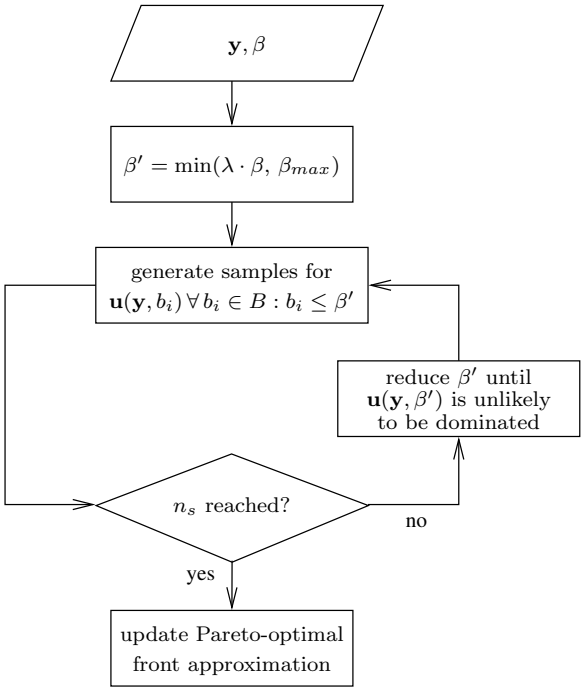


Figure 3: Flow chart of tMOPSO's CPV tuple assessment procedure.

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 (10)-(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.

3.4 Tuning Optimization Algorithm

The tuning multi-objective particle swarm optimization (tMOPSO) algorithm is a particle swarm based [25] algorithm. A number of multi-objective PSO variants have been presented [32, 33, 25]. tMOPSO is, however, the first variant specialized for tuning single objective optimization algorithms 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 (8), tMOPSO uses a decision vector of the following form:

$$\mathbf{x} = \begin{bmatrix} \ln \beta \\ y_1 \\ y_2 \\ \vdots \\ y_{n_y} \end{bmatrix}, \tag{19}$$

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.

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:

$$\mathbf{x}_0^j = \mathbf{I}^L + \mathbf{r}() \circ (\mathbf{I}^U - \mathbf{I}^L), \tag{20}$$

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:

$$\mathbf{x}_i^j = \mathbf{x}_{i-1}^j + \mathbf{v}_i^j, \tag{21}$$

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:

$$\begin{aligned} \mathbf{v}_{i+1}^j = \omega \mathbf{v}_i^j &+ c_p\, \mathbf{r}() \circ \left(\mathbf{x}_p^j - \mathbf{x}_i^j\right) \\ &+ c_g\, \mathbf{r}() \circ \left(\mathbf{x}_g - \mathbf{x}_i^j\right), \end{aligned} \tag{22}$$

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.

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 $x_{i,1}^j$ and velocity $v_{i,1}^j$ in the assessment OFE budget dimension as follows:

$$\ln \beta_g = x_{i,1}^j + \omega\, v_{i,1}^j + c_\beta\, r_g()\, \ln \beta_{max} \tag{23}$$

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

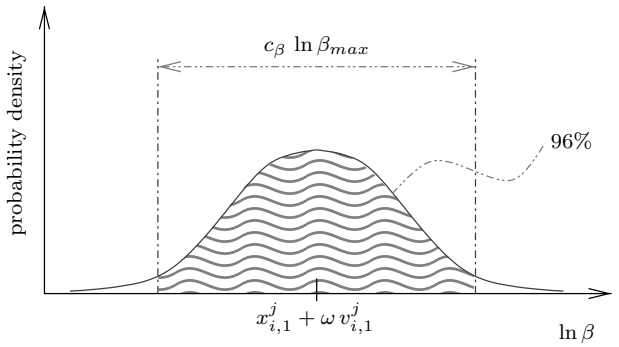


Figure 4: Probability density of the OFE budget value used to select the guiding decision vectors in tMOPSO's velocity update rule.

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 4 shows the probability density function which is used to generate β_g.

Each particle's velocity is updated as follows:

$$\begin{aligned}\mathbf{v}^j_{i+1} = \omega \mathbf{v}^j_i &+ c_p\, \mathbf{r}() \circ \left(\mathbf{x}^j_p() - \mathbf{x}^j_i\right) \\ &+ c_g\, \mathbf{r}() \circ \left(\mathbf{x}_g() - \mathbf{x}^j_i\right) + \beta_c,\end{aligned} \tag{24}$$

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:

$$-0.5(c_p + c_g)\, \omega\, v^j_{i,1}. \tag{25}$$

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 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 5.

4 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

```
procedure tMOPSO
    Generate particles initial positions                          ▷ (20)
    assess generated CPV tuples                                   ▷ Subsection 3.1
    update local PF approximations
    i ← 1
    while γ < γ_max do                                            ▷ main loop
        for j ∈ {1, 2, ..., N} do
            repeat
                v_i^j                                             ▷ (24)
                x_i^j                                             ▷ (21)
            until x_i^j valid
        end for
        assess generated CPV tuples                               ▷ Subsection 3.1
        update local PF approximations
        i ← i + 1
    end while
end procedure
```

Figure 5: tMOPSO pseudocode

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.

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.

4.1 Application Layers

Selected optimization algorithms are tuned to problems from the CEC 2005 special session on real-parameter optimization [34]. 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 [35]. 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

$$\hat{\epsilon} = \hat{w} \cdot \epsilon \tag{26}$$

The value for $\hat{w}$ was approximated numerically 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.

4.2 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) optimization algorithm. A brief description of each algorithm, together with information on which control parameters are tuned, follow.

4.2.1 DE

Differential evolution was developed to optimize non-differentiable, non-linear cost functions which are multi-modal [36, 37]. The *rand/1/bin* [36] variation of DE, using the bound constraint handling mechanism proposed by [38], 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 [35], 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:

$$5 \leq N \tag{27}$$
$$0 \leq C_r \leq 1 \tag{28}$$
$$0 \leq F. \tag{29}$$

4.2.2 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 [38]. 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 [39, 40]. The PSO tuning is constrained as follows:

$$5 \leq N \tag{30}$$
$$0 \leq c_p \tag{31}$$
$$0 \leq c_g \tag{32}$$
$$0 \leq \omega. \tag{33}$$

Additional constraints such as $\omega \leq 1$ and $c_p + c_g \leq 4$ [40] are omitted, since these common recommendations may be detrimental for low OFE budgets were swarm explosion may be beneficial.

4.2.3 CMA-ES

The covariance matrix adaptation evolutionary strategy [41] was developed to handle badly scaled quadratic problems and is invariant against linear transformations of the search space [42]. 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

$$5 \leq N \tag{34}$$
$$1 \leq \lfloor N \cdot \mu_f \rfloor \tag{35}$$
$$\mu_f \leq 1 \tag{36}$$
$$0.01 \leq \sigma_r. \tag{37}$$

4.3 Tuning Algorithms Compared

The multiple OFE budget algorithms compared are tMOPSO, two tMOPSO variants and the Flexible Budget method (FBM) [21]. 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 [23] 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) [43] 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 [44] (HV) 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.

4.3.1 tMOPSO and its variants

The CPV tuple candidates for the CPV sweeps for tMOPSO and its variants were chosen according to selected PSO literature [32, 39, 40, 29]. 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].

4.3.2 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 [21]. 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 [21], the implemented FBM uses area under the curve for tie-breaking. At each generation, size 2 tournaments are conducted, one

[1] https://pypi.python.org/pypi/optTune/

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 B control parameters. Additionally, in accordance with the tuning problem formulation in (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.

4.3.3 REVAC

The relevance estimation and value calibration method is a single objective tuning algorithm which uses Shannon entropy models to guide the tuning process [9]. 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 [9] 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.

4.3.4 SPO

The sequential parameter optimization framework uses surrogate modeling to tune an optimization algorithm [8]. Numerous surrogate modeling options and initialization strategies are available. In these numerical experiments, SPO is set up to use Kriging Gaussian models [45] 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 [12], 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.

4.3.5 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,

[2] http://cran.r-project.org/web/packages/SPOT/

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,

$$I_\sigma = 0.5\,(\mathbf{b}^U - \mathbf{b}^L)\,e^{\zeta \ln I_r}, \tag{38}$$

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%.

5 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. For brevity, the statistical significance tests together with the complete tMOPSO tuning results are not included in this paper. These tests and results are available at http://ieeexplore.ieee.org.

5.1 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 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.

[3]http://cran.r-project.org/web/packages/irace/

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

		mean HV found ($\times 10^3$) for $\gamma = 3 \times 10^7$ by				mean HV found ($\times 10^3$) for $\gamma = 15 \times 10^7$ by			
algorithm	CEC problem	tMOPSO	tMOPSO$^-$	tMOPSO$^=$	FBM	tMOPSO	tMOPSO$^-$	tMOPSO$^=$	FBM
DE	3	**29.870**	29.783	29.714	29.666	**29.885**	29.878	29.834	29.856
	5	**28.530**	28.337	28.076	28.084	**28.639**	28.576	28.449	28.527
	6	**29.947**	29.855	29.936	29.910	**29.949**	29.877	29.944	29.942
	8	**0.991**	0.990	0.974	0.987	**1.001**	*1.000*†	0.988	0.998
	10	**27.937**	27.557	26.927	27.073	**28.204**	28.124	27.551	27.997
PSO	3	**29.871**	29.844	29.806	29.816	**29.891**	29.885	29.851	29.874
	5	**27.599**	27.413	27.122	27.165	**27.795**	27.712	27.441	27.605
	6	**29.953**	29.947	29.944	29.930	**29.955**	29.954	29.950	29.950
	8	**1.867**	1.686	1.479	1.487	**1.968**	1.953	1.736	1.863
	10	**27.789**	27.616	27.281	27.476	**27.974**	27.892	27.626	27.847
CMA-ES	3	**29.906**	29.894	29.885	29.859	**29.914**	29.911	29.904	29.902
	5	**29.563**	29.535	29.471	29.497	**29.602**	29.589	29.550	29.592
	6	**29.945**	29.939	29.942	29.905	**29.948**	29.947	29.944	29.942
	8	*0.990*	**0.997**	0.975	*0.988*	*1.008*	**1.011**	0.989	1.000
	10	**29.262**	29.223	29.045	29.144	**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.

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.

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. 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 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.

5.2 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 parameter 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 3.

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 4. Furthermore, for 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%. 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 4 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 5, 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 5. For the remaining 10 data points, which

Table 3: 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	$S_k = 30$	$S_n = 8$	$S_o = 21$

Table 4: Comparison between tuning algorithms based on mean of the mean minimum solution error found for various application layer evaluation budgets (γ). Rankings are given in parenthesis next to the values.

		$-\log_{10}(\bar{\epsilon})$ for $\gamma = 3 \times 10^7$					$-\log_{10}(\bar{\epsilon})$ for $\gamma = 15 \times 10^7$				
algorithm	prob.	tMOPSO	tPSO	I/F-race	SPO	REVAC	tMOPSO	tPSO	I/F-race	SPO	REVAC
DE	3	3.350(2)	**3.440**(1)	*3.079*†(3)	2.726(4)	2.648(5)	3.604(3)	**3.636**(1)	3.612(2)	2.973(4)	2.927(5)
	5	1.585(3)	*1.644*(2)	**1.653**(1)	1.550(4)	1.498(5)	1.678(3)	**1.696**(1)	1.696(2)	1.641(4)	1.544(5)
	6	9.724(3)	**9.901**(1)	*9.892*(2)	8.661(5)	8.835(4)	9.902(3)	**9.997**(1)	9.969(2)	9.641(5)	9.788(4)
	8	*0.016*(5)	*0.016*(4)	*0.016*(2)	**0.016**(1)	*0.016*(3)	0.016(5)	0.016(4)	0.016(3)	0.017(2)	**0.020**(1)
	10	1.366(4)	*1.468*(2)	**1.479**(1)	1.426(3)	1.149(5)	1.491(3)	**1.524**(1)	1.513(2)	1.488(4)	1.391(5)
PSO	3	3.049(2)	**3.174**(1)	3.049(3)	3.010(4)	2.859(5)	3.322(2)	**3.351**(1)	3.302(3)	3.194(4)	3.144(5)
	5	1.210(4)	*1.250*(2)	**1.252**(1)	*1.243*(3)	1.184(5)	1.269(5)	*1.291*(4)	**1.297**(1)	*1.296*(2)	*1.292*(3)
	6	*9.172*(2)	**9.372**(1)	*7.428*(3)	6.651(5)	7.284(4)	9.649(2)	*9.614*(3)	**9.682**(1)	8.632(5)	8.752(4)
	8	*0.031*(2)	*0.031*(3)	*0.029*(4)	**0.032**(1)	0.027(5)	0.033(2)	**0.033**(1)	0.033(3)	0.032(5)	0.032(4)
	10	1.238(4)	*1.286*(2)	**1.288**(1)	1.244(3)	1.191(5)	1.292(3)	**1.324**(1)	1.311(2)	1.285(4)	1.265(5)
CMA-ES	3	7.312(4)	**7.570**(1)	7.454(3)	7.459(2)	7.125(5)	7.672(3)	**7.792**(1)	7.712(2)	7.634(5)	7.654(4)
	5	5.263(4)	*5.620*(2)	5.523(3)	**5.631**(1)	5.008(5)	5.655(5)	**5.787**(1)	*5.770*(2)	5.747(3)	5.659(4)
	6	10.16(4)	*10.26*(2)	**10.27**(1)	*10.23*(3)	10.02(5)	10.33(5)	*10.43*(3)	*10.46*(2)	10.41(4)	**10.47**(1)
	8	0.015(5)	*0.016*(2)	0.016(4)	**0.017**(1)	*0.016*(3)	0.016(4)	0.019(3)	0.016(5)	0.020(2)	**0.026**(1)
	10	2.243(5)	*2.330*(2)	**2.349**(1)	2.274(3)	2.259(4)	2.367(3)	**2.403**(1)	*2.401*(2)	2.337(5)	2.342(4)
$\sum$ Ranks		53	28	33	43	68	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.

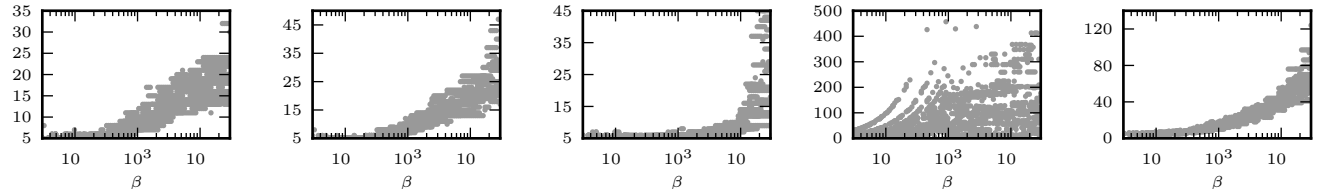
Table 5: Table showing what fraction extra γ tMOPSO requires to achieve a mean best solution error at β_{max} equal to that of the best single OFE budget tuning algorithms. The best single OFE budget algorithm which tMOPSO was compared to varies depending upon the tuning problem and the γ.

		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†	> 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.

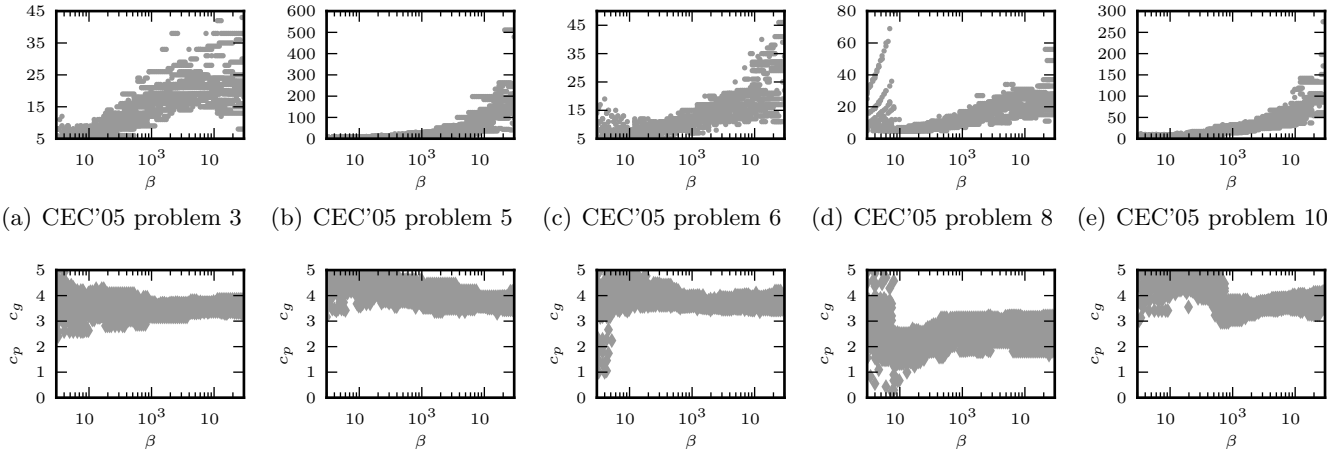
correspond to tuning DE to CEC'05 problem 8 and when tuning CMA-ES to CEC'05 problem 8, some of the tMOPSO runs got stuck in what could be viewed as a tuning local minimum, and cannot compete no matter how much extra γ is allocated. Increasing tMOPSO's population size is expected to reduce the risk of this occurring for these two Ackley based tuning problems. For 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



(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 6: 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 sub figure 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) CEC'05 problem 3 (g) CEC'05 problem 5 (h) CEC'05 problem 6 (i) CEC'05 problem 8 (j) CEC'05 problem 10

Figure 7: 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 sub figure shows the results for PSO tuned to a different CEC'05 optimization problem.

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.

5.3 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 [3], 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 [3].

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 6. 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 7, also recommend an increasing optimal swarm size as the OFE budget increases. Furthermore, in agreement with a literature recommendation [40], 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. However to be certain, an extensive sensitivity study such as in [46] 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 and Table 4. 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 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.

6 Conclusions

A new algorithm named tMOPSO is presented for tuning stochastic optimization algorithms under multiple OFE budgets. Central to the proposed algorithm is the use of a multi-objective tuning problem formulation which allows for CPV sensitivity to OFE budgets to be directly incorporated into the tuning problem. 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.

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 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.

Future work is the development of a many-objective tuning algorithm. 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. Development of such a tuning algorithm would require combining the core elements of tMOPSO with those of many objective optimization. This task will be complicated by the fact that many of the tMOPSO mechanics exploit the bi-objective nature of tMOPSO's problem formulation, in order to reduce computational overhead. Therefore, the design of a many objective tuning algorithm would require more than simply adding extra objectives to tMOPSO's problem formulation, and warrants further research.

Acknowledgments

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 authors and are not necessarily to be attributed to the NRF. 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.

References

[1] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] K. Malan and A. Engelbrecht, "Quantifying ruggedness of continuous landscapes using entropy," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2009, pp. 1440–1447.

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

[4] S. K. Smit and A. E. Eiben, "Parameter tuning of evolutionary algorithms: Generalist vs. specialist," *Applications of Evolutionary Computation*, pp. 542–551, 2010.

[5] ——, "Comparing parameter tuning methods for evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 399–406.

[6] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[7] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Citeseer, 2002, pp. 11–18.

[8] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss, "Sequential parameter optimization," in *The 2005 IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2005, pp. 773–780.

[9] V. Nannen and A. Eiben, "Relevance estimation and value calibration of evolutionary algorithm parameters," in *Proceedings of the 20th International Joint Conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., 2007, pp. 975–980.

[10] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, no. 1, pp. 267–306, 2009.

[11] F. Hutter, H. Hoos, K. Leyton-Brown, and K. Murphy, "An experimental investigation of model-based parameter optimisation: SPO and beyond," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2009, pp. 271–278.

[12] T. Wagner and S. Wessing, "On the effect of response transformations in sequential parameter optimization," *Evolutionary Computation*, 2012.

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

[14] T. Bartz-Beielstein, K. Parsopoulos, and M. Vrahatis, "Design and analysis of optimization algorithms using computational statistics," *Applied Numerical Analysis & Computational Mathematics*, vol. 1, no. 2, pp. 413–433, 2004.

[15] S. K. Smit and A. E. Eiben, "Beating the 'world champion' evolutionary algorithm via REVAC tuning," in *2010 IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[16] M. López-Ibáñez and T. Stützle, "Automatic configuration of multi-objective ACO algorithms," *Swarm Intelligence*, pp. 95–106, 2011.

[17] Z. Yuan, M. de Oca, M. Birattari, and T. Stützle, "Modern continuous optimization algorithms for tuning real and integer algorithm parameters," *Swarm Intelligence*, pp. 203–214, 2011.

[18] A. Radulescu, M. López-Ibáñez, and T. Stützle, "Automatically improving the anytime behaviour of multiobjective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization*. Springer Berlin Heidelberg, 2013, pp. 825–840.

[19] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement," *Hybrid Metaheuristics*, pp. 108–122, 2007.

[20] W. Conover, *Practical Nonparametric Statistics*, 3rd ed. John Wiley & Sons, 1999.

[21] J. Branke and J. Elomari, "Meta-optimization for parameter tuning with a flexible computing budget," in *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference*. ACM, 2012, pp. 1245–1252.

[22] J. Dréo, "Multi-criteria meta-parameter tuning for mono-objective stochastic metaheuristics," in *2nd International Conference on Metaheuristics and Nature Inspired Computing*, 2008.

[23] ——, "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*. ACM, 2009, pp. 2197–2200.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[25] A.P. Engelbrecht, *Computational Intelligence: An Introduction*. Wiley, 2007.

[26] L. Bui, H. Abbass, and D. Essam, "Localization for solving noisy multi-objective optimization problems," *Evolutionary Computation*, vol. 17, no. 3, pp. 379–409, 2009.

[27] H. Beyer, "Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 239–267, 2000.

[28] O. Maron and A. Moore, "The racing algorithm: Model selection for lazy learners," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 193–225, 1997.

[29] M. Pedersen, "Tuning & simplifying heuristical optimization," Ph.D. dissertation, University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group, 2010.

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

[31] A. Berry and P. Vamplew, "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*. ACM, 2006, p. 626.

[32] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[33] M. Sierra and C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and $\in$-dominance," in *Evolutionary Multi-Criterion Optimization*. Springer, 2005, pp. 505–519.

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

[35] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55 –66, 2011.

[36] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[37] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, no. 99, pp. 1–28, 2010.

[38] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[39] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*. Springer, 1998, pp. 591–600.

[40] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[41] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

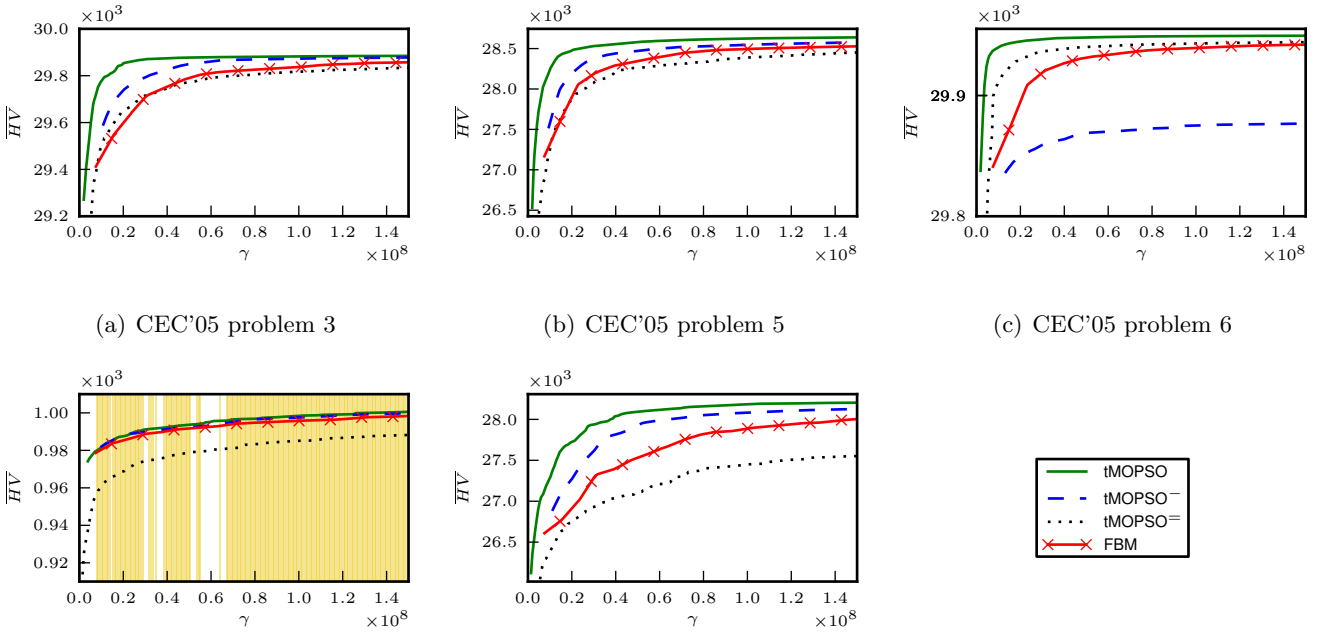
[42] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *The 2005 IEEE Congress on Evolutionary Computation*, vol. 2, 2005, pp. 1777 – 1784 Vol. 2.

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

[44] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117 – 132, 2003.

[45] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.

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



(a) CEC'05 problem 3

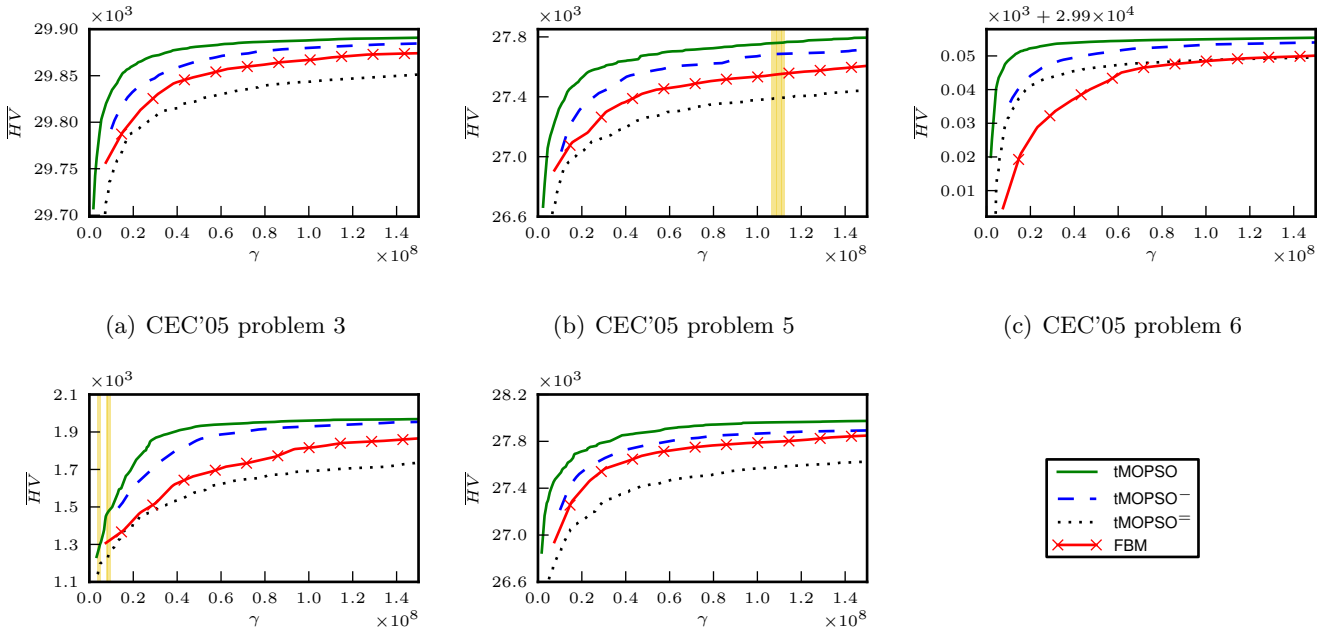
(b) CEC'05 problem 5

(c) CEC'05 problem 6

(d) CEC'05 problem 8

(e) CEC'05 problem 10

Figure 1: 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.



(a) CEC'05 problem 3

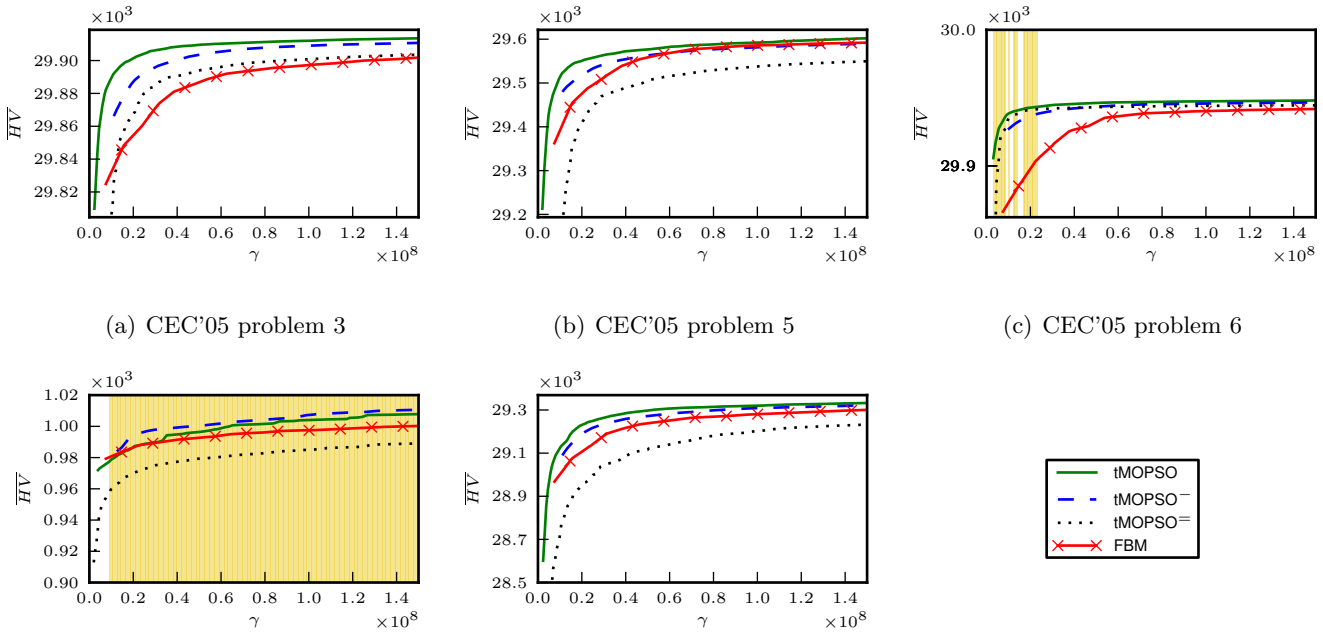
(b) CEC'05 problem 5

(c) CEC'05 problem 6

(d) CEC'05 problem 8

(e) CEC'05 problem 10

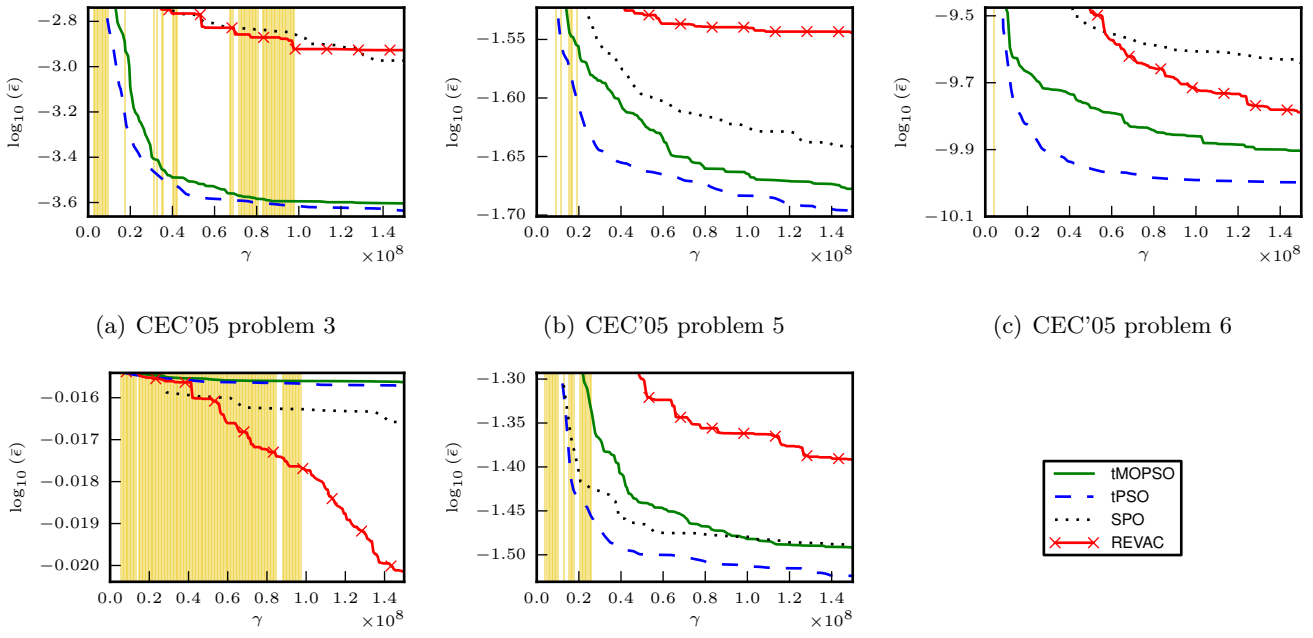
Figure 2: Mean hypervolume obtained ($\overline{HV}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the PSO tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the 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

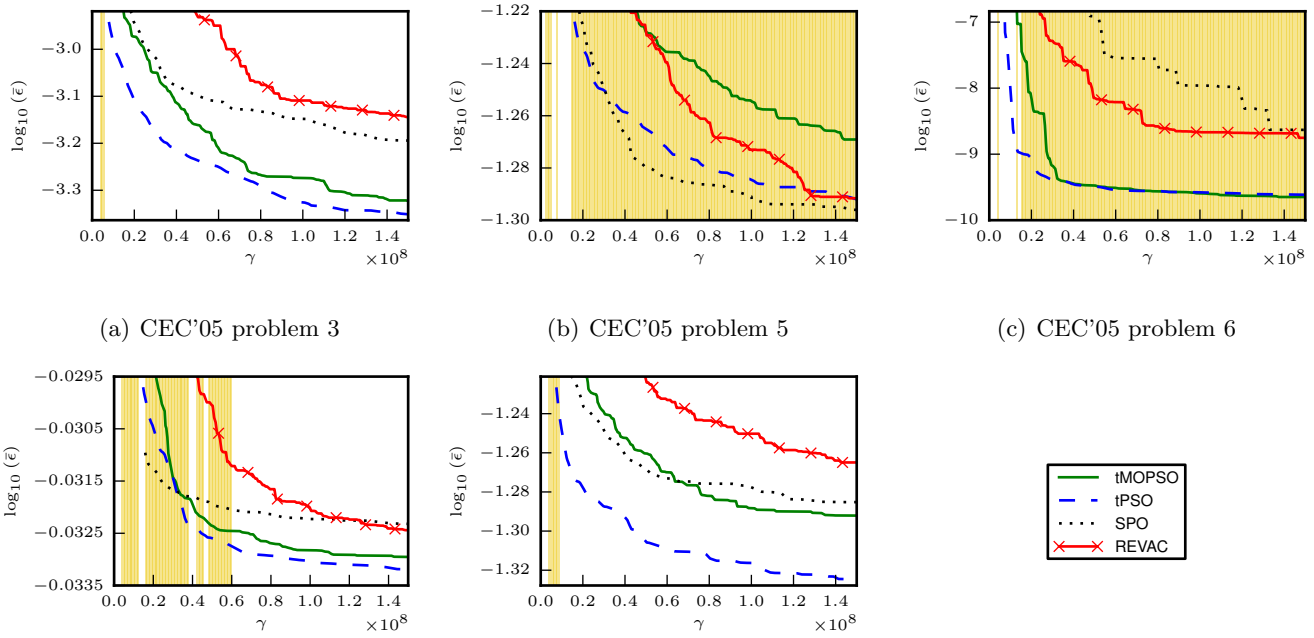
Figure 3: Mean hypervolume obtained ($\overline{HV}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the CMA-ES tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.



(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

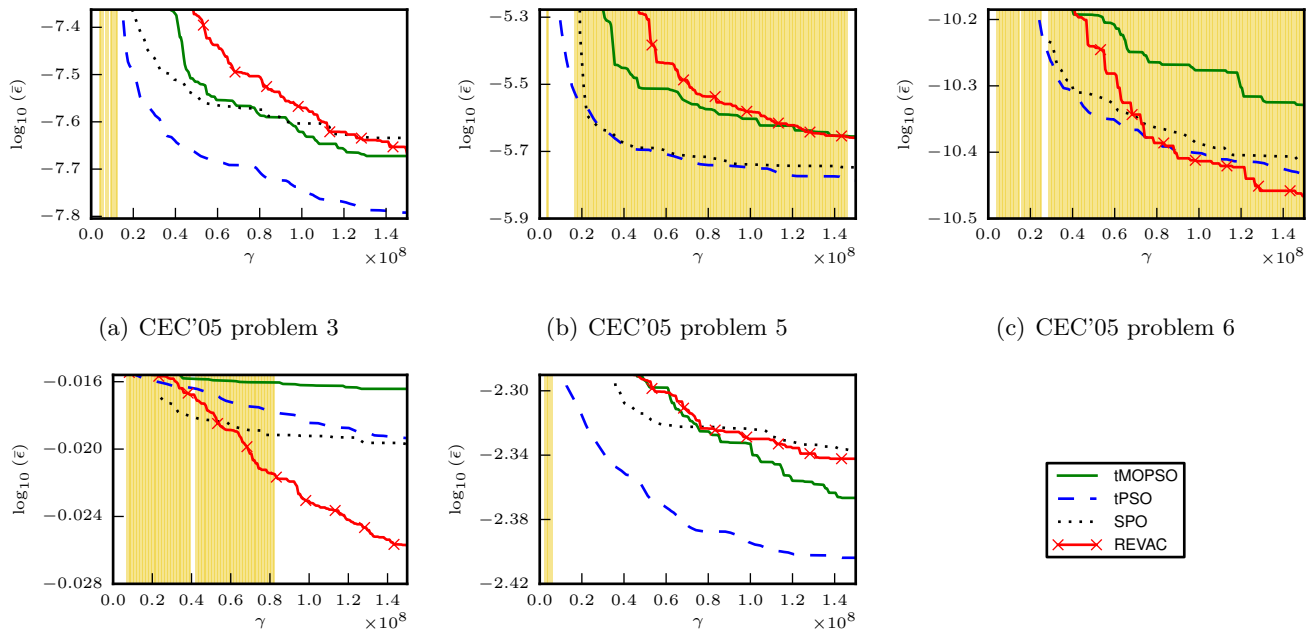
Figure 4: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the DE tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.



(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

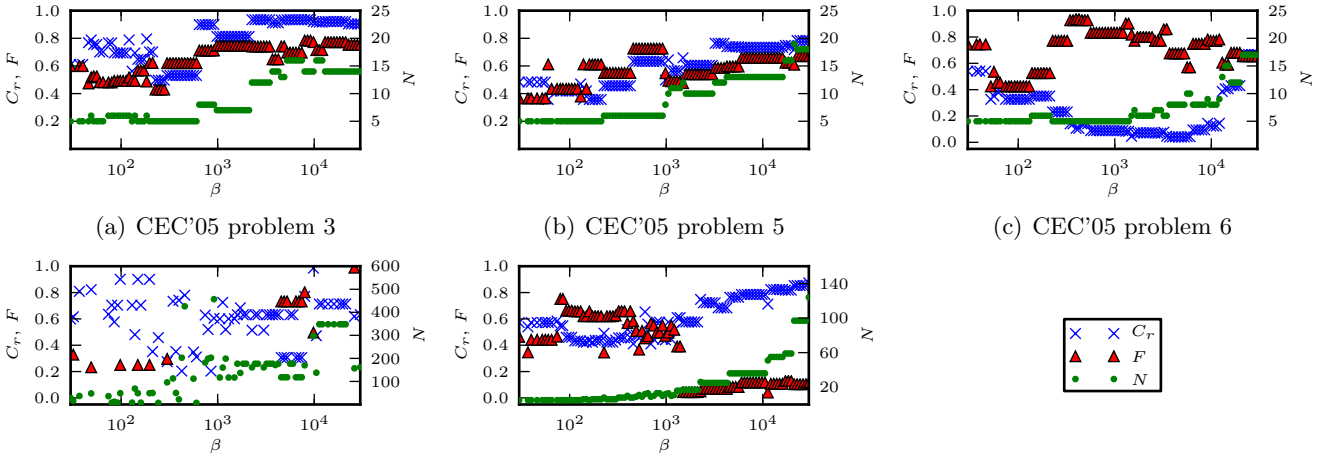
Figure 5: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the PSO tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.



(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

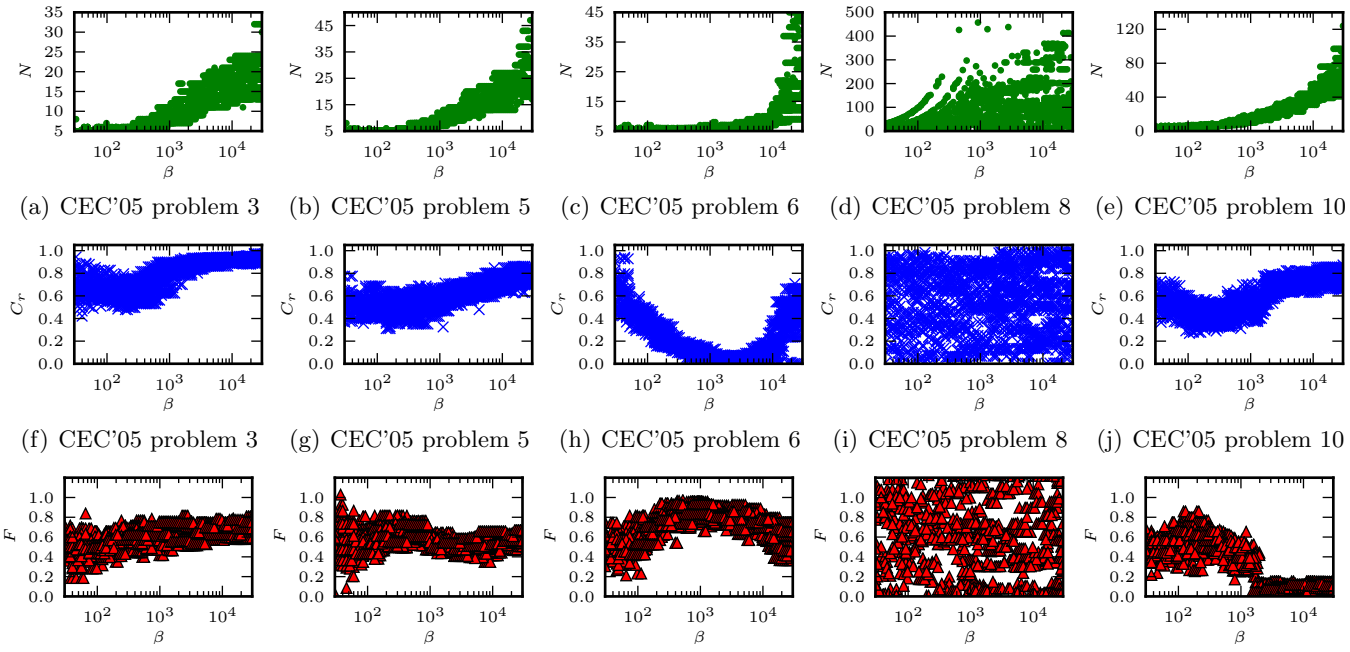
Figure 6: Best mean minimum solution error ($\bar{\epsilon}$) versus the number of application layer evaluations made (γ) of the multi-objective tuning algorithms, applied to the CMA-ES tuning problems. Shaded regions indicate γ where a Mann-Whitney U-test failed to show with a confidence level of 95% that the leading sample mean is better than the other sample means.



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

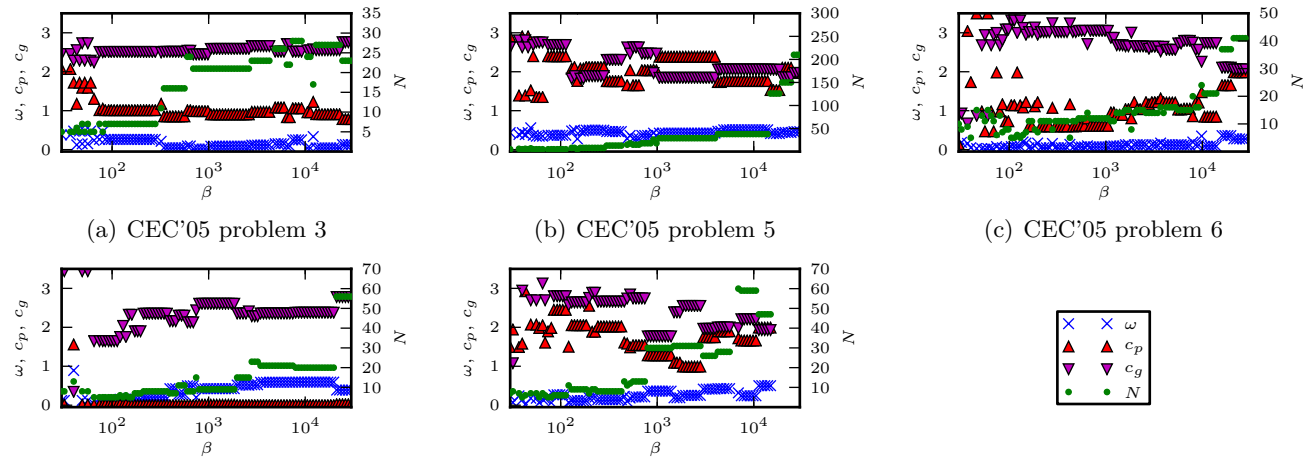


(a) CEC'05 problem 3 (b) CEC'05 problem 5 (c) CEC'05 problem 6 (d) CEC'05 problem 8 (e) CEC'05 problem 10

(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

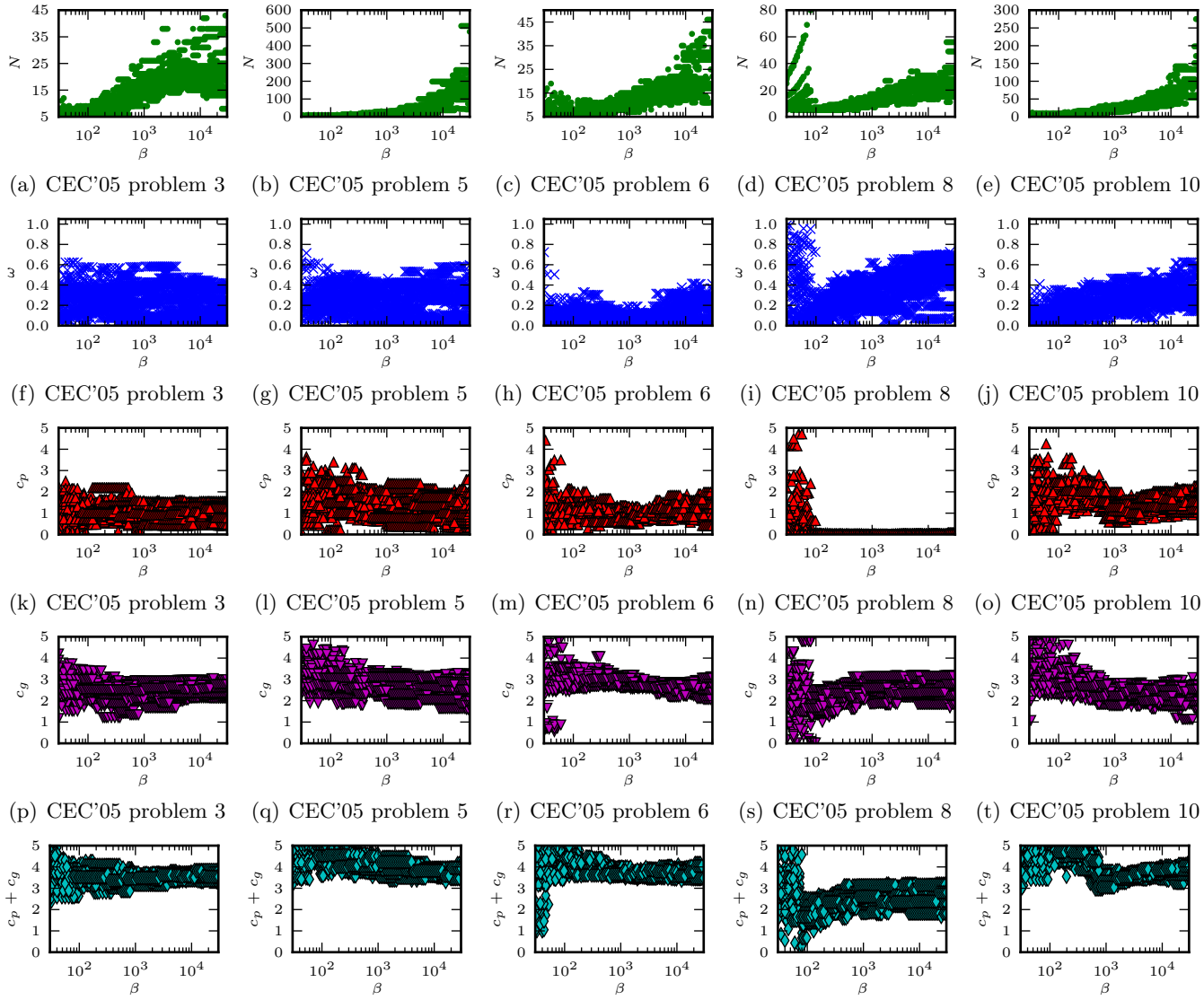
Figure 8: Scatter plot of the optimal CPVs versus the OFE budget found available, for each of the 20 tMOPSO independent tuning runs for DE. Variance is expected since the algorithm being tuned is stochastic, and the resampling approach used only reduces and does not eliminate noise.



(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 9: tMOPSO's results for tuning PSO under multiple OFE budgets, where the results shown are for the 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

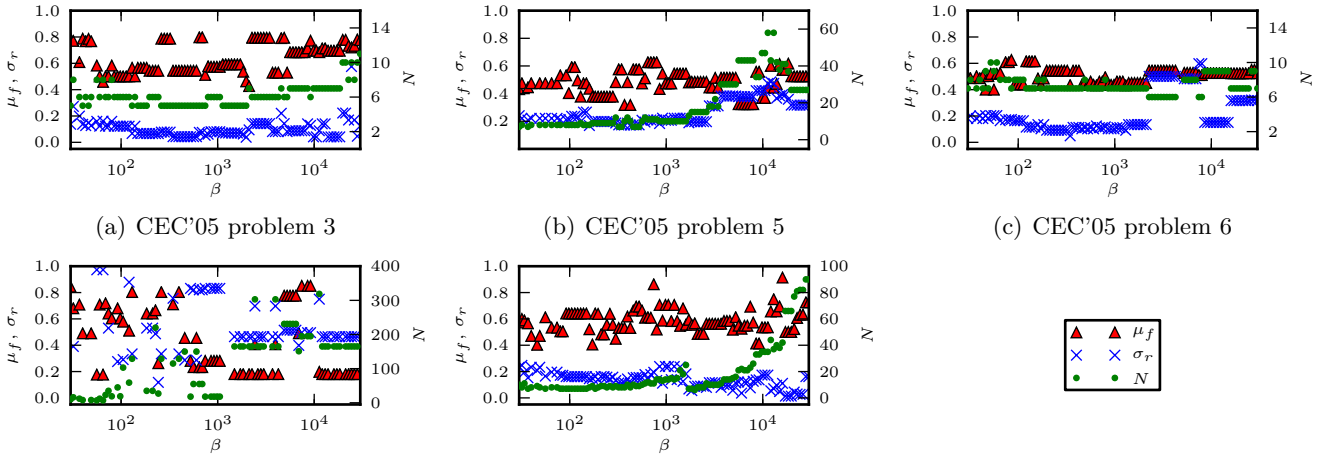
(f) CEC'05 problem 3 (g) CEC'05 problem 5 (h) CEC'05 problem 6 (i) CEC'05 problem 8 (j) CEC'05 problem 10

(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

(p) CEC'05 problem 3 (q) CEC'05 problem 5 (r) CEC'05 problem 6 (s) CEC'05 problem 8 (t) CEC'05 problem 10

(u) CEC'05 problem 3 (v) CEC'05 problem 5 (w) CEC'05 problem 6 (x) CEC'05 problem 8 (y) CEC'05 problem 10

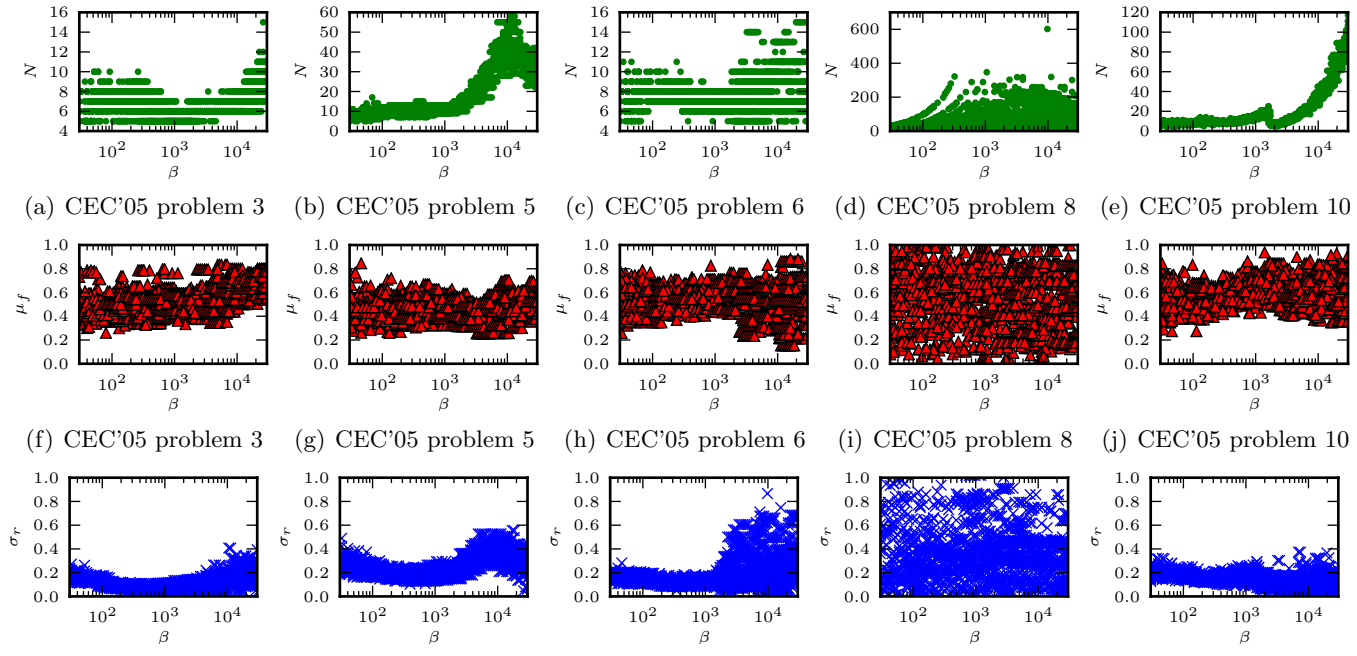
Figure 10: Scatter plot of the optimal CPVs versus the OFE budget found available, for each of the 20 tMOPSO independent tuning runs for PSO. Variance is expected since the algorithm being tuned is stochastic, and the resampling approach used only reduces and does not eliminate noise.



(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 11: tMOPSO's results for tuning CMA-ES under multiple OFE budgets, where the results shown are for the 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 12: Scatter plot of the optimal CPVs versus the OFE budget found available, for each of the 20 tMOPSO independent tuning runs for CMA-ES. Variance is expected since the algorithm being tuned is stochastic, and the resampling approach used only reduces and does not eliminate noise.

Table 1: 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