MOTA: A Many-Objective Tuning Algorithm Specialized for Tuning under Multiple Objective Function Evaluation Budgets

Antoine S. Dymond antoine.dymond@gmail.com
Schalk Kok schalk.kok@up.ac.za
Department of Mechanical and Aeronautical Engineering,
University of Pretoria, South Africa

P. Stephan Heyns stephan.heyns@up.ac.za
Department of Mechanical and Aeronautical Engineering,
University of Pretoria, South Africa

doi:10.1162/EVCO_a_00163

Abstract

Control parameter studies assist practitioners to select optimization algorithm parameter values that are appropriate for the problem at hand. Parameter values are well suited to a problem if they result in a search that is effective given that problem's objective function(s), constraints, and termination criteria. Given these considerations a many-objective tuning algorithm named MOTA is presented. MOTA is specialized for tuning a stochastic optimization algorithm according to multiple performance measures, each over a range of objective function evaluation budgets. MOTA's specialization consists of four aspects: (1) a tuning problem formulation that consists of both a speed objective and a speed decision variable; (2) a control parameter tuple assessment procedure that utilizes information from a single assessment run's history to gauge that tuple's performance at multiple evaluation budgets; (3) a preemptively terminating resampling strategy for handling the noise present when tuning stochastic algorithms; and (4) the use of bi-objective decomposition to assist in many-objective optimization. MOTA combines these aspects together with differential evolution operators to search for effective control parameter values. Numerical experiments consisting of tuning NSGA-II and MOEA/D demonstrate that MOTA is effective at many-objective tuning.

Keywords

Tuning, multiobjective optimization, many-objective optimization, objective function evaluation budgets.

1 Introduction

The performance of an optimization algorithm differs depending on the control parameter values (CPVs) specified by the practitioner. For this reason control parameter studies are often conducted on well-understood testing problems, to better understand the effects of different CPVs on an algorithm's search behavior. Two important considerations when performing control parameter studies are sensitivity to characteristics of the optimization problem, and sensitivity to the termination criteria used. The characteristics of the optimization problem, such as the objective function(s), the constraints imposed, and the initialization conditions, need to be taken into consideration when

Manuscript received: May 27, 2014; revised: March 20, 2015 and July 10, 2015; accepted: July 22, 2015.

selecting CPVs, since different CPV tuples are better suited to certain characteristics than others. In particular, the search mechanics of an algorithm, which are controlled by the CPVs, can be beneficial or detrimental depending on the problem at hand (Wolpert and Macready, 1997). Similarly, CPV tuples well suited to certain termination criteria perform poorly for others; take, for example, the sensitivity of CPV performance to objective function evaluation (OFE) budgets (Dymond et al., 2013).

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

The design of MOTA is motivated by the in-depth control parameter studies a many-objective tuning algorithm would allow (Deb and Srinivasan, 2006). Consider studies investigating robust or generalist CPV tuples that perform well over numerous problems (Smit and Eiben, 2010b; Eiben and Smit, 2011). Multiobjective tuning can efficiently search for these robust CPV tuples by solving a tuning problem with an objective corresponding to each problem that the robust CPVs are required to perform well on. After tuning is completed, the generalist CPV tuples can be found by examining the CPVs tuples found during the multiobjective optimization, each of which is optimal for a different trade-off compromise among the tuning objectives. Furthermore, common practice when assessing a multiobjective algorithm's performance is to make use of a series of unary performance indicators (Zitzler et al., 2003), each of which measures a different aspect of the solution quality. As such, tuning according to multiple performance indicators would allow multiobjective algorithms to be tuned more holistically compared to tuning them according to one performance metric only. Moreover, even if an optimization algorithm is to be tuned to multiple problems separately to determine CPVs well suited to individual problems only, tuning an algorithm to all problems congruently may result in a higher efficiency compared to handling each problem in isolation, since common CPV trends may be present.

The outline of this article is as follows. Related work and MOTA's contribution are discussed in Section 2. The MOTA algorithm is described in Section 3. Thereafter, Section 4 presents the numerical setup used to assess MOTA's performance, and Section 5 gives the results from those numerical experiments.

2 Related Work

The proposed tuning algorithm is related to the fields of control parameter tuning and many-objective optimization.

2.1 Control Parameter Tuning

Control parameter tuning entails adjusting an algorithm's CPVs in order to improve performance (Smit and Eiben, 2009). Accordingly, the control parameters adjusted could refer to real-valued control parameters such as crossover probability, or option-based control parameters such as the crossover method used, or both. The MOTA algorithm is designed for real-valued control parameter tuning. Tuning is done according to a utility metric, which measures the performance of the algorithm being tuned as a function of the CPV tuple being assessed. Traditional examples of utility measures are the objective function value achieved on a given problem for a specified OFE budget, and the number of OFEs required to reach a specified solution accuracy on a given problem. Tuning an algorithm entails solving an optimization problem, where the decision variables are the CPVs, and the objective function consists of the utility metric.

Control parameter tuning differs from parameter control (Eiben et al., 1999). For parameter control an algorithm's CPVs are varied throughout the optimization run according to a predefined strategy, whereas control parameter tuning aims to determine CPVs that remain constant throughout the course of an optimization run. Adaptive algorithms are built on the principle of parameter control, with adaptive algorithms adjusting their CPVs throughout the course of an optimization run in order to tune themselves to the problem being optimized. Superficially adaptive algorithms therefore eliminate the need for control parameter tuning, since CPVs are tuned online by using feedback from the optimization process itself. However, in reality, the practitioner's task simply changes from selecting approximate CPVs to selecting appropriate parameter control strategies for the problem at hand (Pedersen, 2010). Moreover, since parameter control strategies can be expressed parametrically, the task of selecting appropriate control strategies can be expressed as a control parameter tuning problem itself.

Numerous applications of control parameter tuning have been conducted. Initially, Grefenstette (1986) used a genetic algorithm to tune another genetic algorithm to five testing problems. François and Lavergne (2001) proposed the use of statistical methods to model the relation between an algorithm's CPVs and the resulting performance, with the aim of helping practitioners select CPVs for genetic algorithms. Bartz-Beielstein et al. (2005) presented the sequential parameter optimization (SPO) tuning framework, which has been used to tune numerous algorithms such as particle swarm optimization (PSO) algorithms (Eberhart and Kennedy, 1995). Similarly, Nannen and Eiben (2007) proposed the relevance estimation and value calibration (REVAC) tuning algorithm, and demonstrated its effectiveness by tuning the mutation and crossover rates of genetic algorithms. Hutter et al. (2009) presented the ParamILS framework and applied it to tune the CPLEX mixed-integer programming solver. Smit and Eiben (2010a) improved the performance of the winning algorithm from the CEC 2005 competition (Suganthan et al., 2005) using control parameter tuning. There have also been applications of control parameter tuning to investigate the speed versus accuracy trade-off present in many evolutionary algorithms using multiobjective optimization (Dréo, 2008, 2009; Ugolotti and Cagnoni, 2014).

Substantial work has gone into tuning stochastic algorithms. When tuning stochastic algorithms, the utility resulting from a CPV tuple forms a probabilistic distribution. Consequently, stochastic algorithms are typically tuned to the mean of the utility distribution. However, since analytical expressions for the utility distribution mean are normally unavailable or too difficult to calculate, numerical techniques are normally used to determine which CPV tuple results in the best mean utility. The resampling strategy (Beyer, 2000) is the easiest of these numerical techniques to implement, and entails running the algorithm being tuned multiple times for each CPV tuple assessed, to approximate the mean of the CPV tuple's utility distribution. Although effective, the resampling strategy is prohibitively expensive, since the computational cost of assessing each CPV tuple is multiplied by the resampling sample size, a sample size that needs to be large to accurately approximate the mean of a CPV tuple's utility distribution. Pedersen (2010) showed that the computational cost of resampling can be drastically reduced through the use of preemptive termination, whereby the sampling gathering process for a CPV tuple is interrupted if it is already clear that the CPV tuple being assessed is worse than CPV tuples previously evaluated. The F-race algorithm proposed by Balaprakash et al. (2007) makes use of such a preemptively terminating resampling technique. Starting with a large number of candidate CPV tuples, F-race generates one additional sample run for each CPV tuple still in the race. After each iteration of sample generation is completed, a Friedman statistical test is conducted on the sampled utility values. CPV tuples found to be inferior according to the Friedman test given a specified significance level, are then eliminated from the race. This process of generating additional samples and elimination continues until only one CPV tuple remains in the race. Since many CPV tuples are eliminated early in the race, a large reduction of computational expense is achieved.

Control parameter tuning is not intended to be applied directly to application problems, which typically consist of objective or constraint functions that are computationally expensive to evaluate. Rather, control parameter tuning is intended to be applied to computationally cheap testing problems to perform control parameter studies. The extent to which the results of the control parameter studies can be effectively applied to an unseen application problem depends upon various factors. Among these factors are similarity of the tuning problem's objective and constraints to those of the application problem as well as the similarity in termination criteria used. Given these considerations, the applicability of tuning results generated using a single-objective utility is limited. Tuning according to multiple criteria via multiobjective optimization is therefore preferred.

Currently multiobjective tuning algorithms can be split into two groups, namely, tuning to multiple problems each for a single termination criterion, and tuning algorithms designed to tune to a single problem under multiple termination criteria. Smit et al. (2010) proposed the M-FETA algorithm, which is designed for tuning an optimization algorithm to multiple problems, each using one termination criterion. Branke and Elomari (2012) developed the Flexible Budget Method for tuning to a single problem under multiple OFE budgets. For most tuning setups when approximating the performance of a CPV tuple after a specified OFE budget, the algorithm being tuned needs to be run from initialization to that specified OFE budget. The Flexible Budget Method uses this history information and other heuristics to tune under multiple OFE budgets. In particular, each CPV tuple being assessed is run up to the maximum OFE budget of interest. The CPV tuple's run history is then used to gauge performance at each OFE budget being tuned under. Dymond et al. (2015) proposed the tMOPSO algorithm for tuning a single problem under multiple OFE budgets. tMOPSO tunes an optimization algorithm according to a bi-objective utility measure consisting of the best objective function value found, and the number of OFEs used to generate that objective function value. Solving a tuning problem formulated using this bi-objective utility measure allows tMOPSO to determine multiple CPV tuples, each of which is optimal for a different OFE budget. Similarly to the Flexible Budget Method, tMOPSO uses the history information from the CPV assessment calculations to enhance efficiency. Additionally, tMOPSO uses Mann-Whitney U tests (MWUTs; Conover, 1999) specialized for its bi-objective formulation to efficiently perform preemptively terminating resampling.

Here, we propose an algorithm named MOTA, for tuning an algorithm to multiple performance measures under multiple OFE budgets. Such a tuning problem has a utility measure consisting of at least three objectives. When the utility measure consists of four or more objectives, then MOTA needs to solve a many-objective optimization problem.

2.2 Many-Objective Optimization

Multiobjective optimization entails searching for the optimal trade-off compromises for a problem consisting of multiple conflicting objectives. In the context of real-valued optimization problems, each objective f maps a point in the searched decision vector space $\mathbf{x}$ to a real value, $f : \Re^{n_x} \mapsto \Re$, where n_x is the dimensionality of the search space. For the case when the multiobjective optimization problem has constraints, the valid search space is typically expressed through n_g inequality functions and n_h equality functions. Formally, a constrained real-valued multiobjective minimization problem (Engelbrecht, 2007) is defined as

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

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

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

where $\mathbf{F}$ is the multiobjective function, n_f is the number of objectives comprising $\mathbf{F}$, $g_1, g_2, \ldots, g_{n_g}$ are the inequality constraints, and $h_1, h_2, \ldots, h_{n_h}$ are the equality constraints.

The principle of Pareto dominance is commonly used to identify which $\mathbf{x}$ are optimal for different trade-off compromises. A decision vector $\mathbf{x}_1$ Pareto dominates another decision vector $\mathbf{x}_2$ ($\mathbf{x}_1 \prec \mathbf{x}_2$), if $\mathbf{x}_1$ is better or equal than $\mathbf{x}_2$ in all objectives, and if $\mathbf{x}_1$ is better than $\mathbf{x}_2$ for at least one objective. For minimization problems this implies that $\mathbf{x}_1 \prec \mathbf{x}_2$ when

$$f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2), \forall\, i \in 1, 2, \ldots, n_f, \tag{4}$$

and

$$\exists i \in 1, 2, \ldots, n_f : f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2). \tag{5}$$

If neither $\mathbf{x}_1 \prec \mathbf{x}_2$ nor $\mathbf{x}_2 \prec \mathbf{x}_1$, then $\mathbf{x}_1$ and $\mathbf{x}_2$ are relatively nondominated. A decision vector is nondominated if no decision vector exists in the search space that Pareto dominates it.

The set of all nondominated decision vectors for a multiobjective optimization problem is referred to as the Pareto-optimal set (PS), where the PS is often of infinite size. The set of objective function values corresponding to the PS is referred to as the Pareto-optimal front (PF). Two special points in the objective space that are commonly used by multiobjective optimization algorithms are the utopia point $\mathbf{F}^u$ and the nadir point $\mathbf{F}^n$, where for minimization problems $F_i^u = \min\{F_i \,\forall\, \mathbf{F} \in \text{PF}\}$ and $F_i^n = \max\{F_i \,\forall\, \mathbf{F} \in \text{PF}\}$. When the optimization problem consists of two or three objectives, multiobjective evolutionary algorithms typically aim to determine a finite evenly spaced set of nondominated decision vectors to accurately approximate the entire PF. However, approximating the entire PF for many-objective optimization is intractable.

Approximating the entire PF for many-objective optimization problems is problematic for two reasons. First, the computational overhead of maintaining the PF approximations (Mostaghim and Teich, 2005) grows linearly as the size of the approximation increases. Consequently, the computational overhead requirements are too high to approximate the entire PF of many-objective optimization, since the size of the set required to represent the entire PF grows exponentially with the number of objectives. Second, even if a huge Pareto-optimal front approximation (PFA) could be maintained efficiently, the limiting factor would be the OFE budget of the multiobjective optimization algorithm. Suppose that an ultimate multiobjective optimization algorithm existed that with every new decision vector, evaluation was able to find a new nondominated decision vector. Even this ultimate algorithm's PFA would be limited to the size of the OFE budget assigned to it.

Many-objective optimization algorithms therefore do not try to approximate the entire PF but rather make use of other criteria in addition to Pareto dominance to guide the optimization process. A commonly used approach to differentiate between Pareto nondominated decision vectors during the course of an optimization run is to make use of performance indicators (Zitzler and Künzli, 2004). An example is the SMS-EMOA algorithm proposed by Beume et al. (2007), which uses the hypervolume (HV) performance indicator in conjunction with Pareto dominance in order to optimize a multiobjective problem. Alternatively, Di Pierro et al. (2007) proposed a preference-ordering strategy that considers a decision vector's dominance status according to various subsets of objectives, thereby allowing for further differentiation. Then there are decomposition-based approaches (Zhang and Li, 2007), for which the multiobjective problem is divided into subproblems that are all solved simultaneously. All these many-objective approaches ultimately require some a priori input from the practitioner as to which sections of the PF or preference articulations are more important than others. The indicator-based approaches favor decision vectors aligned with indicators chosen, whereas preference-ordering strategies favor decisions close to the center or the edges of the PF depending on the parameters specified, and decomposition-based approaches focus on areas of the PF specified in the subproblem construction. This a priori input is undesirable, as it breaks from the clean a priori free approach followed when three or fewer objectives are optimized. However, because of the intractability of approximating the entire PF for many-objective optimization problems, some a priori input is required.

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

3 MOTA Algorithm

The many-objective tuning algorithm (MOTA) is specialized for tuning stochastic algorithms to multiple criteria under multiple OFE budgets. MOTA's specialization consists of four parts:

- A tuning problem formulation that has both a speed objective and a speed decision variable (Section 3.1)
- A CPV tuple assessment procedure that uses the history information from a CPV tuple assessment run in order to gauge performance simultaneously at multiple OFE budgets (Section 3.2)
- A bi-objective decomposition approach to assist in the many-objective optimization (Section 3.3)
- A preemptively terminating resampling strategy to handle the noise resulting from tuning stochastic algorithms (Section 3.4)

The discussion of these four aspects of MOTA's specialization is followed by an algorithm overview in Section 3.5.

3.1 Tuning Problem Formulation

The tuning problem formulation solved by MOTA makes use of a decision vector consisting of control parameter values $v_1, v_2, \ldots, v_n$ together with an auxiliary variable β_a, where β_a specifies at which OFE budget the $v_1, v_2, \ldots, v_n$ CPV tuple is to be assessed. MOTA's tuning problem formulation uses a multiobjective utility measure to tune an optimization algorithm to multiple utility indicators for multiple OFE budgets. The multiobjective utility measure $\mathbf{u}$, which MOTA minimizes, is defined as

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_{n_u} \\ \beta_a \end{bmatrix}, \tag{6}$$

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

The choice of utility indicators for $\mathbf{u}$ depends on the control parameter study being performed. When tuning a single-objective algorithm to multiple problems, a sensible choice for the utility indicators would be the lowest solution error achieved for each of those problems. Alternatively, when tuning a multiobjective optimization algorithm, a utility indicator for each unary performance indicator of interest could be used. The computational cost of evaluating the utility indicators should be accounted for by tuning practitioners when specifying how extensively the history information from a CPV tuple assessment run should be used.

3.2 Utilizing the History Information from a CPV Tuple Assessment Run

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

$$u_i(N) \rightarrow u_i(2N) \rightarrow \cdots \rightarrow u_i(\beta_a), \tag{7}$$

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

Because of computational overhead considerations and storage requirements, tuning practitioners are normally not interested in determining effective CPV tuples for *every* OFE budget less than the maximum OFE budget of interest, $\beta_{\max}$. Normally, a tuning practitioner is only interested in a subset of OFE budgets $\in \{1, 2, \ldots, \beta_{\max}\}$, such as OFE budgets spaced logarithmically between the minimum OFE budget of interest and $\beta_{\max}$. Given this consideration, MOTA calculates the following utility indicator values

$$u_i(\beta) \, \forall \, \beta \in B : \beta \leq \beta^+, \tag{8}$$

where B is the target OFE budgets selected by the tuning practitioner, and the overshoot budget β^+ specifies the maximum OFE budget for which the utility indicator values are to be calculated from the sampling runs. For MOTA, β^+ is calculated according to a user-specified function of β_a, for example, $\beta^+ = 1.6\beta_a + 100$. The optimal function for determining β^+ is expected to be problem-specific but is also not expected to drastically alter performance because of the noise-handling strategy used by MOTA (see Section 3.4).

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

3.3 Bi-objective Decomposition

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

In the case of MOTA, bi-objective decomposition is used, as it is well suited to processing the additional utility indicator values generated from a CPV tuple's sampling run. The generalized objective function for the bi-objective decompositions that are to be minimized is composed of one objective consisting of a scalarization of the $u_1, u_2, \ldots, u_{n_u}$ utility indicator values and the second objective, the OFE budget used to calculate those utility indicator values. Two commonly used scalarization approaches are the aggregated or weighted sum approach and the Tchebycheff approach (Zhang and Li, 2007). Both of these approaches make use of a weights vector $\mathbf{w}$ in the scalarization process. Specifically, the weighted sum scalarized value $\hat{u}_w$ is determined as

$$\hat{u}_w = \sum_{i=1}^{n_u} w_i u_i, \tag{9}$$

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

$$\hat{u}_T = \max\{w_i(u_i - z_i) \quad \forall\, i \in 1, 2, \ldots, n_u\}, \tag{10}$$

where the z_i values correspond to a chosen reference point. In order to make the process of selecting $\mathbf{w}$ easier for tuning practitioners, MOTA by default normalizes the objective values passed to the selected scalarization function. Specifically, the u_i values are normalized between the utopia and nadir points of MOTA's current PFA. When the objective values are normalized, zero values are used for all the z_i reference values in Eq. (10).

The choice of a scalarization approach for the utility indicator values should be made according to the PF characteristics and according to preference articulations of interest. The Tchebycheff approach is able to handle both convex and concave PFs, while the weighted sum approach can only handle concave PFs, as illustrated in Figure 1. However, the Tchebycheff approach is more prone than the weighted sum approach to being controlled by a single objective only. If the user selects the Tchebycheff approach for the j'th subproblem, the corresponding bi-objective minimization function that needs to be minimized $\mathbf{u}_j^{2D}$ is defined as

$$\mathbf{u}_j^{2D} = \begin{bmatrix} \max\{w_i\, \tilde{u}_i \quad \forall\, i \in (1, 2, \ldots, n_u)\} \\ \beta_a \end{bmatrix}, \tag{11}$$

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

Another key aspect to decomposition-based approaches is neighborhoods. Neighborhoods are vital, since they are used to share information between subproblems, thereby differentiating decomposition-based approaches from approaches where subproblems are optimized in isolation from each other. Here, neighborhoods are split into two categories:

- Candidate generation neighborhoods. When generating a candidate decision vector for a target subproblem, this neighborhood specifies the additional

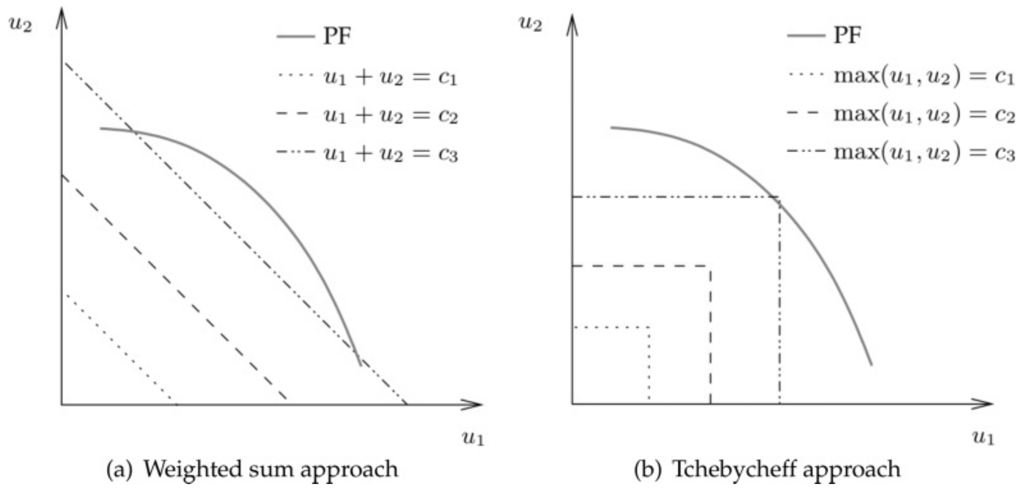


Figure 1: Illustration of the weighted sum and the Tchebycheff scalarization approaches. Contour lines of equal value according to the scalarization approach are shown for the values c_1, c_2, and c_3.

subproblems from which information is used for operations such as crossover and mutation.

- Update neighborhoods. After evaluating a decision vector generated for a target subproblem, the resulting objective function values are also used to update the solutions of the subproblems in this neighborhood.

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

3.4 Handling the Noise Resulting from Tuning Stochastic Algorithms

When tuning optimization algorithms with stochastic elements, standard utility indicator values become probabilistic distributions. A noise-handling strategy aims to approximate a probabilistic distribution when an analytical expression is unavailable, which is typically the case. Noise handling in the context of parameter tuning under multiple OFE budgets is complicated by varying noise distribution characteristics depending on the location in the objective space (Dymond et al., 2015). This variance rules out many specialized noise (Bui et al., 2009) or uncertainty (Xi et al., 2012) handling strategies, which assume a uniform uncertainty distribution throughout the objective space.

Given the varying uncertainty distributions throughout the objective space, MOTA uses a resampling approach (Beyer, 2000). Resampling-based approaches entail tuning algorithms according to the approximated mean of the utility indicator's probabilistic distribution. Specifically, the resampling technique consists of performing multiple independent runs for the CPV tuple being assessed, and then using the resulting sample to approximate the mean utility indicator value. Based on tMOPSO (Dymond et al., 2015), MOTA uses a preemptively terminating resampling strategy whereby the sampling gathering processes is interrupted if Mann-Whitney U tests (MWUTs; Conover, 1999) indicate that the decision vector being assessed is unlikely to result in an improvement on the current approximation of the PF.

MOTA's noise-handling procedure starts with a group of decision vectors X, where each $\mathbf{x} \in X$ has an associated CPV tuple, assessment OFE budget, and target subproblem. Initially a small number of samples are generated for each $\mathbf{x}$, for the OFE budgets $\mathbf{x}_B$. As outlined in Section 3.2, $\mathbf{x}_B$ is controlled by the user-specified target OFE budgets B and the overshoot function β^+ as

$$\mathbf{x}_B = \{\beta \,\forall\, \beta \in B : \beta \leq \beta^+\}. \tag{12}$$

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

$$P_k \preceq_\alpha \mathbf{u}'^{2D}_j \quad \forall\, k \in T_u, \tag{13}$$

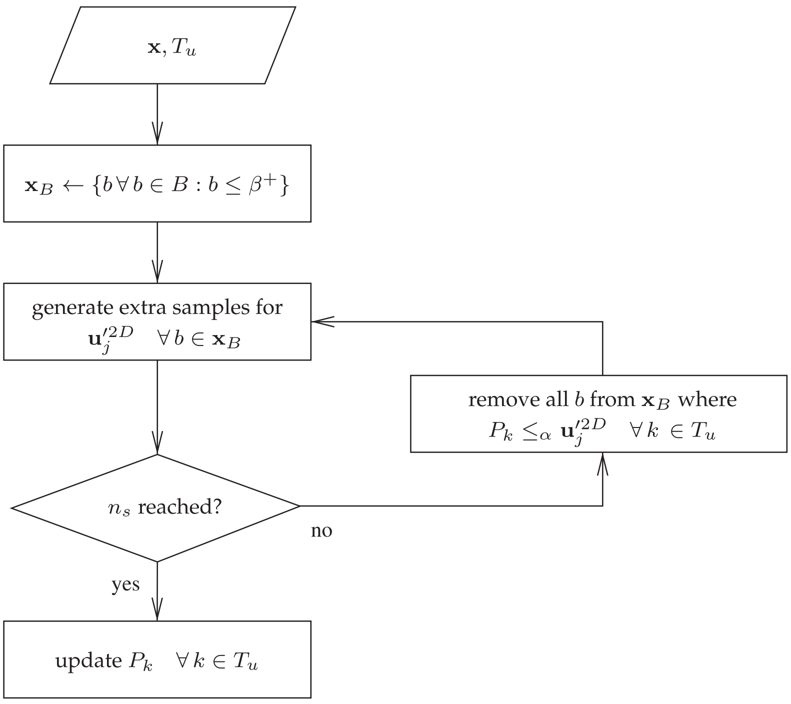
where $\preceq_\alpha$ denotes "likely to be dominated" given the significance level of α, and P_k is the k'th subproblem's PFA.

Depending on the computational cost of calculating the utility indicators compared to performing a CPV sampling run, two different options are available for conducting Pareto nondominance likelihood checks:

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

For the case where a cheap utility indicator value is used, such as the objective solution error achieved by a single-objective algorithm, the option of reducing the maximum OFE is sensible. Alternatively, when an expensive utility indicator such as HV is used, then the option of checking all the OFE budgets is more appropriate.

Resampling interruption checks continue until either $\mathbf{x}_B$ is empty or the desired resampling size n_s is reached. If the desired resampling size n_s is reached, the approximated utility values are used to update the T_u subproblems' PFAs. MOTA users control the aggressiveness of the resampling interruption through two control parameters, namely, the number of sample increments between resampling interruption checks, Δ_{n_s}, and the interruption significance level used, α. The bi-objective nature of the subproblems is exploited in order to efficiently perform PFA dominance and dominance likelihood checks, as in Dymond et al. (2015). A flow chart for the CPV tuple assessment procedure for the *check all* noise-handling approach is shown in Figure 2.



$\mathbf{x}, T_u$
$\mathbf{x}_B \leftarrow \{b \forall b \in B : b \leq \beta^+\}$
generate extra samples for $\mathbf{u}_j^{\prime 2D} \quad \forall b \in \mathbf{x}_B$
remove all b from $\mathbf{x}_B$ where $P_k \leq_\alpha \mathbf{u}_j^{\prime 2D} \quad \forall k \in T_u$
n_s reached?
no
yes
update $P_k \quad \forall k \in T_u$

Figure 2: Flow chart of MOTA's CPV tuple assessment procedure when computationally expensive utility indicator values are calculated.

3.5 Algorithm Overview

MOTA tunes an optimization algorithm to multiple criteria over a range of OFE budgets, using the aforementioned concepts. The decision space MOTA searches is set up according to the tuning formulation presented in Section 3.1, where each decision vector is of the following form:

$$\mathbf{x} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ \ln \beta_a \end{bmatrix}, \tag{14}$$

where $v_1, v_2, \ldots, v_n$ are the real CPVs optimized, and β_a is the OFE budget at which the $v_1, v_2, \ldots, v_n$ CPV tuple is to be assessed. The natural logarithm of β_a is optimized in place of β_a directly, because of the logarithmic trends typically observed when tuning under multiple OFE budgets (Dymond et al., 2015).

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

$$\mathbf{x}_0 = \mathbf{I}_l + \mathbf{r}() \circ (\mathbf{I}_u - \mathbf{I}_l), \tag{15}$$

where $\mathbf{I}_l$ is the lower initialization bound, $\mathbf{I}_u$ is the upper initialization bound, $\circ$ is the Hadamard product operator, and $\mathbf{r}()$ is a function returning a vector whose individual element values are randomly generated independently of each other between 0 and

1 with a uniform probability density. These initial decision vectors are evaluated as outlined in the Section 3.4 to generate the initial subproblem PFAs.

After initialization, MOTA uses operators based upon differential evolution (DE; Storn and Price, 1997) to generate new candidate decision vectors. Before explaining the manner in which the DE operators are extended into the context of MOTA's tuning formulation, the traditional single-objective DE operators are described. DE has numerous strategies available, such as *rand/1/bin* and *best/1/bin*. The DE *rand/1/bin* candidate decision vector generation process for the i'th member of the population begins with generating a mutant vector $\mathbf{m}_i$ as follows:

$$\mathbf{m}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}), \tag{16}$$

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

$$x_{i,k}^c = \begin{cases} m_{i,k} & \text{if } r() < C_r \text{ or } k = k_f \\ x_{i,k} & \text{otherwise,} \end{cases} \tag{17}$$

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

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

$$\ln \beta_r = \ln \beta_t + r_g() \cdot \beta_\delta \cdot (\ln B_{\max} - \ln B_{\min}), \tag{18}$$

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

$$\mathbf{m}_i = \mathbf{x}_{s_1} + \mathbf{r}() \circ F(\mathbf{x}_{s_2} - \mathbf{x}_{s_3}), \tag{19}$$

where the $\mathbf{r}()$ term is added to promote search diversity (Salehinejad et al., 2014). DE crossover is then conducted between the resulting mutant vector and the decision vector from the i'th subproblem, which is optimal for an OFE budget of β_t, as in Eq. (17). Last, the assessment OFE budget of the generated candidate decision vector is set to β_t.

Constraint handling is achieved by regenerating candidate vectors until all constraints are satisfied. Although this approach is not viable for applications consisting of computationally expensive constraints, it is acceptable for tuning applications, since tuning constraints are usually computationally cheap, e.g., $N > 5$. MOTA does not enforce or have a specialized strategy for handling bound constraints, since for many tuning problems sensible CPV bounds are difficult to determine a priori. Additionally, MOTA has an internal constraint that requires that the candidate decision vector be different from the $\mathbf{x}_i$ vector it is trying to improve upon. This internal constraint is necessary for the beginning stages of MOTA's tuning optimization, for which the subproblem's PFAs consist of a low number of unique decision vectors. If for a given subproblem, candidate generation using the DE operations in Eqs. (19) and (17) fails to satisfy the constraints 10 times in a row during a single iteration, then a randomly generated valid decision vector is used, as outlined in Eq. (15). Once the candidate decision vectors are generated for all the subproblems, these candidate decision vectors are assessed to update the subproblem PFAs, as outlined in Section 3.4.

Generation of the candidate decision vectors continues until all the subproblems become inactive. A subproblem becomes inactive when one of its termination criteria is satisfied, signaling that no more candidate vectors should be generated for that subproblem. If a subproblem is inactive but is in the update neighborhood of an active subproblem, the inactive subproblem's PFA is still updated when the active subproblem's candidate vector is assessed. Per subproblem termination or inactivity criteria are appealing, since they allow for greater control compared to making all subproblems inactive at the same time. Take, for instance, allocating differing amounts of computational resources for each subproblem, or adding stagnation termination criteria whereby a subproblem becomes inactive if no substantial improvements has been made recently.

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

4 Numerical Setup

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

- be more efficient at tuning an optimization algorithm to each problem of a test suite compared to tuning an algorithm to each problem instance in isolation;
- be better suited to determining generalist CPV tuples that perform well over an entire problem test suite; and
- would be able to tune MOEAs more holistically by tuning them according to multiple unary performance metrics simultaneously.

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

Algorithm 1 MOTA pseudocode

```
procedure MOTA
    for $s_p \in$ subproblems do
        repeat
            generate $\mathbf{x}_0$                                   ▷ (15)
        until $\mathbf{x}_0$ valid
    end for
    evaluate all the generated $\mathbf{x}_0$                     ▷ Section 3.4
    while a subproblem is active do                            ▷ main loop
        for $s_p \in$ active subproblems do
            repeat
                compute $\beta_r$                                    ▷ (18)
                compute $\mathbf{m}$                                    ▷ (19)
                compute $\mathbf{x}^c$                                   ▷ (17)
            until $\mathbf{x}^c$ valid
        end for
        evaluate all the generated $\mathbf{x}^c$                  ▷ Section 3.4
    end while
end procedure
```

MOTA is benchmarked with regard to the first two statements. For brevity, the tuning of multiobjective algorithms to multiple unary performance metrics is left for future work.

4.1 Tuning Problems Used

The tuning problems used are built around the commonly used ZDT (Zitzler et al., 2000), DTLZ (Deb et al., 2005), and WFG (Huband et al., 2006) multiobjective test problem suites. These test problem suites are designed to gauge an algorithm's performance given a range of objective function characteristics. Furthermore, the ZDT problems are scalable in the number of decision variables, while the DTLZ and WFG problems are scalable in both the number of decision variables and the number of objectives. Since the numerical experiments entail tuning algorithms to these problem suites, lower-than-normal dimensional versions of the WFG and DTLZ problems are used, to make computational costs of the experiments more manageable. For the bi-objective ZDT problems, algorithms are tuned to ZDT problems 1, 2, 3, 4, and 6, where the standard setup of 30 decision variables is used for problems 1, 2, and 3, while 10 decision variables are used for problems 4 and 6. The fifth ZDT problem is omitted because it has binary decision variables. Regarding the WFG problems, two position decision variables, ten distance decision variables, and two objectives are used for all the problems. For the DTLZ problems the number of decision variables is kept at the commonly used values of 7, 12, 12, 12, 12, 12, and 22 for problems 1 through 7, respectively, while the number of objectives is reduced from the standard three to only two.

Selected multiobjective optimization algorithms are tuned to these problem suites, according to inverted generational distance (IGD; Zhang et al., 2008), which is sensitive to both the distance to and coverage of the true PF. Initially tuning according to hypervolume (HV; Zhang et al., 2008) was considered, but was later discarded because of the problem-specific effort of selecting HV reference points sensitive to the performance at low OFE budgets. The tuning objectives are to minimize the IGD for each problem in the test suite the algorithm is being tuned to while minimizing the OFEs used. Therefore, tuning an algorithm to the nine WFG problems entails solving a tuning problem with the ten objectives:

$$F = \begin{bmatrix} IGD_{WFG1} \\ IGD_{WFG2} \\ \cdots \\ IGD_{WFG9} \\ \beta_a \end{bmatrix}, \tag{20}$$

where IGD_{WFGi} is the IGD achieved on the i'th WFG problem given the CPV tuples being assessed using an OFE budget of β_a. Regarding the OFE budgets that algorithms are to be tuned under, 51 OFE budgets spaced logarithmically between 10 to 10,000 are used.

Separate problems are constructed for each algorithm tuned in order to have a tuning problem that entails determining specialist CPVs only, and to have a tuning problem for determining generalist and specialist CPVs together. For each specialist tuning problem, the bi-objective PF sections of interest correspond to each problem in the test suite in isolation together with the OFE budget objective. For example, the WFG specialist tuning problems have nine subproblems with the utility weight vectors of $[1, 0, \ldots, 0]$, $[0, 1, 0, \ldots, 0]$, $\ldots$, $[0, \ldots, 0, 1]$. For the generalist tuning problems, additional preference articulations are added to determine CPV tuples that perform well for all utility objectives, i.e., $[1, 1, \ldots, 1]$, and for all leave-one-out combinations, i.e., $[0, 1, \ldots, 1]$, $[1, 0, 1, \ldots, 1]$, $\ldots$, $[1, \ldots, 1, 0]$. All the generalist subproblems make use of weighted sum scalarization in the normalized objective space for their bi-objective decomposition. The leave-one-out preference articulations are added to enable scrutiny of the results of the $[1, 1, \ldots, 1]$ articulation, to determine if one objective has a disproportionate effect even though scalarization uses a normalized objective space.

4.2 Algorithms Tuned

The algorithms tuned to the ZDT, DTLZ, and the WFG problems are the commonly used second version of nondominated sorting genetic algorithm (NSGA-II; Deb et al., 2002) and the multiobjective evolutionary algorithm based on decomposition (MOEA/D; Beume et al., 2007). The NSGA-II and MOEA/D implementations, from the jMetal software package (Durillo and Nebro, 2011)[2] are used in these experiments. Selected real and integer CPVs are tuned for NSGA-II and MOEA/D, while option-based control parameters are left on their jMetal defaults. For NSGA-II the tuned control parameters are the population size N, the crossover probability c_p, and the mutation probability m_p. The selection, crossover, and mutation operators are fixed to binary tournament selection, simulated binary crossover (Deb and Agrawal, 1994), and polynomial mutation, respectively. The tuning initialization bounds for NSGA-II are $N \in [10, 200]$, $c_p \in [0, 1]$,

[2]jMetal version 4.3 from http://jmetal.sourceforge.net/ is used. The default NSGA-II and MOEA/D implementations in jMetal 4.3 handle the OFE budget termination criteria in different manners. The NSGA-II implementation checks against the OFE budget constraint after every pair of offspring are created, i.e., every two OFEs. In contrast the MOEA/D implementation checks against the OFE budget constraint at the end of each iteration after all the subproblem candidates have been evaluated. The NSGA-II implementation is modified so that the same behavior as MOEA/D is achieved, so that the tuning results can be compared against previous studies (Dymond et al., 2013).

and $m_p \in [0, 1]$. The tuning constraints for NSGA-II are

$$5 \leq N \leq 500 \tag{21}$$

$$0 \leq c_p \leq 1 \tag{22}$$

$$0 \leq m_p \leq 1. \tag{23}$$

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

$$5 \leq N_s \leq 500 \tag{24}$$

$$0 \leq T_f \leq 1 \tag{25}$$

$$2 \leq N_s T_f \tag{26}$$

$$0 \leq C_r \leq 1 \tag{27}$$

$$0 \leq F \leq 2. \tag{28}$$

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

The computational budget allocated to the tuning problems corresponds to the upper limit of what is deemed to be the standard use-case scenario. This upper limit is chosen as the computational work produced by a high-end desktop or laptop computer left to tune overnight. Specifically, the allowable computational budget is 12 hours of fully utilizing a fourth-generation Intel® Core™ i7-4700MQ processor. For the tuning problems described, the WFG generalist tuning problems are the most computationally expensive. Given this limiting factor, a computational budget is assigned to each subproblem equivalent to assessing 1,000 CPV tuples up to the maximum OFE budget of 10,000 without resampling on a single problem of the relevant test suite, giving a tuning budget (γ) of 10^7. Since evaluating a CPV tuple on a generalist subproblem entails assessing the CPV tuple on all problems in the test suite being tuned to, the specified γ allows for fewer CPV tuples to be evaluated compared to a specialist subproblem.

4.3 Tuning Algorithms Compared

The tuning algorithms compared in the numerical experiments are MOTA, tMOPSO, and a base-line heuristic RAND^M. tMOPSO is compared against MOTA, since it has been shown to be effective at tuning optimization algorithms under multiple OFE budgets (Dymond et al., 2015). RAND^M, which is a basic random search heuristic equipped with MOTA's CPV assessment procedure, is added to the numerical experiments to gauge the benefits of MOTA's DE-based process for generating candidate decision vectors. RAND^M generates a candidate decision vector for the i'th subproblem, $\mathbf{x}_i^c$ using a uniform random distribution as follows:

$$x_{i,k}^c = \begin{cases} \ln \beta_t & \text{if } k = 1 \\ x_{i,k}^{b_t} + (r() - 0.5)(I_{u,k} - I_{l,k}) & \forall\, k \in 2, n_x, \end{cases} \tag{29}$$

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

In order to apply the bi-objective tMOPSO tuning algorithm to the many-objective tuning problems, each tuning problem is broken up into uncoupled bi-objective subproblems where no information is shared. The tMOPSO runs to determine the generalist CPV tuples occur last, since those runs require information on the nadir and utopia points, which are taken from the tMOPSO runs targeted on specialist CPVs. Regarding the generalist tuning problems, MOTA and RAND^M have an advantage over tMOPSO in that these algorithms share information between the solution approximations of the different subproblems. tMOPSO as a bi-objective tuning algorithm has no mechanics for sharing this many-objective information. MOTA and RAND^M use the same neighborhood topology for the tuning problems. In particular, the specialist subproblem and the overall generalist subproblem have a zero-sized update neighborhood, while each of the leave-one-out generalist subproblems have an update neighborhood of size 3, consisting of two other leave-one-out generalist subproblems as well as the overall generalist subproblem. Admittedly, the choice of the update neighborhoods for the leave-one-out generalist subproblems is rather arbitrary, being based only on what we deemed to be sensible based upon our experience with the problems. For all cases, MOTA and RAND^M use a candidate generation neighborhood consisting of all the subproblems.

The same resampling interruption procedure is used by the compared tuning algorithms. This commonality should ensure that any performance difference observed is not influenced by the use of difference noise-handling procedures. For MOTA, tMOPSO, and RAND^M a total sampling size of 25 is used, with resampling interruption checks occurring after sampling increments of $1, 2, 3, 4, 5$, and 10 ($\Delta_{n_s} = [1, 2, 3, 4, 5, 10]$) using MWUTs given an interruption significance (α) of 60%. Since the IGD calculations are of moderate computational cost, resampling interruption checks are conducted for each OFE budget constraint under which a CPV tuple is being assessed. A common constraint-handling approach is also used by the compared tuning algorithms. In particular, the candidate generation process repeats until a valid decision vector is found, subject to a threshold of 10 attempts. After 10 attempts, random values are generated inside the initialization bounds until a valid candidate is generated. Finally, all algorithms use an OFE budget overshoot function of $\beta^+ = 1.6\beta_a + 100$.

The comparison of MOTA and tMOPSO is complicated by the fact that the performance of these algorithms is sensitive to their CPVs. These algorithms are therefore tuned before being compared to ensure that MOTA and tMOPSO use CPVs suitable for the problems on which they are going to be compared. In contrast, the base-line RAND^M does not require precomparison tuning because it does not have any control parameters. The precomparison tuning of MOTA and tMOPSO is done using the tuning particle swarm optimization (tPSO) algorithm (Dymond et al., 2015). tPSO is set up to search for CPVs well suited for the NSGA-II ZDT specialist problem and for the NSGA-II DTLZ specialist problem. The results from these tPSO runs are to be compared against each other to check consistency. MOTA and tMOPSO are not tuned to the NSGA-II WFG specialist tuning problem, since the WFG tuning problems are computationally expensive in this numerical setup, being more than 10 times more expensive than their ZDT and DTLZ counterparts. tPSO tunes each respective algorithm according to the utility measure u_{tPSO}. u_{tPSO} is a weighted sum performance aggregation over all the respective subproblems for the tuning problem. For the NSGA-II DTLZ specialist problem which has seven subproblems or preference articulations, tPSO minimizes

$$u_{tPSO} = -\sum_{i=1}^{7} \tau_i, \tag{30}$$

where τ_i is the bi-objective HV of the i'th bi-objective subproblem's PF, which has been normalized between a commonly used nadir and utopia point. The same setup is used for tuning to the NSGA-II ZDT specialist problem. To account for stochastic effects, resampling over 20 independent runs is conducted. Based on Dymond et al. (2015), the tPSO settings are a swarm size of 10 and an inertia factor of 0.5. The tPSO tuning budget is 400 CPV tuple evaluations, which is equivalent to assessing 20 CPV tuples using a resampling size of 20. tPSO performs preemptive resampling termination checks after every resampling iteration using MWUTs and an interruption significance level of 66%. Regarding handling tuning constraints, if an invalid CPV tuple is generated, tPSO continues to regenerate that tuple until all constraints are satisfied.

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

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

5 Numerical Results

The results from the numerical experiments are presented in three parts. First the results are given from the tPSO tuning to determine effective CPVs for MOTA and tMOPSO. Thereafter, the numerical results from the specialist tuning problems are presented, followed by those of the generalist tuning problems. For brevity, extensive results are not included in this paper but are made available as supplementary material. Included in the supplementary material are the utopia and nadir points used in the result analysis.

When analyzing the results from the tuning problems, the objective function values of respective PFAs are normalized using a common objective normalization function. Use of a common objective normalization function is required, since some variance is expected in the utopia and nadir points approximations made by MOTA, tMOPSO, and RAND^M. Therefore the objective value results are renormalized between a common nadir and utopia point, for purposes of fair comparison. The comparison utopia and nadir points were calculated after MOTA, tMOPSO, and RAND^M were applied to the specialist tuning problems, by combining the tuning results. Specifically, a PFA was constructed for each tuning problem by combining all the results for that problem. Thereafter, the utopia and nadir points used to compare the tuning algorithms were taken from these constructed PFAs. This approach could not, however, be followed for the tPSO tuning of MOTA and tMOPSO, since tPSO needs to compare performances during the course of the tuning run, and as such cannot postpone the calculation of the

Table 1: CPV tuples that tPSO found effective for MOTA on specialist problems.

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

Table 2: CPV tuples that tPSO found effective for tMOPSO on specialist problems.

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

normalization utopia and nadir points. Therefore tMOPSO was applied to the relevant specialist tuning problem to determine normalization utopia and nadir points for use in the tPSO tuning runs. In particular, the results from 10 independent runs of tMOPSO using the CPV settings from Dymond et al. (2015) were combined into one PFA to determine the tPSO normalization utopia and nadir points.

5.1 Selecting the CPVs for the Compared Tuning Algorithms

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

Table 1 shows the CPV tuples that tPSO found to be effective for MOTA on the NSGA-II ZDT and DTLZ specialist problems. An acceptable level of consistency is observed among the tPSO CPV recommendations for MOTA, with similar values for F, C_r, and β_δ being returned. Regarding tMOPSO, Table 2 shows tPSO's CPV recommendations. An acceptable level of consistency in terms of exploitation versus exploration is observed again, if the combined effects of ω, $c_p + c_g$ (Clerc and Kennedy, 2002) and the tMOPSO OFE perturbations c_β factor are considered. For example, the tPSO recommendation that has a far higher ω than the other tPSO ω recommendations is accompanied by a lower $c_p + c_g$ value relative to the other tPSO runs. As expected, different CPV recommendations were made for the NSGA-II ZDT specialist problem compared to that of the NSGA-II DTLZ specialist problem.

Table 3: Mean performances on the NSGA-II specialist tuning problems.

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

Notes: Friedman test: χ^2 24.100, p-value 5.8×10^{-6}. Boldface entries indicate the best value in each row. 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.

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

5.2 Specialist Tuning Results

MOTA, tMOPSO, and RAND^M were applied to the specialist and generalist tuning problems in order to generate 20 independent runs for each problem. Analysis of the results from these tuning problems is conducted according to the τ performance metric, where τ measures the HV achieved for a given bi-objective decomposition in the normalized objective space as outlined in Eq. (30).

Comparison according to performance on the NSGA-II specialist tuning problems is conducted quantitatively according to MWUTs, as summarized in Table 3. Two sample means are considered statistically similar if a MWUT indicates that the difference of the sample means is not statistically significant given a confidence level of 95%. On

Table 4: Mean performances on the MOEA/D specialist tuning problems.

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

Notes: Friedman test: χ^2 25.200, p-value 3.4×10^{-6}. Boldface entries indicate the best value in each row. 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.

the NSGA-II ZDT specialist problem, tMOPSO outperforms MOTA on 2/5 subproblems and is outperformed by MOTA on the 1/5 subproblems, and performs statistically similar on the remaining 2/5 subproblems. For the NSGA-II DTLZ specialist problem, tMOPSO beats MOTA on 2/7 subproblems, is outperformed by MOTA on one subproblem, and is statistically similar on the other 4/7 subproblems. Regarding the NSGA-II WFG specialist problem, tMOPSO outperforms MOTA on 1/9 subproblems, is outperformed by MOTA on 2/9 subproblems, and is statistically similar to MOTA on the remaining 6/9 subproblems. RAND^M is outperformed by tMOPSO and MOTA on all the subproblems for the NSGA-II specialist tuning problems, with the exception of the DTLZ$_4$ subproblem, where the difference in sample means is not statistically significant.

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

The τ MWUT comparisons are supported through visual inspection of box plot comparisons, through visual inspection of the PFA determined by the compared tuning algorithms, and through Friedman tests. The box plot comparisons show that there is a clear difference between the two samples, where a MWUT has shown the means to be different given a 95% significance level. Comparing the bi-objective PFA of the subproblems shows that the τ performance metric is adequate for comparing tuning performance in these experiments. The box plot comparisons and PFAs found by the tuning algorithms are available in the supplementary material of this paper. Friedman test p-values, which are also available in Tables 3 and 4, show that the null-hypothesis of there being no difference between tMOPSO, MOTA, and RAND^M can be safely rejected.

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

5.3 Generalist Tuning Problems

On the generalist tuning problems, MOTA's CPV tuple assessment procedure, which utilizes update neighborhoods, had a significant effect. On the NSGA-II generalist problems MOTA outperformed tMOPSO on all the ZDT, DTLZ, and WFG subproblems, as shown in Table 5. RAND^M is more competitive against MOTA on the NSGA-II generalist problems compared to the specialist problems. Specifically, on 6/24 subproblems the difference in sample τ means between RAND^M and MOTA is not statistically significant according to MWUTs given a 95% confidence level. MOTA outperformed RAND^M on the remaining 18/24 NSGA-II generalist subproblems. Regarding the MOEA/D generalist problems MOTA outperformed tMOPSO on all but 1/24 subproblems, as shown in Table 6. RAND^M outperformed MOTA on 1/24 subproblems, while performing worse than MOTA on the remaining 23/24 subproblems. Additional result verification similar to that done for the specialist tuning problems shows that MOTA's superior performance according to the τ performance metric is supported by box plot comparisons, plots of the PFAs found by the tuning algorithms, and Friedman tests. The box plot comparisons and PFAs found for the generalist tuning problems are available in the supplementary material for this paper.

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

Table 5: Mean performances on the NSGA-II generalist tuning problems. The $\mathbf{1}^{i \neq j}$ notation indicates a vector whose elements are all equal to 1 with exception of the j'th element, which is equal to 0.

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

Notes: Friedman test: χ^2 39.000, p-value 3.4×10^{-9}. Boldface entries indicate the best value in each row. 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.

to tMOPSO completely outperforming RAND^M on the specialist tuning problems, further emphasizes the importance of information sharing.

MOTA's results were scrutinized in regard to producing sensible CPV recommendations. Subproblems were added in the generalist tuning problem formulations for all the leave-one-out generalist combinations. Comparing the results from these subproblems against the $\mathbf{w} = [1, 1, \ldots, 1]$ subproblem gives an indication of the validity of MOTA's results. For the NSGA-II problems, a high level of consistency among the generalist subproblems in terms of crossover and mutation probabilities for similar OFE budgets is observed, while the majority of population size CPV recommendations followed a similar trend. In particular, for very low OFE budgets the optimal population is equal to the OFE budget. Thereafter, a drop in the optimal population size is observed, at the OFE budget where the NSGA-II operators become more effective than random search. The optimal population size then continues to increase as the available OFE budget increases. The observed population size trend is consistent with Dymond et al.

Table 6: Mean performances on the MOEA/D generalist tuning problems. The $\mathbf{1}^{i\neq j}$ notation indicates a vector whose elements are all equal to 1 with exception of the j'th element, which is equal to 0.

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

Notes: Friedman test: χ^2 40.333, p-value 1.7×10^{-9}. Boldface entries indicate the best value in each row. 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.

(2013). For the MOEA/D problems, an acceptable level of consistency is also observed, with similar DE scaling factors, crossover probability, and neighborhood fractions being recommended among similar OFE budgets, while similar population size trends as for the NSGA-II generalist problem were observed. Given the levels of consistency being observed, and agreement with previous work, MOTA's results are deemed acceptable.

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

6 Conclusion

The MOTA algorithm is proposed for tuning stochastic optimization algorithms according to multiple utility measures under multiple OFE budgets. MOTA uses many-objective optimization to achieve this end, with an objective for each utility measure and an extra speed objective to tune under multiple OFE budgets. Decomposition is used to solve the resulting many-objective optimization problem, with the original problem being broken up into multiple bi-objective subproblems. When tuning stochastic algorithms, analytical expressions for determining utility as a function of the CPV tuple being assessed at the assessment OFE budget are typically unavailable. Therefore, an assessment run is used to calculate the utility of a CPV tuple for stochastic algorithms, where the algorithm being tuned is run from initialization to the assessment OFE budget. MOTA utilizes the history information from these assessment runs, so that a CPV tuple assessment run is used to gauge utility at multiple OFE budgets. In particular, MOTA's bi-objective decomposition scheme is aligned to make use of this history information. To further increase efficiency, MOTA uses a preemptively terminating resampling approach based on MWUTs for handling noise. Finally, MOTA uses DE-based operators to generate new candidate designs using candidate generation neighborhoods, while using different update neighborhoods to propagate information among the bi-objective subproblems.

Numerical experiments were conducted to gauge MOTA's performance. The numerical experiments entailed tuning NSGA-II and MOEA/D in order to determine specialist and generalist CPVs for the ZDT, DTLZ, and WFG test problem suites, resulting in a total of 12 tuning problems. Using these problems, MOTA was compared against the tMOPSO and the RAND^M tuning algorithms. For fair comparison, MOTA and tMOPSO were compared using CPVs that are well suited for tuning problems used in these numerical experiments. These CPV tuples were determined by tuning MOTA and tMOPSO to the DTLZ specialist problem. For the specialist tuning problems, MOTA is effective at determining specialist CPV tuples over a range of OFE budgets, having a comparable performance to the tMOPSO algorithm. For the many-objective generalist tuning problems, MOTA clearly outperforms the other tuning algorithms. This superior performance is attributed to MOTA's being designed from the ground up as a many-objective tuning algorithm.

Acknowledgments

The South African Centre for High Performance Computing (CHPC) is acknowledged for the computing resources they made available for this research. The financial assistance of the National Research Foundation (NRF) of South Africa toward 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.

References

Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In *Proceedings of International Workshop on Hybrid Metaheuristics*, pp. 108–122. Lecture Notes in Computer Science, Vol. 4771.

Bartz-Beielstein, T., Lasarczyk, C., and Preuss, M. (2005). Sequential parameter optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 773–780.

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

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

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

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

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

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

Conover, W. (1999). *Practical nonparametric statistics*. 3rd ed. New York: Wiley.

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

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

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

Deb, K., and Srinivasan, A. (2006). Innovization: Innovating design principles through optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 1629–1636.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In A. Abraham and R. Goldberg (Eds.), *Evolutionary multiobjective optimization*, pp. 105–145. London: Springer.

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

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

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

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

Dymond, A. S., Engelbrecht, A. P., Kok., S., and Heyns, P. S. (2015). Tuning optimization algorithms under multiple objective function evaluation budgets. *IEEE Transactions on Evolutionary Computation*, 19(3): 341–358.

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

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

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

Eiben, A. E., and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1): 19–31.

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. New York: Wiley.

François, O., and Lavergne, C. (2001). Design of evolutionary algorithms: A statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2): 129–148.

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

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

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

Mostaghim, S., and Teich, J. (2005). Quad-trees: A data structure for storing Pareto sets in multiobjective evolutionary algorithms with elitism. In A. Abraham and R. Goldberg (Eds.), *Evolutionary multiobjective optimization*, pp. 81–104. London: Springer.

Nannen, V., and Eiben, A. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the International Joint Conference on Artifical intelligence*, pp. 975–980.

Pedersen, M. (2010). Tuning and simplifying heuristical optimization. Unpublished doctoral dissertation, School of Engineering Sciences, Computational Engineering and Design Group, University of Southampton, UK.

Salehinejad, H., Rahnamayan, S., Tizhoosh, H., and Chen, S. (2014). Micro-differential evolution with vectorized random mutation factor. In *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2055–2062.

Saxena, D. K., Duro, J. A., Tiwari, A., Deb, K., and Zhang, Q. (2013). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1): 77–99.

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

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

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

Smit, S. K., Eiben, A. E., and Szlávik, Z. (2010). An MOEA-based method to tune EA parameters on multiple objective functions. In *Proceedings of the International Joint Conference on Computational Intelligence*, pp. 261–268.

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

Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., and Tiwari, S. (2005). *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. KanGAL Report 2005005, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, India.

Ugolotti, R., and Cagnoni, S. (2014). Analysis of evolutionary algorithms using multi-objective parameter tuning. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 1343–1350.

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

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

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

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

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

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

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. In *Proceedings of International Conference on Evolutionary and Deterministic Methods for Design, Optimization, and Control with Applications to Industrial Problems*, pp. 95–100.

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