Balancing Constraints and Objectives by Considering Problem Types in Constrained Multi-objective Optimization

Yi Xiang, Xiaowei Yang, Han Huang, Senior Member, IEEE and Jiahai Wang Senior Member, IEEE

Abstract—Constrained multi-objective optimization problems widely exist in real-world applications. To handle them, the balance between constraints and objectives is crucial, but remains challenging due to non-negligible impacts of problem types. In our context, the problem types refer particularly to those determined by the relationship between the constrained Pareto-optimal front (PF) and the unconstrained PF. Unfortunately, there has been little awareness on how to achieve this balance when faced with different types of problems. In this paper, we propose a new constraint handling technique (CHT) by taking into account potential problem types. Specifically, inspired by prior work, problems are classified into three primary types: I, II and III, with the constrained PF being made up of the entire, part and none of the unconstrained counterpart, respectively. Clearly, any problem must be one of the three types. For each possible type, there exists a tailored mechanism being used to handle the relationships between constraints and objectives (i.e., constraint priority, objective priority or the switch between them). It is worth mentioning that exact problem types are not required because versions of one or more of them are known, and a real-world problem (with unknown types) in search-based software engineering. Results demonstrate that, within both decomposition-based and non-decomposition-based frameworks, the new CHT can indeed achieve a good trade-off among different problem types, being better than several state-of-the-art CHTs.

Index Terms—Constrained multi-objective optimization; constraint handling techniques; trade-off model; problem types

I. INTRODUCTION

CONstrained multi-objective optimization problems (CMOPs) are common in real-world applications, such as the optimal scheduling of energy storage systems [1], the optimization of biped robot gaits [2] and the reliability-based optimization for crashworthy structures [3]. Without loss of generality, a CMOP can be formulated as follows.

\[
\begin{align*}
\text{Min } & \quad F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\
\text{s.t. } & \quad g_i(x) \leq 0, i = 1, \ldots, p \\
& \quad h_j(x) = 0, j = 1, \ldots, q \\
& \quad x \in \Omega \subset \mathbb{R}^n,
\end{align*}
\]

where \(F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \in \mathbb{R}^m\) and \(x = (x_1, x_2, \ldots, x_n)^T \in \Omega\) (the decision space) are the objective vector and decision vector, respectively. In (1), \(g_i(x) \leq 0\) defines the \(i\)-th of \(p\) inequality constraints, and \(h_j(x) = 0\) defines the \(j\)-th of \(q\) equality constraints. According to [4], CMOPs with more than three objectives are known as constrained many-objective optimization problems (CMAOPs).

From the perspective of decision makers, feasible solutions are naturally emphasized over infeasible ones. A solution to CMOP (1) is feasible provided that it meets all the equality and inequality constraints. If any constraint is violated, this solution is infeasible. To distinguish between solutions, one applicable and often-used principle is Pareto-dominance. \(x_1\) is said to Pareto-dominate (or simply dominate) \(x_2\) as \(x_1 < x_2\) if and only if \(f_i(x_1) \leq f_i(x_2)\) for all \(i = 1, \ldots, m\) and \(f_j(x_1) < f_j(x_2)\) for at least one \(j \in \{1, \ldots, m\}\). For a solution \(x^*\), if there exist no other solutions dominating it, then \(x^*\) is called a Pareto-optimal solution. All the Pareto-optimal solutions constitute the Pareto-optimal set (PS), and the Pareto-optimal front (PF) is defined as the image of PS in the objective space.

According to [5], CMOPs can be classified based on the relationships between constrained PF (CPF) and unconstrained PF (UPF). As shown in Fig. 1, CMOPs are divided into three types. For Type I, as seen from Fig. 1 (a), CPF is exactly the same as UPF. For Type II, CPF is or contains part(s) of the UPF. As illustrated in Fig. 1 (b) and (c), CPF consists of part(s) of the unconstrained front, while in Fig. 1 (d) CPF contains not only a part of the UPF but also a part of the feasible boundary\(^1\). Finally, as illustrated in Fig. 1 (e), CPF

\(^1\)In [5], the type illustrated in Fig. 1 (d) is classified independently. In our work, however, this type is combined with those illustrated in Fig. 1 (b) and (c) to form Type II.
for Type III has no intersection with the UPF. Mathematically, if the intersection of CPF and UPF is , then \( |A| = |UPF| \), \( 0 < |A| < |UPF| \) and \( |A| = 0 \) hold for Types I, II and III, respectively. Here \( |\cdot| \) denotes the cardinality of a set. Moreover, each type of CMOPs may have infeasible regions blocking the way of converging towards the CPF \([4], [6]\). To emphasize this feature, CMOPs with infeasible barriers are called Type-I’, Type-II’ and Type-III’ problems in this paper.

Generally speaking, CMOPs are much more difficult than their unconstrained counterparts, due to that, in addition to balancing convergence and diversity, how to achieve a balance between constraints and objectives should also be properly addressed. In fact, most of existing constraint handling techniques (CHTs) are designed to attempt to achieve this balance \([7]\). These CHTs include constraint-domination principle (CDP) \([8]\), penalty functions \([9]\), stochastic ranking (SR) \([10]\), \( \varepsilon \) constrained method \([11]\) (called \( \varepsilon \) for brevity), methods based on multi-objective optimization \([12]\), and hybrid methods \([13]\). Although some of the above CHTs were originally proposed for constrained single-objective optimization problems (CSOPs), they have been integrated into multi-objective evolutionary algorithms (MOEAs) to handle CMOPs as well, leading to some representative constrained multi-objective evolutionary algorithms (CMOEsAs). Like the taxonomy adopted in the unconstrained scenario \([14]–[16]\), CMOEsAs can be divided into three mainstreams according to the used selection mechanism, i.e., Pareto-dominance-based CMOEsAs \( (e.g., [8], [17]–[20]) \), decomposition-based CMOEsAs \( (e.g., [7], [21]–[26]) \) and indicator-based CMOEsAs \( (e.g., [27]) \).

Recently, Ma and Wang \([5]\) have comprehensively compared several representative CHTs, e.g., CDP, SR and \( \varepsilon \), under the frameworks of both NSGA-II \([8]\) and MOEA/D \([28]\). Their results suggest that the performance of existing CHTs is type-dependent—different CHTs have their own strengths and weaknesses on different types of CMOPs \([5]\). Therefore, to select or design the most suitable CHT in a specific scenario, one might need to have some prior knowledge on the type of the problem at hand. In real-world applications, however, this may not be always possible because the type of a problem is often not known \( a priori \). Given that any CMOP must be one of the three types according to our taxonomy, it makes sense to develop a CHT that considers all possibilities in advance, and, for each of them, tailors a mechanism to handle relationships between objectives and constraints. This way, the CHT is expected to achieve a trade-off among different types of problems.

In fact, objectives and constraints should be emphasized differently for different types of problems \([5], [29]\). Intuitively, as shown in Fig. 1, objective priority is suitable for Type-I problems. To handle this type of CMOPs, what we need to do is to drive solutions towards UPF by optimizing objectives only. In addition, for Type-I’, Type-II’ and Type-III’ problems, emphases on objectives can help the working population get across obstacles caused by infeasible regions \([6], [7]\). For Type-III problems, the CPF is completely made up of boundary of the feasible region. As explicitly mentioned in \([5]\), constraint priority may well fit this type of problems because it enables a good exploitation around the feasible boundary. As for Type II, the switch between objective priority and constraint priority is desired \([5]\), yet remains challenging.

With the above in mind, we suggest a Trade-off Model (ToM) to balance the performance on different types of problems. Specifically, ToM explicitly divides the evolutionary process into two stages. In the first stage, objective priority is implemented (tailored for Type-I problems, and those with infeasible barriers), while in the second stage, constraint priority is executed (tailored for Type III). The balance between objectives and constraints is realized by the automatic switch between objective priority and constraint priority. As discussed previously, this switch caters to Type-II problems. For this type, according to Fig. 1, we need to find some part(s) of the UPF, and a portion of the feasible boundary. Clearly, ToM takes into account potential problem types, but this does not mean that we need to exactly know the types. Instead, we just consider the possibility of each type, and tailor a handling mechanism, i.e., objective priority, constraint priority and the switch between them. Since all the mechanisms are integrated in ToM, there must be one mechanism that fits the true type of the problem at hand. Requiring no prior knowledge on true types of problems is a good property, which makes it possible to apply ToM to real-world applications. Moreover, since the (potential) problem type is a common (rather a particular) property of CMOPs, ToM is less likely to cause overfitting issues. Results in Section IV show that ToM indeed obtains satisfying performance on a range of problems, both artificial and practical.

The proposed ToM can be easily integrated into both decomposition-based frameworks (e.g., MOEA/D \([28]\)) and
The remainder of the paper is organized as follows. Section II reviews related work on CHTs and corresponding CMOEAs. In Section III, we give details on the proposed ToM framework and its concrete implementations. Section IV presents experimental study, followed by some important discussions given in Section V. Finally, Section VI concludes the paper and outlines possible directions for future studies.

II. RELATED WORK

The CDP, first proposed in the NSGA-II paper [8], and also adopted in C-NSGA-III [17] and the MOEA/D framework [23], may be the simplest and most widely used CHT for CMOPs. This principle consistently prefers feasible solutions to infeasible ones. Such a preference drives the population to feasibility before improving the objectives [18], and hence may encounter difficulties in getting across obstacles caused by infeasible regions. Moreover, empirical results in [32] showed that CDP may also struggle on CMOPs with very small and narrow feasible regions.

Another common approach to handle constraints is to use penalties. Penalty-function-based techniques normally convert CMOPs into unconstrained MOPs by adding a penalty term to the objectives [7], [9]. In these techniques, the penalty factor plays an important role in maintaining the balance between minimizing the objectives and satisfying the constraints. However, the tuning of this factor is not always easy because its value is often problem-dependent [5]. This limitation promotes researchers to develop self-adaptive penalty (SP) functions [33], [34], in which penalty factors are adaptively adjusted on the basis of the feedback taken from the search process. To handle CMOPs, Woldesenbet et al. [20] proposed to modify each objective by using a distance measure, and an SP function in which two penalties are involved. The first one is based on objective functions and the second is based on the constraint violation. The balance between the two penalties is adaptively controlled by the ratio of feasible individuals presented in the current population.

In SR [10], for any two solutions, they are compared according to either the objective function with the probability $P_f$ (a user-defined parameter), or the constraint violations with the probability $1 - P_f$. Although SR was originally proposed for CSOPs, it can be applied to handle CMOPs as well. In [35], a new CHT with infeasible elitists preservation and SR-based selection is proposed to handle constraints in multi-objective optimization. Given that Pareto dominance may lead to incomparable solutions, the new method assigns each solution a fitness value based on its ranking in the nondominated sorting and its crowding distance. In [23], an extended version of SR was implemented for the first time in the MOEA/D framework. To solve CMOPs effectively, Ying et al. [24] proposed an adaptive SR mechanism in MOEA/D. In this mechanism, the probability $P_f$ is adaptively adjusted by borrowing the idea of metropolis acceptance criterion.

In $\varepsilon$ constrained method [11], constraint violations are relaxed by the $\varepsilon$-level. Solutions with constraint violations less than $\varepsilon$ are treated as feasible ones, then they are compared...
according to objective functions. In [7], [21], [22], [36], this CHT was adopted in the framework of MOEA/D to deal with CMOPs. Since MOPs in MOEA/D are transformed into a number of scalar sub-problems using weight vectors, the \( \varepsilon \) method can be directly applied. However, finding an optimal (or near-optimal) value for \( \varepsilon \) is not at all trivial. Therefore, to balance between constraints and objectives, the \( \varepsilon \) level in most cases is set dynamically during the evolutionary process [7], [22], [36].

By converting constraints into one or more extra objectives, a constrained problem is reformulated as an unconstrained one, which can be directly solved by MOEAs [37]. Typical work using this method includes Cai and Wang's method [12], the infeasibility driven evolutionary algorithm [18], a general framework of dynamic CMOEOs for constrained optimization [38], and the tri-goal evolution framework for CMAOPs [39], etc. Inversely, a CMOP can be handled by first transforming a CMOP into a CSOP to find promising feasible areas, and then by applying a specific CMOEA to obtain final solutions [40].

Constraints can be handled by combining several popular CHTs together in either different evolutionary stages or different sub-populations. For example, the adaptive tradeoff model (ATM), proposed by Wang et al. [13], divides the search process into three scenarios. If solutions in the current population are all infeasible, a multi-objective approach is adopted to deal with constraints. If the population contains both feasible and infeasible solutions, ATM uses a penalty function to select solutions for the next generation. In case of all feasible solutions, the comparison is performed based on entirely objective values. Despite that ATM was originally proposed for CSOPs, Ma et al. [5] have recently investigated its performance on CMOPs, showing that ATM could obtain competitive results as well, particularly on those with narrow feasible regions. The ensemble of constraint handling methods (ECHM), suggested by Qu et al. [41], was used to tackle CMOPs. In ECHM, three CHTs are involved, including an SP function [20], the superiority of feasible solutions [42] and the \( \varepsilon \) constrained method [11]. Note that ECHM uses three sub-populations, each of which employs a different CHT.

Exploiting information carried by infeasible solutions can be sometimes effective in handling CMOPs. For example, in both IDEA [18] and C-VaEA [19], a small percentage of infeasible solutions are retained to exploit information carried by them. This allows the algorithms to approach the constrained boundaries from both the feasible and infeasible sides of the search space. In PPS-MOEAD/D [7], the search process is divided into push and pull stages. In the push stage, constraints are entirely ignored, which allows the existence of infeasible solutions. In this way, the population is pushed toward UPF as closely as possible. In the pull stage, the population is pulled back to CPF using an improved \( \varepsilon \) constraint-handling approach. Similarly, CMOEA-MS [29] is also an algorithm based on two stages, in which the priority of objectives and constraints is automatically adjusted based on the status of the current population. According to [29], this algorithm is particularly effective in tackling problems with complex feasible regions. In C-TAEA [6], two collaborative archives are maintained simultaneously. The convergence-oriented archive (CA) is the driving force to push the population toward the PF, and the diversity-oriented archive (DA) focuses on maintaining the population diversity. One of the main characteristics of the update mechanism for DA is that feasibility of solutions is not taken into consideration.

In fact, different CHTs have their own strengths and weaknesses in different scenarios [5], [29]. For example, CDP [8] can concentrate on the search on the boundary of the feasible region, but may be quite ineffective in getting cross infeasible barriers. The \( \varepsilon \) enables a switch from objectives to constraints, but this requires a proper control of the \( \varepsilon \)-level [7]. In addition, \( \varepsilon \) may struggle on some Type-III problems with complicated feasible boundary [5]. Including infeasible solutions as done in [18], [19] can help the population to converge towards CPF from both feasible and infeasible sides, but this may not make sense when the included infeasible solutions are far away from CPF. The CHT in PPS-MOEAD/D [7] ignores constraints in the first stage so as to push the population toward UPF as closely as possible. This way, the CHT is able to easily overcome infeasible obstacles. For some problems whose feasible regions are disjoint from the UPF, e.g., MW13, the convergence of the population obtained by this CHT may be poor [29]. The goal of existing CHTs is to seek a balance between constraints and objectives. To our best knowledge, however, there has been little awareness on how to achieve this goal when faced with different types of CMOPs. Indeed, as explicitly demonstrated in [5], the performance of most existing CHTs varies on different types of problems. This motivates us to propose a trade-off model by taking into account potential types of CMOPs. Conceptually, as discussed in Section I, a prominent advantage of the proposed new CHT over existing ones is that it can achieve a trade-off on different problem types. The above argument will be confirmed by comprehensive experimental studies to be performed in Section IV.

III. THE PROPOSED TO M AND ITS IMPLEMENTATIONS

The basic design principle of ToM is to achieve a trade-off between constraints and objectives considering potential problem types. The general framework of ToM is outlined in Algorithm 1. As seen, ToM has three control parameters: \( N \), \( G \) and \( \alpha \). The first two parameters are common in MOEAs: they specify the population size and the number of maximum generations, respectively. The last one is important, and unique to ToM: it determines the switch between objective priority and constraint priority. The effect of this parameter will be experimentally investigated in Section V-C.

According to Line 1 of Algorithm 1, ToM starts with initialization of two populations, \( P = \{x_1, \ldots, x_N\} \) and \( P' = \{x'_1, \ldots, x'_N\} \). The former, as in a general MOEA, maintains solutions at the current generation, while \( P' \) simply records the best feasible solutions found during the search process. Commonly, \( P \) is initialized with \( N \) randomly generated solutions, and \( P' \) is subsequently initialized with feasible solutions in \( P \) (if any). The next main procedures of this framework consist of evolving/updating populations \( P \) and
Algorithm 1 The proposed ToM

Input: population size ($N$), number of maximum generations ($G$), and the parameter determining the switch between objective priority and constraint priority ($\alpha$)

Output: Final population

1: Initialize populations $P$ and $P'$ /* $P$ consists of $N$ random solutions, while $P'$ stores all the initial feasible solutions (if any)*/
2: $\text{switch} \leftarrow \text{false}$ // The variable $\text{switch}$ specifies whether objective priority is switched to constraint priority
3: $q \leftarrow 1$ // The number of current generation
4: $\text{SIG} \leftarrow 0$ // The number of successive infeasible generations
5: while $q \leq G$ do
6: if all members in $P$ are infeasible then
7: $\text{SIG}++$
8: else
9: $\text{SIG} \leftarrow 0$
10: end if
11: $\{P, P'\} \leftarrow \text{Evolution}(P, P', \text{switch})$ /* Evolve $P$ using any MOEA framework (see Algorithm 2). During this process, $P'$ is updated accordingly. */
12: if $\neg \text{switch}$ then
13: if ($q \geq 0.5G$ || $\text{SIG} \geq \alpha G$) then
14: $\text{switch} \leftarrow \text{true}$
15: type $\leftarrow \text{EstimateType}(P, P')$ // Estimate types of the problem
16: $P \leftarrow \text{Reconstruct}(P, P', \text{type})$ // Reconstruct $P$ according to estimated problem types
17: end if
18: end if
19: $q++$
20: end while
21: Perform constrained non-dominated sorting [17] to $P \cup P'$, and identify the first layer $L_1$
22: return $L_1$

$P'$ (Line 11), estimating types of problems (Line 15) and reconstructing $P$ (Line 16).

A. Evolution of populations

The population $P$ can be evolved using any MOEA framework, e.g., MOEA/D [28] and NSGA-II [8]. Algorithm 2 gives an abstract prototype, in which the boolean variable $\text{switch}$ specifies whether or not objective priority has been switched to constraint priority. If $\text{switch}$ is false, then objective priority is awaked. That is, objectives are the only criterion to compare individuals in $P$. More specifically, the solution $x_i \in P$ will be replaced by a new solution $y_i$ (generated from $x_i$) in case that $y_i$ outperforms $x_i$ concerning comparisons with respect to objectives. If $\text{switch}$ is true, then comparisons between new and old solutions are made considering first constraints, and second objectives. Only when solutions are incomparable in terms of constraint violation (CV) [8], [43], objectives are used as a secondary criterion to determine which solution survives. The CV was widely used to measure the degree of constraints violated by a solution. Formally, it is given as follows.

$$CV(x) = \sum_{i=1}^{p} \max\{g_i(x), 0\} + \sum_{j=1}^{q} \max\{|h_j(x)| - \phi, 0\},$$

where $\phi$ is a very small positive value (e.g., $10^{-4}$) [5] adopted to relax equality constraints. Since exactly satisfying equality constraints is very difficult or even impossible for evolutionary algorithms due to their stochastic characteristics, a common way of handling this issue is to relax equality constraints using a small positive value. Apparently, the CV value is equal to 0 for feasible solutions, and greater than 0 for infeasible ones.

Algorithm 2 $\{P, P'\} \leftarrow \text{Evolution}(P, P', \text{switch})$

Input: $P, P'$ and $\text{switch}$

Output: updated $P$ and $P'$

1: if $\neg \text{switch}$ then
2: Objective priority: Evolve $P$ considering only objectives
3: else
4: Constraint priority: Evolve $P$ considering first constraints, and second objectives
5: end if
6: Update $P'$ accordingly
7: return $\{P, P'\}$

During the evolution of $P$, as shown in Line 6 of Algorithm 2, the feasible population $P'$ should be updated accordingly. Specifically, if new solutions produced during the evolution of $P$ are feasible, and better than solutions in $P'$ (in terms of objectives), then they will be replaced by those new solutions. It is worth mentioning that Algorithm 2 is highly abstract, and concrete implementations can be integrated in both MOEA/D and NSGA-II, leading to MOEA/D-ToM and NSGA-II-ToM, respectively.

For instance, the evolution in MOEA/D-ToM is presented in Algorithm 3. As in the original MOEA/D, a CMOP is decomposed into $N$ subproblems using $N$ weight vectors $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$. For each subproblem $i$, the algorithm maintains two solutions, $x_i \in P$ and $x'_i \in P'$, representing the best solution and the best feasible solution, respectively. In case that feasible solutions are not available for this subproblem, $x'_i$ is marked with NULL. Lines 2-5 in Algorithm 3 are routine procedures in the original MOEA/D. Specifically, $B(i)$ in Line 2 denotes the neighborhood of the $i$-th subproblem, and $\delta$ is a control parameter for which 0.9 is recommended [44]. The set $D$ is set to either $B(i)$ (with probability $\delta$) or $\{1, \ldots, N\}$ (with probability $1 - \delta$). According to Lines 3 and 4, for each subproblem $i$, a new solution $y_i$ is generated and evaluated, and will be used to update subproblems (Lines 8-16).

- In the objective priority phase (i.e., $\neg \text{switch}$), $\text{ObjectivePriorComparator}$ (Algorithm 4) is adopted to determine whether $y_i$ should replace $x_i$ or not. As seen, the replacement is permitted if and only if $y_i$ dominates $x'_q$, or $y_i$ and $x_q$ are non-dominated, but $y_i$ has a better scalarization function value than $x_q$. In Line...
Algorithm 3 $\{P, P'\} \leftarrow \text{MOEA/D Evo}(P, P', \text{switch})$

Input: $P = \{x_1, \ldots, x_N\}, P' = \{x'_1, \ldots, x'_N\}$ and switch
Output: updated $P$ and $P'$
1: for each subproblem $i \in \{1, \ldots, N\}$ do
2:  $D = \left\{ \begin{array}{ll}
B(i), & \text{if rand}() < \delta \text{ (a parameter)} \\
\{1, \ldots, N\}, & \text{otherwise}
\end{array} \right.$
3: Randomly select two different parents from $D$, then apply crossover and mutation operators to generate a new solution $y$
4: Compute objectives and evaluate constraints for $y$
5: $t \leftarrow 0$
6: for each $q \in D$ do
7:  // Evolve $P'$
8:  if $\neg \text{switch}$ then
9:   if ObjectivePriorComparator($x_q, y$) then
10:     $x_q \leftarrow y$; $t \leftarrow t + 1$
11:   end if
12:  else if ConstraintPriorComparator($x_q, y$) then
13:     $x_q \leftarrow y$; $t \leftarrow t + 1$
14:  end if
15: end if
16:  if $CV(y) = 0$ then
17:    if $x'_q = \text{NULL}$ then
18:      $x'_q \leftarrow y$
19:    else if ObjectivePriorComparator($x'_q, y$) then
20:      $x'_q \leftarrow y$
21:    end if
22:  end if
23:  if $t \geq n_r$, then break // $n_r$ is a control parameter
24: end for
25: return $\{P, P'\}$

5 of Algorithm 4, $g^*$ denotes any scalarization function, which can be the weighted sum, Tchebycheff or penalty-based boundary intersection (PBI) [28]. In this paper, we use PBI, defined as follows [28].

$$g^{\text{pbi}}(y, \lambda_q) = d_1 + \theta d_2, \quad (3)$$

where $\theta$ is a user-specified penalty parameter; $d_1$ and $d_2$ are given as

$$d_1 = \frac{(F(y) - z^*)^T \lambda_q}{||\lambda_q||}, \quad d_2 = \left\| F(y) - (z^* + d_1 \frac{\lambda_q}{||\lambda_q||}) \right\|. \quad (4)$$

Here $z^* = (z_1^*, \ldots, z_m^*)$ is an estimation to the ideal point, and each of its component $z_i^*$ represents the best objective value for the $i$-th objective found during the search.

- In constraint priority phase, $y$ and $x_q$ are compared based on ConstraintPriorComparator (Algorithm 5) that compares two solutions considering first constraints, and second objectives. When comparing solutions based on objectives, Algorithm 5 follows the same procedures as in Algorithm 4.

Algorithm 4 flag $\leftarrow$ ObjectivePriorComparator($x_q, y$)

Input: $x_q, y$
Output: flag
1: flag $\leftarrow$ $false$
2: if ($y \prec x_q$) then
3:   flag $\leftarrow$ $true$
4: else if ($y \not\prec x_q$) $\land$ ($x_q \not\prec y$) then
5:   if $g^*(y, \lambda_q) < g^*(x_q, \lambda_q)$ then
6:     flag $\leftarrow$ $true$
7: end if
8: end if
9: return flag

According to Line 25 in Algorithm 3, the number of replaced solutions is limited by $n_r$, which is usually set to $0.1 T$ where $T$ is the neighbor size, being usually $0.1 N$). It should be mentioned that, to introduce randomness, we do not follow a fixed order (from the first element to the last one) when scanning the set $D$ in Line 6 of Algorithm 3. Instead, members in $D$ are visited in a random way such that each one has an equal chance of being chosen.

Algorithm 5 flag $\leftarrow$ ConstraintPriorComparator($x_q, y$)

Input: $x_q, y$
Output: flag
1: if $CV(y) < CV(x_q)$ then
2:   flag $\leftarrow$ $true$
3: else if $CV(y) = CV(x_q)$ then
4:   if ($y \prec x_q$) then
5:     flag $\leftarrow$ $true$
6: else if ($y \not\prec x_q$) $\land$ ($x_q \not\prec y$) then
7:   if $g^*(y, \lambda_q) < g^*(x_q, \lambda_q)$ then
8:     flag $\leftarrow$ $true$
9: end if
10: end if
11: end if
12: return flag

The new solution $y$ is also utilized to update the feasible population $P'$ (Line 18-24 in Algorithm 3). The prerequisite is that $y$ should be feasible (i.e., $CV(y) = 0$). Further, $y$ is allowed to substitute $x'_q$, if $x'_q$ is NULL; or $x'_q$ is not NULL and $y$ performs better than $x'_q$ concerning ObjectivePriorComparator.

B. Switch between objective priority and constraint priority

The switch between objective priority and constraint priority is realized by Line 13 of Algorithm 1. Specifically, objective priority is switched to constraint priority if $g \geq 0.5 G$ or $SIG \geq \alpha G$, where $g$ is the number of current generations; $SIG$ is the number of successive infeasible generations, and $\alpha$ is a control parameter. According to Lines 6-10 in Algorithm...
1. \(SIG\) is increased by 1 in case that all the members in \(P\) are infeasible; otherwise, \(SIG\) is reset to 0.

![Diagram showing Feasible region and Feasible solution](image)

**Fig. 2.** Switching in advance between objective priority and constraint priority is beneficial to Type-III problems

Using the first condition, i.e., \(g \geq 0.5G\), implies that the first half of computational resources are allocated to objective priority, while the second half are allocated to constraint priority. According to our experimental results (to be shown and discussed in Section V-A), this simple switch strategy can be effective for most of the Type-I, -II, -I′, -II′ problems. For some Type-III problems, however, it does not perform well enough. In fact, this is not incidental. As illustrated in Fig. 2, there may be a large gap between UPF and CPF for Type-III problems. Emphasizing more on objective priority will pull the population \(P\) closer to UPF; but this may hamper the exploitation around CPF. In this case, it would be better to switch in advance to avoid wasting computational resources.

Therefore, we introduce the second condition, i.e., \(SIG \geq \alpha G\). This condition allows the algorithm exploring infeasible regions for at most \(\alpha G\) generations. In case that the population cannot pass through infeasible regions within \(\alpha G\) generations, objective priority is switched in advance to constraint priority. On the one hand, as discussed before, this is useful to avoid wasting computational resources for some Type-III problems. On the other hand, by allowing the exploration of infeasible regions, it could also be able to properly handle Type-I/-II problems with infeasible barriers. Nevertheless, this requires to properly tune the parameter \(\alpha\). In Section V-A, we will experimentally demonstrate the benefits brought by using the two conditions. Moreover, in Section V-C, we intend to empirically tune \(\alpha\) by examining some typical values.

C. Estimate types

According to definitions given in Section I, knowing precisely the type of the problem at hand is difficult or even impossible, because this requires to have priori knowledge on the problem’s exact UPF and CPF (which are not available in practical applications). Nevertheless, we can roughly estimate the type by viewing the populations \(P\) and \(P′\) as approximations to UPF and CPF, respectively. With this in mind, we provide in Algorithm 6 a simple method for the estimation of problem types. As shown in Line 1, the intersection of \(P\) and \(P′\) (i.e., \(A\)) should be worked out first; then its size (i.e., \(|A|\)) is used to estimate types. Specifically, if the size is equal to the population size \(N\), then the problem is likely to be Type I. In case that \(|A| = 0\), the problem is estimated to be Type III. Otherwise, the problem may be Type II. Notice that, since both UPF and CPF are approximated, this estimation method offers no guarantees on 100% correctness. According to the results presented in Section V-B, however, the estimation accuracy is reasonably high for most of the problems under investigation.

### Algorithm 6: \(\text{type} \leftarrow \text{EstimateType}(P, P′)\)

**Input:** population \(P\) and feasible population \(P′\)

**Output:** estimated type of the problem

1. \(A \leftarrow P \cap P′\) // intersection of \(P\) and \(P′\)
2. \(\text{type} \leftarrow \text{NULL}\)
3. if \(|A| = N\) then  
   4. \(\text{type} \leftarrow \text{I}\)
5. else if \(|A| = 0\) then  
   6. \(\text{type} \leftarrow \text{III}\)
7. else
   8. \(\text{type} \leftarrow \text{II}\)
9. end if
10. return type

It is worth mentioning that the estimation is performed after objective priority is switched to constraint priority, and that the estimated types are adopted to guide the subsequent reconstruction of the population \(P\) (Line 16 in Algorithm 1). In the next subsection, we will depict how \(P\) is reconstructed.

D. Reconstruct population

The reconstructed population serves as the initial population in the constraint priority phrase. The general framework of reconstruction is given in Algorithm 7. If the estimated type is I, nothing should be do in this case. In other words, the population \(P\) is retained. For Type-II problems, we first include all the intersection of \(P\) and \(P′\), and then randomly select individuals from either \(P\) or \(P′\). As for Type-III problems, we use individuals in \(P′\) to replace those in \(P\). Notice that the number of individuals in \(P′\) may not be \(N\). In this case, the reconstructed \(P\) contains all members in \(P′\), and also some individuals in \(P\). In the extreme case in which no feasible solution is found in \(P′\), then there is no need of reconstructing \(P\).

### Algorithm 7: \(P \leftarrow \text{Reconstruct}(P, P′, \text{type})\)

1. if \(\text{type} = \text{I}\) then
2. Do nothing
3. else if \(\text{type} = \text{II}\) then
4. \(\mathcal{P} \leftarrow P \cap P′\) // \(\mathcal{P}\) is a temporary population
5. Fill \(\mathcal{P}\) with individuals coming from either \(P\) or \(P′\)
6. \(P \leftarrow \mathcal{P}\)
7. else
8. Use individuals in \(P′\) to replace those in \(P\)
9. end if
10. return \(P\)

For a better illustration, the reconstruction in MOEA/D-ToM is presented in Algorithm 8. As seen, for Type II, \(x_i\)
is replaced by \( x'_i \), if \( x_i \) is infeasible, and \( x'_i \) is not NULL. In addition, the condition \( r < 0.5 \) is used to introduce randomness. As a result, the reconstructed \( P \) consists of both feasible solutions from \( P' \), and infeasible solutions from \( P \). The rationale behind this is to make use of information carried by infeasible solutions. For Type III, \( x_i \) is replaced by \( x'_i \) as long as \( x'_i \) is not NULL. This way, emphases are put on feasible solutions in \( P' \).

**Algorithm 8** \( P \leftarrow \text{MOEA/D}_\text{Reconstruct}(P, P', \text{type}) \)

```plaintext
1: if \( \text{type} = 1 \) then
2: \quad Do nothing
3: else if \( \text{type} = 11 \) then
4: \quad for each \( x'_i \in P' \) do
5: \quad \quad if \( CV(x_i) > 0 \) and \( x'_i \neq \text{NULL} \) and \( r < 0.5 \) then
6: \quad \quad \quad \( x_i \leftarrow x'_i \)
7: \quad end if
8: \quad end for
9: else
10: \quad for each \( x'_i \in P' \) do
11: \quad \quad if \( x'_i \neq \text{NULL} \) then
12: \quad \quad \quad \( x_i \leftarrow x'_i \) // Use individuals in \( P' \) to replace those in \( P \)
13: \quad end if
14: end for
15: end if
16: return \( P \)
```

As mentioned previously, ToM can also be integrated in NSGA-II. Due to space issues, complete pseudocode for NSGA-II-ToM is not provided here, but in Section S-I of the online supplementary materials.

**IV. EMPIRICAL STUDY**

Our empirical study contains two parts. In the first part, we compare ToM with several state-of-the-art CHTs in both MOEA/D and NSGA-II frameworks. We expect that the proposed ToM would perform better than these CHTs regarding the overall performance. In the second part, the resulting algorithms MOEA/D-ToM and NSGA-II-ToM are compared with a number of popular CMOEs, including PPS-MOEA/D [7], C-NSGA-III [17], C-MOEA/D [17], C-MOEA/DD [25] and C-TAEA [6]. We anticipate that, equipped with ToM, the basic MOEA/D and NSGA-II could perform better than or at least competitively to those tailored algorithms. All CHTs and all CMOEs except C-TAEA\(^2\) are implemented by Java, and source codes are publicly available\(^3\). Experiments are conducted on a PC computer equipped with an Intel(R) Core(TM)i7-7700 CPU@3.60GHz and 8GB of RAM running Window 10 Enterprise G.

**A. Experiment setup**

In this section, we give a brief introduction to the CHTs under examination, the used test suites, performance metrics and parameter settings.

\(^2\)C-TAEA [6] was originally implemented by C. We use the code provided by the authors to perform experiments.

\(^3\)https://github.com/gzhuxiangyi/ToM

1) **CHTs under examination:** In addition to ToM, four well-known CHTs are integrated in both MOEA/D and NSGA-II. Specifically, under the MOEA/D framework, CDP [23], SP [45], SR [23], and \( \varepsilon \) [7]\(^4\) are implemented. Similarly, the four CHTs are also integrated into NSGA-II, following the practice in [8], [20], [35] and [46], respectively.

2) **Test suites:** To perform empirical evaluations, we choose four suites of CMOPs/CMAOPs, i.e., MW test problems proposed by Ma and Wang [5], DC-DTLZ test problems proposed by Li et al. [6], DAS-CMOPs/DAS-CMAOPs test problems proposed by Fan et al. [4], and the real-world constrained OSPS problem proposed by Xiang et al. [30]. These test suites contain 39 problems with 161 test instances, covering both CMOPs and CMAOPs, both artificial and real-world problems, and both continuous and discrete problems. In addition to problems with less than 100 decision variables, there are also problems involving large-scale decision variables. Diverse categories make these problems quite suitable to evaluate algorithms’ performance. Details about these test suites can be found in Section S-II-A of the online supplement.

3) **Performance metrics:** Each algorithm is independently run 100 times in each test problem-instance, and medians of performance metrics are used to compare algorithms. The performance metrics used in this study are HV [47] and IGD+ [48]. Notice that we choose IGD+, instead of IGD [49], because IGD+ is weakly Pareto compliant [48]. In fact, according to [50], the use of the above two performance metrics should be encouraged as both are able to simultaneously measure convergence and diversity. A larger HV value indicates a better solution set. In contrast, a small IGD+ value is desired. In the calculation of both HV and IGD+, only feasible solutions are considered. For a solution set without feasible solutions, its HV and IGD+ values are set to 0 and Infinity, respectively. A brief introduction to both performance metrics can be found in Section S-II-B in the supplement.

4) **Parameter settings:** Due to space limit, parameter settings used in our study are summarized in Section S-II-C in the online supplement. These settings include the population size, termination conditions, reproduction operators along with corresponding parameter settings, and control parameters in peer algorithms.

**B. Comparisons within the MOEA/D framework**

Within the MOEA/D framework, experimental results obtained by the five CHTs are tabulated in Tables S-5–S-8 in the online supplement. To make analysis easy, Table I summarizes percentages of the best and second-best HV\(^5\) results obtained by all the CHTs for each test suite and each type of problems. For example, as the table shows, ToM performs either best or second best on 85.7% of the Type-I MW test problems. Regarding the percentage of the best/second-best HV results, it is clear from Table I that ToM is the best CHT in all the cases, except for the Type-III DAS-CMOP. In this case, the best CHT is SR, followed by CDP, and then by ToM. The

\(^4\)The method suggested in [7] is used to control the \( \varepsilon \)-level.

\(^5\)The performance regrading HV is in general consistent with that regarding IGD+. Therefore, we only present comparisons in terms of HV.
above results suggest that ToM is capable of well handling different types of problems within MOEA/D.

<table>
<thead>
<tr>
<th>Type</th>
<th>ToM</th>
<th>CDP</th>
<th>SP</th>
<th>SR</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
<td>I</td>
<td>85.7%</td>
<td>28.6%</td>
<td>0.0%</td>
<td>42.9%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>60.0%</td>
<td>30.0%</td>
<td>40.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>100.0%</td>
<td>66.7%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>DC-DTLZ</td>
<td>I</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>100.0%</td>
<td>37.5%</td>
<td>25.0%</td>
<td>25.0%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>100.0%</td>
<td>25.0%</td>
<td>37.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>DAS-CMOP</td>
<td>I</td>
<td>88.9%</td>
<td>55.6%</td>
<td>44.4%</td>
<td>11.1%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>88.9%</td>
<td>44.4%</td>
<td>11.1%</td>
<td>44.4%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>44.4%</td>
<td>55.6%</td>
<td>33.3%</td>
<td>66.7%</td>
</tr>
<tr>
<td>DAS-CMaOP</td>
<td>I</td>
<td>55.6%</td>
<td>44.4%</td>
<td>44.4%</td>
<td>22.2%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>83.3%</td>
<td>55.6%</td>
<td>11.1%</td>
<td>11.1%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>88.9%</td>
<td>88.9%</td>
<td>0.0%</td>
<td>11.1%</td>
</tr>
</tbody>
</table>

Going one step further, the Wilcoxon’s rank sum test [51], at a 0.05 significance level, is performed to each pairwise comparison between ToM and each of the CHTs in each test instance. The statistical results are represented by three symbols: ●, ○ and †, indicating that ToM performs significantly better than, significantly worse than and equivalently to the compared CHTs, respectively. Table II summarizes these statistical test results. For example, ToM performs (significantly) better than, equivalently to, and (significantly) worse than CDP in 55.0%, 30.0% and 25.0% of all the MW test instances, respectively. For all the four test suites, as can be seen from Table II, the percentage of test instances in which ToM wins is always higher than that in which it loses. This clearly indicates that ToM has statistically significant improvements over the other four CHTs.

<table>
<thead>
<tr>
<th>Type</th>
<th>ToM</th>
<th>CDP</th>
<th>SP</th>
<th>SR</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
<td>●</td>
<td>55.0%</td>
<td>60.0%</td>
<td>55.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>30.0%</td>
<td>40.0%</td>
<td>30.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td>25.0%</td>
<td>0.0%</td>
<td>15.0%</td>
<td>30.0%</td>
</tr>
<tr>
<td>DC-DTLZ</td>
<td>●</td>
<td>95.8%</td>
<td>75.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>4.2%</td>
<td>25.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>DAS-CMOP</td>
<td>●</td>
<td>70.4%</td>
<td>74.1%</td>
<td>63.0%</td>
<td>85.2%</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>7.4%</td>
<td>22.2%</td>
<td>25.9%</td>
<td>14.8%</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td>22.2%</td>
<td>3.7%</td>
<td>11.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>DAS-CMaOP</td>
<td>●</td>
<td>50.0%</td>
<td>81.5%</td>
<td>66.7%</td>
<td>68.5%</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>24.1%</td>
<td>11.1%</td>
<td>24.1%</td>
<td>20.4%</td>
</tr>
<tr>
<td></td>
<td>○</td>
<td>25.9%</td>
<td>7.4%</td>
<td>9.3%</td>
<td>11.1%</td>
</tr>
</tbody>
</table>

Finally, the Friedman test\(^6\) [52] is utilized to compute average rankings of the CHTs regarding both HV and IGD+

\(^6\)The Friedman test is implemented by the KEEL Software Tool available at https://sci2s.ugr.es/keel/.

considering all test instances in each suite. This test procedure, which has been widely employed to evaluate evolutionary algorithms [53], first ranks the CHTs in each test instance separately and then computes an average ranking over all test instances. Clearly, the smaller the ranking, the better the CHT. The Friedman test is very useful to compare overall performance of CHTs. For DC-DTLZ and DAS test suites, as shown in Table III, ToM obtains the best ranking regarding both HV and IGD+. For the WM test suite, ToM ranks first concerning IGD+, and second concerning HV. It can be found that the performance of ToM is steady, obtaining either the best or second-best average rankings in different test suites. In contrast, the performance of some of the other CHTs is not steady. For example, ε is effective for MW test suite, but not quite effective for DC-DTLZ and DAS test suites. Similarly, SP is able to well handle DC-DTLZ, but fails on the other two test suites. The stability of ToM implies its great potential in dealing with different kinds of problems.

C. Comparisons within the NSGA-II framework

In a similar way, the CHTs are compared within the NSGA-II framework. Raw experimental results are given in Tables S-9—S-11 in the supplementary materials, and are summarized in Tables IV—VI. In general, conclusions drawn within NSGA-II are consistent with those drawn within MOEA/D.

<table>
<thead>
<tr>
<th>Type</th>
<th>ToM</th>
<th>CDP</th>
<th>SP</th>
<th>SR</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
<td>I</td>
<td>66.7%</td>
<td>33.3%</td>
<td>33.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>100.0%</td>
<td>37.5%</td>
<td>0.0%</td>
<td>12.5%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>100.0%</td>
<td>33.3%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>DC-DTLZ</td>
<td>I</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>100.0%</td>
<td>25.0%</td>
<td>50.0%</td>
<td>25.0%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td>DAS-CMOP</td>
<td>I</td>
<td>66.7%</td>
<td>22.2%</td>
<td>11.1%</td>
<td>77.8%</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>77.8%</td>
<td>33.3%</td>
<td>44.4%</td>
<td>22.2%</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>33.3%</td>
<td>55.6%</td>
<td>44.4%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

In particular, we have the following observations.

- Table IV emphasizes that, within NSGA-II, ToM is also able to well handle different types of problems, regardless of the test suites. The only exception is Type-III DAS-CMOP, for which ToM does not overwhelmingly outper-
form other CHTs concerning the obtained best/second-best HV results.

- Considering pairwise comparisons between ToM and each CHT, Wilcoxon’s rank sum test results summarized in Table V suggest that ToM is significantly better than other CHTs in at least 64.3% of the test instances for MW and DC-DTLZ. For at least 66.7% of the DAS-CMOP test instances, ToM performs equivalently to CDP, SP and ε. Nevertheless, in this test suite, it outperforms SR, with significant improvements detected in 63.0% of the instances.

- The Friedman test results listed in Table VI indicate that ToM consistently performs best in all the three test suites regarding the overall performance with respect to both HV and IGD+. In contrast, the second-best performing CHT varies, being ε, SR and SP for MW, DC-DTLZ and DAS, respectively. The above results emphasize the steady performance of ToM also within the NSGA-II framework.

In Figs. S-1–S-3 in the supplementary materials, we show final solution sets obtained by the five CHTs within either MOEA/D or NSGA-II on some representative MW, DC-DTLZ and DAS test problems. Visualized comparisons suggest that ToM is the only CHT that is capable of achieving good performance on different types of problems. More detailed discussions can be found in Section S-III in the supplement.

### D. Comparisons with state-of-the-art algorithms

In the previous two sections, we have shown that ToM performs in general better than the four popular CHTs within both MOEA/D and NSGA-II. In this section, we explore how the basic MOEA/D and NSGA-II, when equipped with ToM, perform in comparison with several state-of-the-art CMOEAs/CMAOEAs. Table S-12 in the supplement gives HV results obtained by MOEA/D-ToM, NSGA-II-ToM and five state-of-the-art algorithms mentioned at the beginning of Section IV. By applying Friedman test to these results, we can obtain average rankings of the algorithms (shown in Table VII), and p-values for pairwise comparisons between MOEA/D-ToM (as well as NSGA-II-ToM) and each peer algorithm (shown in Table VIII). It can be found from Table VII that MOEA/D-ToM obtains the best ranking, closely followed by C-TAEA. For NSGA-II-ToM, it performs competitively to PPS-MOEA/D, C-NSGA-III and C-MOEA/DD. Moreover, according to p-values presented in Table VIII, MOEA/D-ToM shows a significant improvement over all the algorithms, except for C-TAEA. Though NSGA-II-ToM is significantly inferior to MOEA/D-ToM and C-TAEA, it is competitive to PPS-MOEA/D, C-NSGA-III and C-MOEA/DD, and significantly superior to C-MOEAD. Based on the above discussions, equipped with ToM, MOEA/D and NSGA-II are capable of achieving better or at least comparable performance to the state-of-the-art algorithms. In particular, MOEA/D-ToM is highly promising: it ranks first among all the algorithms under examination.

### Table V

<table>
<thead>
<tr>
<th></th>
<th>ToM vs. CDP</th>
<th>ToM vs. SP</th>
<th>ToM vs. SR</th>
<th>ToM vs. ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>†</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC-DTLZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>†</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAS-CMOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>†</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table VI

<table>
<thead>
<tr>
<th></th>
<th>ToM</th>
<th>CDP</th>
<th>SP</th>
<th>SR</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC-DTLZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In all the above experiments, we investigate the performance of the algorithms on artificial problems whose types are known. In practice, however, the type of a problem is often not known in advance. The constrained OSPS proposed in [30] is such a problem, which aims at selecting optimal software products from a software product line. This selection should be (nearly) optimal with respect to a set of objectives and
a set of constraints. In this section, we test the algorithms\(^7\) on this practical problem (with 36 test instances in total). Notice that SATVaEA\(^8\) [31], a tailored algorithm for the OSPS problem, is also included in performance comparisons. To be fair, all the algorithms use the same single-point crossover, bit-flip mutation and satisfiability solver based repair operator [30] in the decision space. The HV results are presented in Table S-13 in the online supplement. As seen from that table, MOEA/D-ToM performs best in most of the test instances. Specifically, the algorithm obtains the best HV results in 25 out of the 36 test instances (accounting for 69%). Similarly, the Friedman test is applied to obtain average rankings of the algorithms. As shown in Table IX, MOEA/D-ToM is the best-performing algorithm, while NSGA-II-ToM ranks in the fourth place, following closely to C-NSGA-III. Fig. S-4 in the online supplement compares the distribution of solutions obtained in three 3-objective instances constructed based on toybox, uClinux and 2.6.28.6-ice11 [30]. As observed in that figure, solutions of MOEA/D-ToM are distributed more widely than those of the others, including NSGA-II-ToM. In particular, MOEA/D-ToM is more effective in finding better solutions on the boundaries.

### V. FURTHER DISCUSSIONS

This section presents discussions on usefulness of the switch between objective priority and constraint priority, estimation accuracy of problem types and parameter study on \(\alpha\).

#### A. Usefulness of switch between objective priority and constraint priority

To demonstrate the usefulness of the switch between objective priority and constraint priority, we examine the following three strategies within the framework of MOEA/D. In the first strategy, objective priority is switched to constraint priority once the number of the current generations, \(g\), exceeds 0.5\(G\). We name this strategy SimpleSwitch. In the second strategy, called OnlyObjectivePriority, only objective priority is adopted. Similarly, in the third strategy, which is named OnlyConstraintPriority, only constraint priority is allowed. In other words, switch is omitted in the last two strategies. HV results obtained by the above three strategies are provided in Table S-14 in the supplement. Table X summarizes the percentage of each type of test instances in which the best HV results are obtained. Compared with OnlyConstraintPriority, as seen in the table, OnlyObjectivePriority performs better on Type-I, -II, -I’/II’ problems, but worse on Type-III problems. This finding is consistent with our discussions presented in Section I, stating that emphasizing objectives would be beneficial for Types I/I’ and II/II’, and emphasizing constraints can be useful for some Type-III problems. Moreover, it is observed that SimpleSwitch obtains either the best or second-best overall performance on different types of problems. This clearly implies the usefulness of the switch between objective priority and constraint priority in achieving a trade-off among different problem types.

### TABLE IX

**AVERAGE RANKINGS OF THE ALGORITHMS ON THE PRACTICAL OSPS PROBLEM**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOEA/D-ToM</td>
<td>1.7639</td>
</tr>
<tr>
<td>NSGA-II-ToM</td>
<td>4.6111</td>
</tr>
<tr>
<td>PPS-MOEAD</td>
<td>5.4722</td>
</tr>
<tr>
<td>C-NSGA-III</td>
<td>4.4583</td>
</tr>
<tr>
<td>MOEA/D-CFP</td>
<td>5.0278</td>
</tr>
<tr>
<td>NSGA-II-CFP</td>
<td>5.125</td>
</tr>
<tr>
<td>C-MOEA/DD</td>
<td>6.4444</td>
</tr>
<tr>
<td>SATVaEA</td>
<td>3.0972</td>
</tr>
</tbody>
</table>

In summary, equipped with ToM, both MOEA/D and NSGA-II are able to obtain promising performance on this real-world problem. In particular, MOEA/D-ToM is highly effective, being even better than the tailored SATVaEA. In fact, the good performance of ToM-based algorithms is not accidental. Even though the type of an OSPS instance is not exactly known, it must be one of the types according to our taxonomy. Based on our analysis in Section I, ToM is able to make a trade-off among different types of problems. This conclusion has already been verified by experiments performed on artificial test suites. For the real-world problem, the above conclusion holds as well.

### TABLE X

**THE PERCENTAGE OF EACH TYPE OF TEST INSTANCES IN WHICH THE BEST HV RESULTS ARE OBTAINED BY THE THREE STRATEGIES**

<table>
<thead>
<tr>
<th>Type</th>
<th>SimpleSwitch</th>
<th>OnlyObjectivePriority</th>
<th>OnlyConstraintPriority</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>40%</td>
<td>60%</td>
<td>0%</td>
</tr>
<tr>
<td>II</td>
<td>47%</td>
<td>37%</td>
<td>16%</td>
</tr>
<tr>
<td>I’/II’</td>
<td>45%</td>
<td>36%</td>
<td>18%</td>
</tr>
<tr>
<td>III</td>
<td>42%</td>
<td>8%</td>
<td>50%</td>
</tr>
</tbody>
</table>

\(^7\)Since handling the OSPS problem requires a Java-based library, C-TAEA, which was originally implemented by C/C++, is not tested on this problem. Instead, we add NSGA-II-CFP [8] as a new peer algorithm.

\(^8\)Codes of SATVaEA can be found at [http://doi.org/10.5281/zenodo.3661553](http://doi.org/10.5281/zenodo.3661553)
is set to 0.1. Table XI lists all the test instances in which significant differences are observed between the two switch strategies. Compared with SimpleSwitch, as shown in Table XI, SmartSwitch performs significantly better in three problem instances, and worse in four instances for Types I, II, and I'/II'. It seems that, for these types, the two strategies have similar performance, without one overwhelmingly outperforming the other. For Type III, however, the situation is different: SmartSwitch is significantly better than SimpleSwitch in eight test instances, but worse in only one (i.e., DAS-CMOP2). The effectiveness of SmartSwitch on Type III is attributed to the use of the second condition, which, as already discussed in Section III-B, enables handling this type in a “smart” way. To summarize, the above results emphasize the usefulness of adopting two conditions in the switch between objective priority and constraint priority, particularly for Type-III problems.

B. Estimation accuracy of problem types

In Section III-C, we provide a simple approach to estimate the type of a problem. In this section, we evaluate the accuracy of that approach, and show the performance loss when the type is incorrectly estimated. Table S-15 in the supplement summarizes the estimation accuracy (over 100 runs) and performance loss. It can be found from the table that our estimation approach is able to obtain 100% accuracy in a considerable number of test instances. Precisely, this number is 47, accounting for 66% of all the 71 test instances. If we consider those with estimation accuracy beyond 80% “good” estimations, the number of “good” estimations is up to 61, nearly 86% of all the test instances. As discussed in Section III-C, 100% estimation accuracy is not always available due to the absence of exact UPF and CPF. Nevertheless, according to our results, the proposed approach performs quite well in majority of the cases.

We are also aware that the estimation accuracy is low in several test instances, e.g., MW2 and the 2-objective DC2-DTLZ3. Low accuracy is accompanied with a certain performance loss. Notice that, the performance loss in Table S-15 of the supplement is computed by comparing the HV results obtained by MOEA/D-ToM with those obtained by the algorithm in which the problem type is manually set to the right one. As seen, the performance loss is problem-dependent, with the largest one being 12.24%, observed on the 10-objective DC3-DTLZ3. In all the remaining cases, the loss is within 10%.

Experiments performed in this section bring out the following: the proposed type estimation method enables a high accuracy (beyond 80%) in most cases. Even though low accuracy is obtained in some cases, the resulting performance loss is mostly within an acceptable level.

C. Parameter study

In ToM, α is an important parameter, which determines the switch between objective priority and constraint priority. This section investigates how the performance is affected by settings of this parameter on four representative test problems. Specifically, α is increased from 0.05 to 0.5 with a step size 0.05. In addition, α = 0.01 is also considered as it can be seen as a representative of small values. For each setting of α, logarithmic values of IGD+ obtained by MOEA/D-ToM are shown in Fig. 3 in the form of boxplots. As seen, for Type-II' DC3-DTLZ3, the performance improves obviously as α increases from 0.01 to 0.1, and keeps relatively steady when α ≥ 0.1. It seems that values no less than 0.1 are desirable for this problem. For MW12 (Type III), varying settings for α do not lead to obvious differences in the performance. In other words, ToM is insensitive to α on this test problem. For the Type-III DAS-CMOP7 and DAS-CMOP8, similar variation trends are observed—(logarithmic) IGD+ values monotonously increase with respect to α. This implies the preference to small α on the two problems. To achieve a trade-off among different problems, we can set α to values around 0.1. With this setting, ToM is able to yield nearly optimal performance on DC3-DTLZ3, and not too bad performance on DAS-CMOP7 and DAS-CMOP8.

![Fig. 3. Parameter study of α on four representative problems](image)

In the following, we justify the above setting in general cases. In fact, for Type-I' or Type-II' problems, too small α may not be enough for the population to get across (multiple) infeasible barriers. As a result, the population can be easily stuck at a local front, leading to naturally poor performance. In contrast, for Type-III problems, larger α potentially results in more wastes of computational resources (which has already been discussed in Section III-B). Therefore, considering different types of problems, it makes sense to set α to a reasonably large value, like for example 0.1.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we suggest a new CHT (called ToM) that considers the type of a problem when balancing between constraints and objectives. The CHT explicitly divides the evolutionary process into two stages, in which objective priority and constraint priority are implemented, respectively. The switch between the two stages is automatically realized by...
a smart strategy. In addition, a simple method is proposed to estimate the types of the problems, which are used to guide the population reconstruction when objective priority is switched to constraint priority. According to our analysis, ToM is able to achieve a trade-off among different types of problems. To verify this, we integrate it into both decomposition-based and non-decomposition-based frameworks. Experimental results on both artificial and practical problems show that ToM can indeed maintain a good trade-off among different problem types. Moreover, we experimentally demonstrate that the smart switch between objective priority and constraint priority is useful, particularly for Type-III problems. Besides, we show that the proposed type estimation method works well in most cases, enabling a high accuracy (beyond 80%) in 86% of all the test instances. In summary, it is promising to take into account problem types when balancing constraints and objectives.

Currently, the switch between objective priority and constraint priority is executed only once. In the future, it is possible to perform dynamical switches to properly handle CMOPs where the feasible region is totally separated by infeasible regions (e.g., C1-DTLZ3 [54]). Moreover, ToM may be further improved if it works collaboratively with other CHTs. Finally, exploration of ToM-based algorithms in other real-world applications is also an important part of our future studies.

REFERENCES


Yi Xiang received the B.Sc. and M.Sc. degrees in mathematics from Guangzhou University, Guangzhou, China, in 2010 and 2013, respectively, and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, in 2018. He is currently a Post-Doctoral Fellow at the School of Software Engineering, South China University of Technology, Guangzhou. His current research interests include many-objective optimization and search-based software engineering.

Xiaowei Yang received the B.S. degree in theoretical and applied mechanics, the M.Sc. degree in computational mechanics, and the Ph.D. degree in solid mechanics from Jilin University, Changchun, China, in 1991, 1996, and 2000, respectively. He is currently a Full Time Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His current research interests include algorithms for large-scale pattern recognitions, imbalanced learning, semi-supervised learning, support vector machines, tensor learning, and evolutionary computation. He has published over 100 journals and refereed international conference articles, including the areas of structural reanalysis, interval analysis, soft computing, support vector machines, and tensor learning.

Han Huang (M’15, SM’18) received the B.Eng. degree in applied mathematics, and the Ph.D. degree in Computer Science from the South China University of Technology (SCUT), Guangzhou, in 2008. Currently, he is a professor at School of Software Engineering in SCUT. His research interests include theoretical foundation and application of evolutionary computation and swarm intelligence. Dr. Huang is a senior member of CCF.

Jiahai Wang (M07, SM’19) received the B.Eng. degree in computer science from University of Toyama, Toyama, Japan, in 2005. In 2005, he joined Sun Yat-sen University, Guangzhou, China, where he is currently a full professor with the School of Computer Science and Engineering. He is currently leading an Intelligent Optimization and Learning Laboratory, Sun Yat-sen University. He has published a series of papers in leading conferences and top journals including AAAI, ACL and IEEE Transactions. His main research interests include computational intelligence (deep neural networks and metaheuristics) and its applications. Prof. Wang is a distinguished member of CCF.