Enhancing Decomposition-based Algorithms by Estimation of Distribution for Constrained Optimal Software Product Selection

Yi Xiang, Xiaowei Yang, Yuren Zhou and Han Huang, *Member, IEEE*

***Abstract*—This paper integrates an estimation of distribution based update operator into decomposition-based multi-objective evolutionary algorithms for binary optimization. The probabilistic model in the update operator is a probability vector, which is adaptively learnt from historical information of each subproblem. We show that this update operator can significantly enhance decomposition-based algorithms on a number of benchmark problems. Moreover, we apply the enhanced algorithms to the constrained optimal software product selection problem in the field of search-based software engineering. For this real-world problem, we give its formal definition and then develop a new repair operator based on satisfiability solvers. It is demonstrated by the experimental results that the algorithms equipped with the estimation of distribution operator are effective in dealing with this practical problem, particularly for large-scale instances. The interdisciplinary studies in this paper provide a new real-world application scenario for constrained multi-objective binary optimizers, and also offer valuable techniques for software engineers in handling the optimal software product selection problem.**

***Index Terms*—Decomposition-based multi-objective algorithm, estimation of distribution, constrained multi-objective optimization, optimal software product selection, search-based software engineering.**

I. Introduction

Manuscript received November xx, 2018; revised February xx, 2018; accepted July xx, 2019. Date of publication June x, 2019; date of current version March xx, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61773410, Grant 61673403, Grant 61703183 and Grant 61876207, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2014A030306050, in part by the Guangdong High-Level Personnel of Special Support Program under Grant 2014TQ01X664, in part by the International Cooperation Project of Guangzhou under Grant 201807010047, in part by the Science and Technology Program of Guangzhou under Grant 201607010069, Grant 201802010007 and Grant 201804010276, in part by Guangdong Province Key Area R&D Program under Grant 2018B010109003, and in part by the Science Technique Department of Guizhou Province under Grant [2019]1164. (*Corresponding author: XiaoWei Yang*)

Y. Xiang is with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China, and also with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: xiangyi@scut.edu.cn).

X. Yang and H. Huang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: xwyang@scut.edu.cn; hhan@scut.edu.cn).

Y. Zhou is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China, and also with the Collaborative Innovation Center of High Performance Computing, Sun Yat-sen University, Guangzhou 510006, China (e-mail: zhouyuren@mail.sysu.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier xxxxxxxxxxxx

MULTI-objective evolutionary algorithm based on decomposition (MOEA/D), first proposed by Zhang and Li [1] in 2007, offers an algorithm framework where a multi-objective optimization problem (MOP) is explicitly decomposed into several scalar subproblems by using *weight vectors*. Since the first proposition of MOEA/D, decomposition-based multi-objective evolutionary algorithms (MOEAs) have attracted significant attention from researchers. Over the past ten years, there have been a number of studies on the improvements/extentions of decomposition-based algorithms from different aspects. Among them, several works have employed another type of decomposition, where the objective space is divided into a number of subspaces by using *reference vectors* [2]–[5][1]. For a comprehensive survey of the decomposition-based MOEAs, please refer to [6].

In the original MOEA/D [1], genetic operators are incorporated to generate new solutions. Concretely, the one-point crossover and the bit-flip mutation are used for binary optimization problems; the simulated binary crossover and the polynomial mutation are widely used for continuous optimization problems. As discussed in [6], different reproduction operators may be required across different problems. In order to improve the performance of decomposition-based algorithms, several reproduction operators have been investigated within the MOEA/D framework. These reproduction operators are typically implemented by other metaheuristics, including differential evolution (DE) [7], [8], ant colony optimization (ACO) [9], particle swarm optimization (PSO) [10]–[12], simulated annealing (SA) [13] and covariance matrix adaptation evolution strategy (CMA-ES) [14].

Estimation of distribution algorithms (EDAs) [15]–[18] are stochastic optimization algorithms that guide the search towards the optimum by building and sampling explicit *probabilistic models* of promising candidate solutions. The optimization process is viewed as a number of incremental updates of a probabilistic model, encoding the uniform distribution initially and generating only the global optima eventually. Different from most conventional evolutionary algorithms (EAs) which generate new solutions using one or more variation operators, EDAs generate solutions by constructing and sampling a probabilistic model that could capture the distribution of promising solutions. Using explicit probabilistic models in optimization has allowed EDAs to solve many large and complex problems

[1]In the literature, different terminologies have been used for the second type of decomposition, such as *direction vectors* [2], *reference lines* [3], and *reference vectors* [5]. In this work, we unify them to *reference vectors*.

[17].

There have been several studies aiming at incorporating EDAs into the decomposition-based framework. For continuous optimization problems, typical works can be found in [19]–[22], etc. Since the scope of this paper is about combinatorial optimization, we focus on the review of related works that combines EDAs with MOEA/D for combinatorial optimization problems. Shim et al. [23] integrated EDA into MOEA/D for solving the multi-objective multiple traveling salesman problem. In this algorithm, the used EDA is based on the univariate modeling [16], which is simple, efficient and requires no internal parameter tuning. Zhou et al. [24] proposed an approach, named multi-objective estimation of distribution algorithm based on decomposition, for solving multi-objective traveling salesman problems (MOTSPs). In the algorithm, an MOTSP is decomposed into a set of scalar subproblems, and a probabilistic model, using both priori and learned information, is constructed to guide the search for each subproblem. For each subproblem i, the probabilistic model is represented by a matrix P^i, which is defined based on a pseudo distance matrix D^i and a learned information matrix Q^i. Zangari et al. [25] introduced a general multi-objective decomposition-based EDA using Kernels of Mallows models for solving multi-objective permutation-based optimization problems. In the framework, a Mallows model is learnt using the neighboring solutions of a subproblem. Results on the multi-objective permutation flowshop scheduling problem demonstrated the validity of the proposed algorithm.

Li et al. [26] incorporated an EDA-based reproduction operator into the MOEA/D framework to deal with multi-objective knapsack problems (MOKPs). In the algorithm, a probability vector is maintained for each subproblem, and is built from all the neighboring solutions for a subproblem. Since the neighborhood size of each subproblem is relatively small, however, the learned probability vector maybe easily suffer from premature convergence [27]. To overcome the above drawback, Wang et al. [27] suggested to add a mall value to the probability vector. In this way, however, an extra parameter s is involved. Souza et al. [28] investigated the effect of probabilistic graphical models (PGMs) within MOEA/D, and proposed a framework named MOEA/D-GM for combinatorial optimization problems. Different EDAs can be instantiated in the MOEA/D-GM framework, including the univariate marginal distribution algorithm (UMDA) [16], the population-based incremental learning (PBIL) [15] and the Tree-EDA [29]. All the probabilistic models are learnt using the solutions from the neighborhood of each subproblem. Note that the Tree-EDA is capable of capturing pairwise interactions between decision variables, and the MOEA/D-Tree, a variant of MOEA/D-GM using Tree-EDA, has been shown to be effective in dealing with the bi-objective trap problems.

In this paper, we enhance the performance of decomposition-based algorithms for multi-objective binary optimization by incorporating an estimation of distribution (EoD) update operator. Within the decomposition-based framework, new solutions generated by any reproduction operator can be updated by EoD. For each subproblem or each subspace[2], this operator maintains a probability vector as the probabilistic model. Different from most previous works where the probabilistic model is learnt from neighboring solutions [25]–[28], the probability vector in EoD is constructed based on historical information of each subproblem. The motivations of using historical information are twofold. First, the neighborhood size of a subproblem is often relatively small. In this case, using neighboring information may be insufficient to learn a probabilistic model that could capture well the real distribution of promising solutions [27]. Second, as shown in [20], the previously visited solutions can provide useful information for improving the performance (especially the convergence) of algorithms. Even though the above conclusion is drawn in the context of continuous optimization, it would be of reference value for discrete optimization as well. To use historical information, a key issue is how to record and capture this information. Compared with non-decomposition-based algorithms, those with decomposition in fact provide a more natural way to record historical information. Since the subproblems (or subspaces) are explicit in decomposition-based algorithms, we can easily capture the solutions visited in each subproblem, and then record useful information that will be used to update the probability vector, such as the total number of solutions visited till now (more details will be available in Section III-A).

In this work, we show that the EoD can be easily incorporated into decomposition-based algorithms, and then apply the resulting algorithms to the constrained optimal software product selection (OSPS) problem, a real-world problem derived from the search-based software engineering (SBSE). In the OSPS problem, the goal is to search for optimal software products from a software product line (SPL) by simultaneously optimizing multiple and usually conflicting objectives. The search usually works on binary variables in a large-scale (with up to 20,000+ decision variables [30]) and highly constrained decision space. Therefore, the OSPS problem is a large-scale multi-objective binary optimization problem [30].

Recently, Sayyad et al. [31] have demonstrated that MOEAs are a promising tool for tracking the OSPS problem. They evaluated seven MOEAs, including NSGA-II [32], SPEA2 [33], IBEA [34] in two OSPS instances with five optimization objectives, finding that IBEA outperformed the others. Subsequently, Sayyad et al. [35] improved the performance of both IBEA [34] and NSGA-II [32] in large-scale OSPS instances (with 6,888 decision variables) by adding a rich "seed" in the midst of a randomly-generated initial population. Alternatively, Henard et al. [36] enhanced MOEAs by using satisfiability (SAT) solvers to implement new mutation and replacement operators. The "SIP" method, introduced by Hierons et al. [37], integrates a novel encoding (that shrinks the representation) and an "$1+n$" approach (that prioritises the number of constraints that fail) into MOEAs to improve their

[2]Recall that in the first type of decomposition (using weight vectors), an MOP is usually converted into *subproblems*. In the second type of decomposition (using reference vectors), the objective space is divided into *subspaces*. Hereinafter, we may use only "subproblems" for the sake of brevity.

performance. The MOEAs evaluated in [37] include NSGA-II [32], IBEA [34], MOEA/D [1] and SPEA2+SDE [38]. To the best of our knowledge, the work in [37] first used decomposition-based algorithms to solve the OSPS problem. In our previous work [30], a new algorithm named SATVaEA was proposed by combining two types of SAT solvers with the VaEA framework [39]. In the new algorithm, SAT solvers are used to implement an effective and efficient repair operator. Experimental results in a number of large-scale real-world OSPS instances demonstrated that SATVaEA was capable of returning a set of more diversified software products in comparison with some prior algorithms [36], [37].

In all the previous works [30], [31], [35]–[37], the constraints considered are those constraints whose violations will result in invalid software products. Since they are related to the validity or invalidity of software products, we call them "hard constraints". In practice, from the perspective of an end user, she/he may have limited budget for the software to be selected. In this case, the budget can be viewed as a constraint. Similarly, the constraints on the defects in a software product can be also applied. In contrast to hard constraints, these constraints are typically related to the demands of users, e.g., limitations on budget and defects. Therefore, we can call them "soft constraints". In this paper, we give the formal definition of the OSPS problem with both hard and soft constraints (more details can be found in Section II-B). To our best knowledge, this has not been done previously.

In this paper, we enhance the performance of decomposition-based algorithms by using tbe EoD update operator, and then apply the new algorithms to the constrained OSPS problem. Main contributions of this work are summarized as follows.

- An effective EoD update operator that can be used to improve decomposition-based algorithms for binary optimization. In the proposed EoD, the probabilistic model is a probability vector, which is learnt based on historical information of a subproblem. This learning rule is different from what was often used in the previous works [25]–[28]. Moreover, this learning rule is made self-adaptive, introducing no parameters to be tuned by users.
- The first formal proposition of the constrained OSPS problem in the SBSE field. Previously, the OSPS problem with only hard constraints has been extensively investigated [30], [31], [35]–[37]. This paper attempts to handle this problem considering both hard and soft constraints. Moreover, we develop a new SAT solvers-based repair operator for the problem.
- The interdisciplinary studies in this work can be of significance in both the evolutionary multi-objective optimization (EMO) and SBSE communities. For the EMO community, the constrained OSPS problem provides a new application scenario for binary multi-objective optimization algorithms. Meanwhile, for the SBSE community, a new repair operator and some new algorithms are developed for the OSPS problem.

The remainder of the paper is organized as follows. Section II introduces the constrained OSPS problem. Section III describes the proposed methods in detail. Next, experimental studies and further discussions are presented in Sections IV and V, respectively. Finally, we conclude the paper and outline possible research lines for future studies in Section VI.

II. Problem Statements

In SBSE, one of the most studied problems is the optimal product selection from an SPL. To describe this problem clearly, we need to introduce some important concepts.

A. SPLs and feature models

In an SPL, a set of reusable modular software components are used to systematically configure a family of software products, which share some common functionalities, but differ in some specific aspects of the system functionality [40]. In general, an SPL is represented by a feature model (FM) [41], which is a tree-like structure where each node represents a feature—an abstraction of a functionality or a product characteristic [42], [43]. For example, Fig. 1 shows the feature model (containing 10 features) for a mobile phone SPL.

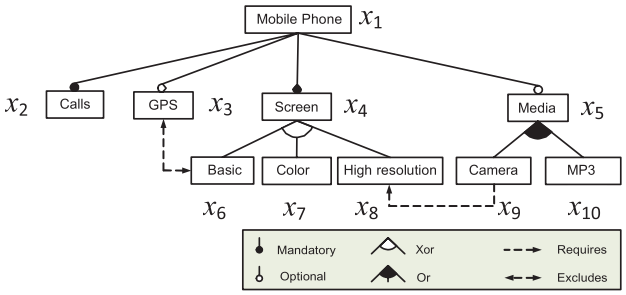


Fig. 1. The feature model for a simplified mobile phone software product line [30].

In an FM, each feature (except the root) has only one parent feature and can have a set of child features. The feature model specifies clearly both the *parent-child relations* (PCRs) and *cross-tree constraints* (CTCs) among features [44]. Given a parent feature X, and its child features $\{x_1, x_2, \ldots, x_n\}$, there are the following four types of PCRs, each of which can be expressed by a propositional formula.

- x_i is a *mandatory* child feature: $x_i \leftrightarrow X$.
- x_i is an *optional* child feature: $x_i \rightarrow X$.
- $\{x_1, \ldots, x_n\}$ is a group of *or* child features: $X \leftrightarrow x_1 \vee \ldots \vee x_n$.
- $\{x_1, \ldots, x_n\}$ is a group of *xor* child features: $(X \leftrightarrow x_1 \vee \ldots \vee x_n) \wedge \{\bigwedge_{1 \le i < j \le n} (\neg(x_i \wedge x_j))\}$.

Given two features x_1 and x_2, there exist the following two CTCs.

- x_1 *requires* x_2: $x_1 \rightarrow x_2$.
- x_1 *excludes* x_2: $\neg(x_1 \wedge x_2)$.

A valid product must simultaneously satisfy all the constraints derived from both PCRs and CTCs. Since any violated constraint will result in software products that are not configurable, we call them "*hard constraints*" (HCs). For example,

HCs for the FM in Fig. 1 can be expressed in the following propositional formula.

$$\begin{aligned}\text{HCs} = {} & x_1 \wedge \{x_1 \leftrightarrow x_2\} \wedge \{x_1 \leftrightarrow x_4\} \wedge \{x_3 \rightarrow x_1\} \\ & \wedge \{x_5 \rightarrow x_1\} \wedge \{x_4 \leftrightarrow \mathit{xor}\{x_6, x_7, x_8\}\} \\ & \wedge \{x_5 \leftrightarrow x_9 \vee x_{10}\} \wedge \{x_9 \rightarrow x_8\} \wedge \{\neg\{x_3 \wedge x_6\}\}\end{aligned} \tag{1}$$

The propositional formula in (1) can be then transformed into the following equivalent formula in conjunctive normal form (CNF).

$$\begin{aligned}\text{HCs} = {} & x_1 \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_4) \\ & \wedge (x_1 \vee \neg x_4) \wedge (x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_5) \wedge (x_4 \vee \neg x_6) \\ & \wedge (x_4 \vee \neg x_7) \wedge (x_4 \vee \neg x_8) \wedge (\neg x_4 \vee x_6 \vee x_7 \vee x_8) \\ & \wedge (\neg x_6 \vee \neg x_7) \wedge (\neg x_6 \vee \neg x_8) \wedge (\neg x_7 \vee \neg x_8) \\ & \wedge (x_5 \vee \neg x_9) \wedge (x_5 \vee \neg x_{10}) \wedge (\neg x_5 \vee x_9 \vee x_{10}) \\ & \wedge (x_8 \vee \neg x_9) \wedge (\neg x_3 \vee \neg x_6)\end{aligned} \tag{2}$$

In (2), there are 19 HCs in total. Notice that each disjunctive formula in (2) is also called a clause.

B. The OSPS problems with soft constraints

For the OSPS problem widely investigated in the literature [30], [31], [36], [37], we need to search for solutions in the decision space $\{0,1\}^n$ by selecting or deselecting each feature x_i $(i = 1, \ldots, n)$. These solutions should 1) satisfy all the HCs specified by the FM, e.g., those in (2), and 2) keep a trade-off among multiple (e.g., four or more) optimization objectives, such as the total cost, the number of deselected features and the known defects [30], [31], [36], [37].

In the previous studies, however, only HCs were considered when selecting software products from an SPL. In practice, as discussed in Section I, we may also need to take into account "*soft constraints*" (SCs), which are related to the demands of users. To construct multi-objective OSPS problems with SCs, we add three attributes to the i-th feature, i.e., $cost_i$, $used\_before_i$ and $defects_i$, following the same suggestions of Sayyad et al. [31], [35], Henard et al. [36] and Hierons et al. [37]. The values of the attributes are generated randomly according to uniform distributions. More specifically, the values of $cost_i$ are distributed uniformly between 5.0 and 15.0, while those of $defects_i$ are random integers between 0 and 10. The $used\_before_i$ takes random boolean values (represented by 0 and 1). In addition, there is a dependency between $used\_before_i$ and $defects_i$: if (not $used\_before_i$) then $defects_i = 0$. Note that the ranges of the attributes are selected following the practice in the prior works [31], [35]–[37].

Based on the above attributes, we can construct the following 2-objective OSPS problem.

$$\text{Minimize} \quad \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \tag{3}$$

$$\text{subject to} \quad \text{HCs in CNF (see Section II-A)} \tag{4}$$

$$\sum_{j=1}^{n} cost_j \cdot x_j \leq \sigma \sum_{j=1}^{n} cost_j \tag{5}$$

$$\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \{0,1\}^n \tag{6}$$

where

$$f_1(\boldsymbol{x}) = n - \sum_{j=1}^{n} x_j \tag{7}$$

$$f_2(\boldsymbol{x}) = \sum_{j=1}^{n} x_j \cdot (1 - used\_before_j) \tag{8}$$

The constraint (4) specifies all the HCs in CNF like those in (2). The constraint condition in (5) means that the cost of the current software product should not be larger than $\sigma \sum_{j=1}^{n} cost_j$, indicating that software engineers or end users may impose restrictions on the budget. The parameter σ in (5) is set to 0.2 in this paper[3]. According to constraint (6), $\boldsymbol{x}$ is an n-dimensional binary vector.

The first optimization objective $f_1(\boldsymbol{x})$, as shown in (7), denotes the number of unselected features. In practice, the products to be configured are expected to provide as many functionalities as possible. Therefore, we seek to maximize the number of selected features, i.e., $\sum_{j=1}^{n} x_j$. Inversely, the number of unselected features should be minimized.

The second optimization objective $f_2(\boldsymbol{x})$, as given in (8), represents the number of features that were not used before. As discussed in [30], [37], features previously not used are more likely to be faulty. Therefore, the number of features that were not used before should be minimized. According to (8), we only consider features that are currently selected. If a feature is not selected, i.e., $x_j = 0$, then $x_j \cdot (1 - used\_before_j)$ contributes 0 to $f_2(\boldsymbol{x})$. Instead, if x_j is selected and was not used before, i.e., $x_j = 1$ and $used\_before_j = 0$, then $x_j \cdot (1 - used\_before_j)$ contributes 1 to $f_2(\boldsymbol{x})$.

By adding the third optimization objective shown in (9),

$$f_3(\boldsymbol{x}) = \sum_{j=1}^{n} defects_j \cdot x_j, \tag{9}$$

we can obtain the following 3-objective OSPS problem.

$$\begin{aligned}&\text{Minimize} \quad \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x})) \\ &\text{subject to} \quad \text{the same constraints as in (4) – (6).}\end{aligned} \tag{10}$$

As seen, the 3-objective OSPS problem has the same constraints as in the 2-objective problem. The added objective, given by (9), is the number of total defects in all the selected features. Definitely, this number should be minimized.

Finally, the 4-objective OSPS problem is in the following form.

$$\text{Minimize} \quad \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x})) \tag{11}$$

$$\text{subject to} \quad \text{the same constraints as in (4) and (5)}$$

$$\sum_{j=1}^{n} defects_j \cdot x_j \leq \delta \sum_{j=1}^{n} defects_j \tag{12}$$

$$\text{the same constraint as in (6)}$$

This problem is constructed by adding the fourth optimization

[3]According to our empirical experiments, there are both feasible and infeasible solutions by setting σ to 0.2. Therefore, the constraint (5) can be used to test an algorithm's ability of distinguishing between feasible and infeasible solutions.

objective as in (13), and a new constraint as in (12). The fourth optimization objective denotes the total cost, which is to be minimized. The added constraint (12) imposes restrictions on the defects allowed. In (12), δ is set to 0.1 in this work[4].

$$f_4(\boldsymbol{x}) = \sum_{j=1}^{n} cost_j \cdot x_j. \tag{13}$$

III. The Proposed Methods

In this section, we start by giving details of the proposed EoD update operator. Then, we integrate this operator into two popular decomposition-based MOEAs, i.e., MOEA/D [1] and NSGA-III [3]. Next, we depict a new repair operator for the OSPS problem. Finally, we show how the soft constraints are handled in both algorithms.

A. The EoD update operator

In most decomposition-based MOEAs, an MOP is decomposed into a number of scalar subproblems by employing N weight vectors. For each subproblem, we can build a probabilistic model using historical information, and then apply this model to update solutions. More specifically, for the i-th ($i = 1, 2, \ldots, N$) subproblem, we maintain a probability vector, denoted by $\boldsymbol{p}_i = (p_{i_1}, p_{i_2}, \ldots, p_{i_n})$, in which the k-th component represents the probability to be "1" (or *true*) in the k-th position. That is to say, $p_{i_k} = P(x_{i_k} = 1)$, where $P(\cdot)$ is the probability of an event. Since the exact p_{i_k} is unknown, we use the following formula to estimate it.

$$p_{i_k} = \alpha \cdot 0.5 + (1 - \alpha) \cdot \frac{T_{i_k}}{S_i}, \tag{14}$$

where T_{i_k} is the accumulated number of 1's in the k-th gene for the i-th subproblem, and S_i denotes the total number of solutions visited till now in the i-th subproblem. Clearly, $\frac{T_{i_k}}{S_i} \leq 1$ as $T_{i_k} \leq S_i$. If we use only the ratio $\frac{T_{i_k}}{S_i}$ to estimate p_{i_k}, this would be problematic if S_i is not large enough, particularly in the early phase of the evolutionary process where a limited number of solutions have been visited.

To handle the above issue, we introduce a learning factor $\alpha \in [0, 1]$, which adds weights to two items, i.e., 0.5 and $\frac{T_{i_k}}{S_i}$. In the early phase, the decision variables take either 0 or 1 with a probability close to 0.5. As the evolution proceeds, the term $\frac{T_{i_k}}{S_i}$ may become more and more accurate to approximate p_{i_k}. Intuitively, α should be large early and small later. Therefore, we can set α according to the following equation.

$$\alpha = 1 - \frac{FEs}{max\_FEs}, \tag{15}$$

where FEs is the number of the current function evaluations, and max_FEs denotes the maximal FEs allowed. Similarly, if the maximal runtime (i.e., max_RT) is used as the termination condition, then α is updated by

$$\alpha = 1 - \frac{RT}{max\_RT}, \tag{16}$$

where RT is the current runtime.

[4]Similar to the parameter σ in (5), δ is also set based on our empirical experiments.

According to (15) and (16), α is linearly decreased from 1.0 to 0.0 during the whole evolutionary process. In the early phase, the second term "$\frac{T_{i_k}}{S_i}$" in (14) is just a coarse estimation to the distribution. Therefore, it is reasonable to give a smaller weight to it, but a larger one to the first term "0.5". That is to say, the p_{i_k} is close to 0.5 in the early phase, doing a random assignment between 1 and 0. In the later phase, however, the p_{i_k} is primarily determined by the second term because the weight $1 - \alpha$ grows as α decreases.

Algorithm 1 *EoD_update_operator* $(\boldsymbol{x}, i, u_p)$

Input: $\boldsymbol{x}$ (the solution to be updated), i (the i-th subproblem) and u_p (the update probability)
Output: $\boldsymbol{x}$ (the updated solution)

```
1:  Construct p_i according to (14)
2:  for k ← 1 to n do
3:      r_1 ← rand(0, 1)
4:      if r_1 < u_p then
5:          r_2 ← rand(0, 1)
6:          if r_2 < p_{i_k} then
7:              x_k ← 1
8:          else
9:              x_k ← 0
10:         end if
11:     end if
12: end for
13: return x
```

The EoD update operator is given in Algorithm 1. As shown in line 1, the probability vector for the i-th subproblem is constructed by using (14) to calculate each component of $\boldsymbol{p}_i$. Similar to mutation operators, the EoD update operator introduces an update probability u_p, which determines the ratio of decision variables to be updated by EoD. For each dimension k, as shown in line 4 of Algorithm 1, the EoD will be applied if a random number r_1 is smaller than u_p. To update x_k by EoD, the procedure needs to check if $r_2 < p_{i_k}$ is true. If so, x_k is set to 1. Otherwise, it is set to 0 (see lines 6-10 in Algorithm 1).

Finally, it is worth mentioning that the EoD update operator can be implemented easily. From the perspective of programming, we just need to maintain a variable for S_i and an array for $\boldsymbol{T}_i = (T_{i_1}, T_{i_2}, \ldots, T_{i_n})$. Moreover, the variable and the array can be automatically updated once a new solution appears in the i-th subproblem. In the following section, we will show that this update operator can be integrated into decomposition-based MOEAs without much effort.

B. Integrating EoD into decomposition-based MOEAs

In this section, we integrate EoD into two popular decomposition-based MOEAs (i.e., MOEA/D and NSGA-III), leading to two new algorithms, MOEA/D-EoD and NSGA-III-EoD.

1) MOEA/D-EoD: The framework of the proposed MOEA/D-EoD is given in Algorithm 2, where the codes related to EoD are underlined. In MOEA/D-EoD, there are two common parameters N and T, which denote the population

size and the neighborhood size, respectively. As in MOEA/D, a set of weight vectors $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ is needed in the new algorithm. Before the algorithm begins, the weight vectors are generated by using systematic approaches [3], [4], [45]. Then, the neighborhoods can be identified by working out T closest weight vectors to each weight vector (see lines 1-2 in Algorithm 2). Next, the population P is randomly initialized. If infeasible solutions appear, then they are repaired according to problem-specific methods. For the OSPS problem, the repair operator will be given in Section III-C. The following are some important steps involved in MOEA/D-EoD.

Algorithm 2 MOEA/D-EoD for binary optimization

Input: N (population size), T (neighborhood size)
Output: The final population P.

1: Initialize N weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$.
2: For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\boldsymbol{w}_{i_1}, \ldots, \boldsymbol{w}_{i_T}$ are T closest (regarding the Euclidean distance) weight vectors to $\boldsymbol{w}_i$.
3: Generate an initial population $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, and repair infeasible solutions according to problem-specific methods.
4: Work out $\boldsymbol{f}(\boldsymbol{x}_1), \ldots, \boldsymbol{f}(\boldsymbol{x}_N)$.
5: **while** *the termination criterion is not fulfilled* **do**
6: Determine $\boldsymbol{z}^{min}$ and $\boldsymbol{z}^{max}$. For each $\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in P$, normalize it to $\widetilde{\boldsymbol{f}}(\boldsymbol{x})$ by (17).
7: Initialize the reference point $\boldsymbol{z}^*$ used in scalarizing functions
8: **for** $i = 1, \ldots, N$ **do**
9: Randomly select two indexes k and l from $B(i)$, and apply crossover operator to $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$ to generate two new solutions $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$.
10: Set $\boldsymbol{y}$ to $\boldsymbol{y}_1$ or $\boldsymbol{y}_2$, both with a probability 0.5. Similarly, set h to either k or l.
11: $EoD\_update\_operator(\boldsymbol{y}, h, u_p)$ // Algorithm 1
12: Repair $\boldsymbol{y}$ using problem-specific methods.
13: Work out $\boldsymbol{f}(\boldsymbol{y})$ and normalize it to $\widetilde{\boldsymbol{f}}(\boldsymbol{y})$ using the same $\boldsymbol{z}^{min}$ and $\boldsymbol{z}^{max}$ as in line 6.
14: Update $\boldsymbol{z}^*$ based on $\widetilde{\boldsymbol{f}}(\boldsymbol{y})$.
15: Update subproblems: For each $j \in B(i)$, if $g^*(\boldsymbol{y}|\boldsymbol{w}_j, \boldsymbol{z}^*) < g^*(\boldsymbol{x}_j|\boldsymbol{w}_j, \boldsymbol{z}^*)$, set $\boldsymbol{x}_j = \boldsymbol{y}$, and update S_j and $\boldsymbol{T}_j$ in EoD. // The g^* is a scalarizing function
16: **end for**
17: **end while**
18: **return** P

*a) **Normalization and estimation of ideal/nadir points**:* The objective vectors are suggested to be normalized (line 6). For each $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T, \boldsymbol{x} \in P$, we normalize it to $\widetilde{\boldsymbol{f}}(\boldsymbol{x}) = (\widetilde{f}_1(\boldsymbol{x}), \ldots, \widetilde{f}_m(\boldsymbol{x}))^T$ by using $\boldsymbol{z}^{min} = (z_1^{min}, \ldots, z_m^{min})^T$ and $\boldsymbol{z}^{max} = (z_1^{max}, \ldots, z_m^{max})^T$ according to the following formula

$$\widetilde{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - z_i^{min}}{z_i^{max} - z_i^{min}}, \tag{17}$$

where z_i^{min} is the minimal value for the i-th objective found so far, and z_i^{max} is the maximal value for the i-th objective in the current population. According to (17), $\widetilde{f}_i(\boldsymbol{x}) \in [0, 1]$.

The components of the ideal point $\boldsymbol{z}^{ideal}$ and the nadir point $\boldsymbol{z}^{nadir}$ define lower and upper bounds for the objective functions, and represent the best and the worst values that each objective function can reach in the PF [46]. Since the true ideal point and nadir point are difficult to obtain, they are usually estimated during the optimization process. After normalization, the ideal point can be estimated by $\boldsymbol{z}^{ideal} = (-0.1, \ldots, -0.1)^T$, which is in fact the same suggestions as in [1] and [47]. According to [46], the nadir point can be estimated to be $\boldsymbol{z}^{max}$ or any objective vector dominated by it. To be consistent with the scale factor (i.e., 0.1) in the estimation of $\boldsymbol{z}^{ideal}$, the nadir point in this work is estimated to be $\boldsymbol{z}^{nadir} = (1.1, \ldots, 1.1)^T$.

In line 7 of Algorithm 2, the reference point $\boldsymbol{z}^*$ should be initialized according to the demands of users. If the ideal point is specified, then $\boldsymbol{z}^*$ is set to $\boldsymbol{z}^{ideal}$. Otherwise, $\boldsymbol{z}^*$ is set to $\boldsymbol{z}^{nadir}$.

*b) **Reproduction operators**:* To generate new solutions, we randomly select two indexes k and l from $B(i)$, and apply a crossover operator to the parents $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$ (line 10). For binary optimization, a number of crossover operators can be applied. Among them, the single-point crossover [48], [49] may be one of the simplest and most used operators. The single-point crossover exchanges the bits of the first parent, from the beginning to the crossover point, with those of the second one. Each time, this will generate two child solutions $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. According to line 10 in Algorithm 2, $\boldsymbol{y}$ is set to either $\boldsymbol{y}_1$ or $\boldsymbol{y}_2$, both with a probability 0.5. That is to say, we randomly select a child solution between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, and assign it to $\boldsymbol{y}$ for further improvement through EoD. Since the genes of $\boldsymbol{y}$ come from either $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$, we use the probability vector of the k-th or the l-th subproblem to update $\boldsymbol{y}$. For this end, h is set to either k or l, and is input as the second parameter in the $EoD\_update\_operator$ (line 11).

*c) **Update subproblems**:* Before subproblems are updated, we need update $\boldsymbol{z}^*$ according to the type of reference point used (line 14 in Algorithm 2). If the ideal point is used, then $\boldsymbol{z}^*$ is updated as follows: For each $j = 1, \ldots, m$, if $\widetilde{f}_j(\boldsymbol{y}) < z_j^*$, then set $z_j^* = \widetilde{f}_j(\boldsymbol{y})$. In the case in which the nadir point is used, the update goes as follows: For each $j = 1, \ldots, m$, if $\widetilde{f}_j(\boldsymbol{y}) > z_j^*$, then set $z_j^* = \widetilde{f}_j(\boldsymbol{y})$.

For each index $j \in B(i)$, both $\boldsymbol{y}$ and $\boldsymbol{x}_j$ are evaluated by a scalarizing function (i.e., g^*) with respect to $\boldsymbol{w}_j$ and the latest $\boldsymbol{z}^*$. As shown in line 15 of Algorithm 2, if $g^*(\boldsymbol{y}|\boldsymbol{w}_j, \boldsymbol{z}^*) < g^*(\boldsymbol{x}_j|\boldsymbol{w}_j, \boldsymbol{z}^*)$, then set $\boldsymbol{x}_j$ to $\boldsymbol{y}$. Since the solution of the j-th subproblem is replaced by $\boldsymbol{y}$, we should update accordingly S_j and $\boldsymbol{T}_j$ in the EoD operator. The S_j is increased by one, and T_{j_k} is also increased by one if $y_k = 1$.

In MOEA/D-EoD, the scalarizing function g^* can be the weighted sum function (g^{WS}) [1], the weighted Tchebycheff function (g^{TCHE1} [1] and g^{TCHE2} [50], [51]), and the penalty-based boundary intersection function (g^{PBI}) [1], [47]. Details on these functions can be found in Section I of the supplementary materials.

Finally, we make the following remarks on the proposed

MOEA/D-EoD.

- A normalization procedure is integrated into the framework. Therefore, the algorithm is capable of handling problems with differently scaled objective functions.
- The scalarizing functions g^{TCHE1}, g^{TCHE2} and g^{PBI} are slightly modified such that the reference point can be either the ideal point $\boldsymbol{z}^{ideal}$ or the nadir point $\boldsymbol{z}^{nadir}$. For more discussions on this, please refer to Section I of the supplementary materials.
- The integration of EoD into MOEA/D is easy. As seen in Algorithm 2, the framework of MOEA/D-EoD differs from the original MOEA/D in only two places (lines 11 and 15).

2) NSGA-III-EoD: The framework of NSGA-III-EoD is presented in Algorithm 3. As seen, the new solutions generated by crossover operators are further improved by EoD (lines 8 and 9). According to line 14, the mixed population $S = P \cup Q$ undergoes the same operations as in the original NSGA-III. After this step, N promising solutions are selected from S to form the population for the next generation. For more details on the non-dominated sorting, the normalization, the association and the niche-preservation operations, please refer to the original study [3].

Algorithm 3 NSGA-III-EoD for binary optimization

Input: N (population size)
Output: The final population P.

1: Initialize N reference points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$.
2: Generate an initial population $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, and repair infeasible solutions according to problem-specific methods.
3: **while** *the termination criterion is not fulfilled* **do**
4: $Q = \emptyset$
5: **for** $i = 1, \ldots, N/2$ **do**
6: Select two random indexes $k, l \in \{1, 2, \ldots, N\} \wedge k \neq l$.
7: Apply crossover operator to $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$ to generate two new solutions $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$.
8: $EoD\_update\_operator(\boldsymbol{y}_1, k, u_p)$ // Algorithm 1
9: $EoD\_update\_operator(\boldsymbol{y}_2, l, u_p)$ // Algorithm 1
10: Repair $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ using problem-specific methods.
11: Add $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ into Q.
12: **end for**
13: $S = P \cup Q$
14: Select N promising solutions from S to construct the next population P by using the non-dominated sorting, the normalization, the association and the niche-preservation operations as in the original NSGA-III.
15: For each $j \in \{1, \ldots, N\}$, update S_j and $\boldsymbol{T}_j$ in EoD.
16: **end while**
17: **return** P

What we focus on here is how to update S_j and $\boldsymbol{T}_j$ in the EoD operator (line 15 of Algorithm 3). In NSGA-III-EoD, the objective space is divided into N subspaces by N reference lines (determined by reference points). In the selection process presented in line 14, each of the solutions for the next population has already been associated with a reference line according to the perpendicular distance and the niche count. Let the niche count for the j-th reference line be ρ_j, then S_j is updated as: $S_j \leftarrow S_j + \rho_j$. The solutions associated with the j-th reference line are used to update $\boldsymbol{T}_j$. For each of the associated solutions, the T_{j_k} is increased by one if the k-th gene of this solution is 1.

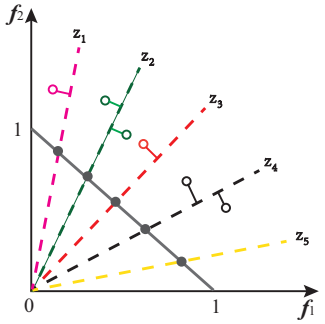


Fig. 2. An illustration of updating S_j and $\boldsymbol{T}_j$ in NSGA-III-EoD.

Fig. 2 shows an example of updating S_j and $\boldsymbol{T}_j$ in NSGA-III-EoD. In this figure, both the first and the third reference lines have only one associated solution, therefore $S_1 = S_1 + 1$ and $S_3 = S_3 + 1$. Similarly, $S_2 = S_2 + 2$ and $S_4 = S_4 + 2$ because two solutions are associated with both the second and the forth reference lines. Since there are no solutions being associated with $\boldsymbol{z}_5$, we do not need to update S_5 and $\boldsymbol{T}_5$.

Finally, it should be mentioned that EoD can be easily integrated into decomposition-based MOEAs, where an MOP is decomposed into a number of subproblems (as in MOEA/D), or the objective space is divided into several subspaces (as in NSGA-III). In both cases, we can easily record the historical information for each subproblem or each subspace, i.e., S_j and $\boldsymbol{T}_j$, $j = 1, \ldots, N$. Given this, the integration of EoD would be easy in decomposition-based algorithms. In this section, we provide two paradigms, i.e., MOEA/D-EoD and NSGA-III-EoD. In practice, one can also integrate EoD into other decomposition-based MOEAs.

C. A new repair operator for the OSPS problem

As shown in Algorithms 2 and 3, infeasible solutions should be repaired in both the initialization and reproduction phases. In general, repair operators are designed according to the properties of the problems at hand. For the OSPS problem, we suggest a new SAT solver-based operator to repair solutions violating HCs.

According to Section II-A, HCs can be expressed in CNF. In fact, the search for solutions satisfying all the HCs is essentially to solve an SAT problem. Therefore, SAT solvers can be naturally applied [30]. In the literature, there are two types of high-performance SAT solvers: conflict-driven clause learning (CDCL) algorithms [53]–[56] and stochastic local search (SLS) algorithms [52], [57]–[59]. Inspired by the work in [30], we propose to use two types of SAT solvers to repair infeasible solutions. The selected CDCL and SLS-type solvers are SAT4J [53] and probSAT [52], respectively.

Algorithm 4 Repair operator for the OSPS problem

Input: $\boldsymbol{y}$ // An infeasible solutions violating HCs
Output: $\boldsymbol{y}$ // The solution after repairing
1: **if** $rand() < \tau$ **then**
2: Repair $\boldsymbol{y}$ using the probSAT solver [52]
3: **else**
4: Repair $\boldsymbol{y}$ using the SAT4J solver [53]
5: **end if**
6: **return** $\boldsymbol{y}$

We choose SAT4J because of its popularity and effectiveness when applied to practical problems in software engineering [30], [36], [60], [61], and choose probSAT as it is one of the simple yet effective SLS algorithms [52]. The SAT4J solver employs a systematic backtracking search procedure to explore the whole decision space, looking for satisfying assignments or proving the unsatisfiability of SAT instances. In contrast, the probSAT solver uses a greedy search procedure to find solutions satisfying as many constraints as possible. More specifically, each time probSAT picks a variable to flip according to its selection probability, which is calculated based on the fitness values of the variables. The fitness value indicates how good a variable is if this variable is flipped. For technical details on SAT4J and probSAT, please refer to [53] and [52], respectively.

The repair operator for the OSPS problem is given in Algorithm 4. For an infeasible solution $\boldsymbol{y}$ (violating HCs), it is repaired by either the probSAT solver [52] with a probability τ, or the SAT4J solver [53] with a probability $1-\tau$. Since the probSAT does not explore the whole decision space, it is computationally cheaper than SAT4J, which has an exponential worst-case time complexity. However, starting from an infeasible solution, this solver has no guarantee to find a feasible solution. Therefore, the SAT4J solver is introduced to remedy the above drawback. The parameter τ controls the computational resources allocated to the two solvers, and its effect will be investigated in Section V-A.

To use SAT4J solver to repair an infeasible solution $\boldsymbol{y}$ (line 4 in Algorithm 4), we find out the variables that are not involved in the violations of constraints. Then, we keep their values and call the solver to find a feasible solution by assigning values to the rest of the variables [36]. For example, consider an FM with 5 features and 3 HCs: $FM = (x_1 \vee x_5) \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5)$. The solution $\boldsymbol{y} = \{0, 0, 1, 1, 0\}$ is infeasible since it violates two constraints $(x_1 \vee x_5)$ and $(x_2 \vee x_5)$, involving three variables x_1, x_2 and x_5. We remove their assignments and make $\boldsymbol{y} = \{\_, \_, 1, 1, \_\}$ partially feasible. Then $\boldsymbol{y}$ is given to the SAT4J solver, which will complete it and return a feasible solution. For example, it may return the following solution: $\boldsymbol{y}\prime = \{1, 1, 1, 1, 0\}$. Therefore, $\boldsymbol{y}$ is repaired to $\boldsymbol{y}\prime$.

D. Handling soft constraints

A solution satisfying all the HCs can violate SCs defined in (5) and (12). To handle this situation, we introduce the constraint violation (CV) value [62]. The CV value of the solution $\boldsymbol{x}$ indicates the degree of the violation of constraints.

Algorithm 5 Update subproblems considering CV values

Input: $\boldsymbol{y}$ and $B(i)$
1: **for** $j \in B(i)$ **do**
2: **if** $CV(\boldsymbol{y}) < CV(\boldsymbol{x}_j)$ **then**
3: Set $\boldsymbol{x}_j = \boldsymbol{y}$, and update S_j and $\boldsymbol{T}_j$ in EoD.
4: **else if** $CV(\boldsymbol{y}) = CV(\boldsymbol{x}_j)$ **then**
5: If $g^*(\boldsymbol{y}|\boldsymbol{w}_j, \boldsymbol{z}^*) < g^*(\boldsymbol{x}_j|\boldsymbol{w}_j, \boldsymbol{z}^*))$, then set $\boldsymbol{x}_j = \boldsymbol{y}$, and update S_j and $\boldsymbol{T}_j$ in EoD.
6: **end if**
7: **end for**

For both the 2- and 3-objective OSPS problems, there is only one SC as given in (5). As suggested in [62], we rewrite (5) in the following form to normalize the constraint.

$$h(\boldsymbol{x}) = \frac{\sum_{j=1}^{n} cost_j \cdot x_j}{\sigma \sum_{j=1}^{n} cost_j} - 1 \leq 0. \tag{18}$$

Then $CV(\boldsymbol{x}) = \max\{0, h(\boldsymbol{x})\}$. A larger CV value means more violations of the constraint. Similarly, we can calculate the CV value of the solutions for the 4-objective OSPS problem. Since there are two SCs, i.e., (5) and (12), we calculate the CV value for each constraint and use the sum of these values as the final CV value. In the presence of SCs, line 15 in Algorithm 2 should be modified to Algorithm 5; in line 14 of Algorithm 3, the Pareto domination in the non-dominated sorting procedure should be replaced by the constrained domination principle [62], which emphasizes feasible and small CV-value solutions.

TABLE I
FEATURE MODELS USED IN THIS STUDY.

FM	Features (#)	HCs (#)
toybox	181	477
axTLS	300	1,657
freebsd-icse11	1,392	54,351
fiasco	631	3,314
uClinux	606	606
busybox-1.18.0	2,845	12,145
2.6.28.6-icse11	6,742	227,009
uClinux-config	5,227	23,951
coreboot	7,566	40,736
buildroot	8,150	37,294
freetz	16,481	85,671
2.6.32-2var	27,077	189,883
2.6.33.3-2var	28,115	195,815

IV. EXPERIMENTAL STUDIES

In this section, we first give the experimental results on two benchmark problems, showing that the EoD can indeed improve the performance of decomposition-based algorithms. Then we conduct a series of experiments on the OSPS problem, showing again the performance gain of EoD, as well as the superiority of the enhanced algorithm over the latest SATVaEA [30] on the OSPS problems considered in this work.

A. Experiments on benchmark problems

Two benchmark problems are chosen in this study. One is the MOKP [47], [63], [64], for which we consider five instances, i.e., 2-500, 4-500, 6-500, 8-500 and 10-500 [47].

TABLE II
AVERAGE HV OBTAINED BY MOEA/D-EOD AND MOEA/D ON THE 2-OBJECTIVE OSPS INSTANCES.

	FM	TCHE1		TCHE2		PBI	
		MOEA/D-EoD	MOEA/D	MOEA/D-EoD	MOEA/D	MOEA/D-EoD	MOEA/D
HV	2.6.28.6-icse11	$7.215E-01$	$7.221E-01$	$7.192E-01$	$7.198E-01$	$7.177E-01$	$7.116E-01$
	freebsd-icse11	$8.291E-01$	$7.278E-01$	$8.047E-01$	$7.009E-01$	$8.102E-01$	$6.780E-01$
	uClinux-config	$7.204E-01$	$7.071E-01$	$7.063E-01$	$6.971E-01$	$7.015E-01$	$6.859E-01$
	buildroot	$7.207E-01$	$7.110E-01$	$7.102E-01$	$7.063E-01$	$6.987E-01$	$6.965E-01$
	freetz	$7.070E-01$	$7.099E-01$	$7.038E-01$	$7.065E-01$	$6.976E-01$	$7.024E-01$
	coreboot	$7.711E-01$	$5.409E-01$	$7.605E-01$	$5.523E-01$	$7.692E-01$	$6.847E-01$
	2.6.33.3-2var	$6.784E-01$	$6.989E-01$	$6.841E-01$	$6.938E-01$	$6.927E-01$	$6.945E-01$
IGD+	2.6.28.6-icse11	$3.120E-02$	$3.157E-02$	$3.296E-02$	$3.243E-02$	$3.265E-02$	$3.710E-02$
	freebsd-icse11	$1.165E-01$	$2.062E-01$	$1.322E-01$	$2.175E-01$	$1.230E-01$	$2.359E-01$
	uClinux-config	$6.689E-02$	$7.946E-02$	$7.559E-02$	$8.604E-02$	$7.854E-02$	$9.368E-02$
	buildroot	$3.733E-02$	$4.257E-02$	$4.340E-02$	$4.529E-02$	$5.049E-02$	$5.114E-02$
	freetz	$2.118E-02$	$1.912E-02$	$2.287E-02$	$2.099E-02$	$2.724E-02$	$2.321E-02$
	coreboot	$5.884E-02$	$1.994E-01$	$6.732E-02$	$1.945E-01$	$5.840E-02$	$1.006E-01$
	2.6.33.3-2var	$2.475E-02$	$1.610E-02$	$2.226E-02$	$1.886E-02$	$1.903E-02$	$1.856E-02$

The number before "-" is the number of objectives, while that after "-" denotes the number of decision variables. The other is the multiobjective unconstrained binary quadratic programming (mUBQP) problem [65], where four parameters are used to define an mUBQP instance, and they are the problem size n, the number of objective functions m, the matrix density d and the objective correlation coefficient ρ. Following the practice in [65], $d = 0.8$, $m \in \{2, 3\}$, $n \in \{1000, 2000, 3000, 4000\}$ and $\rho \in \{-0.5, -0.2, 0.0, 0.2, 0.5\}$. Therefore, we have $2 \times 4 \times 5 = 40$ instances in total. In this paper, the tuple (ρ, n) is used to represent an mUBQP instance for each m. Due to space limitations, mathematical formulations of the above two benchmark problems, the experimental settings, as well as the results and discussions are given in Section II of the supplementary materials. The following are some major observations that we have:

- The EoD significantly improves the performance of MOEA/D on both MOKP and mUBQP test problems, independent of the scalarizing functions and reference points.
- In the EoD operator, the adaptive learning factor α performs better than fixed values concerning both the indicator values and the robustness.
- The update probability u_p in EoD should not be too large. In general, a value around 0.01 could yield satisfactory results.

On the basis of the above observations, we conduct a series of experiments on the OSPS problem in the following section.

B. Experiments on the OSPS problem

We first specify computational settings, and then report experimental results in 2-, 3- and 4-objective OSPS instances.

1) Settings of computational experiments: In our experiments, the settings include:

a) Feature models used: In this paper, we use 13 FMs[5] in Table I to construct instances of the OSPS problem. These models were reverse-engineered from real-world projects, such as the Linux kernel, uClinux and FreeBSD operating systems [66]. According to [30], the FMs can be simplified by removing *mandatory* and *dead* features via boolean constraint propagation (BCP) [67] (for more details please refer to [30]). Table I gives the number of features and that of HCs after applying BCP. We assume that large-scale FMs are those models with more than 3,000 features. As seen, most of these models are large-scale, with even 28,115 features and 227,009 HCs. For each model, according to Section II, we can construct OSPS instances with 2, 3 and 4 objectives.

b) Population size and termination condition: The population size N is set to 100, 105 and 120 for 2-, 3- and 4-objective OSPS problems, respectively. Following the same practice as in [30], we use max_RT as the termination condition, which is set to 6 seconds for toybox and axTLS, 30 seconds for fiasco, uClinux and busybox-1.18.0, and 200 seconds for all the other FMs in Table I. Note that these settings are independent of the number of objectives.

c) Parameter settings in algorithms: In MOEA/D-EoD and MOEA/D, the neighbor size T is set to 10, and the penalty parameter θ is set to 5 if the PBI scalarizing function is used. In both algorithms, we employ the single-point crossover with the crossover probability being 1.0. In MOEA/D, the bit-flip mutation is adopted, and the mutation probability is set to 0.01. According to the results in the previous section, the update probability u_p is set to 0.01 in MOEA/D-EoD. For SATVaEA, the same genetic operators (including parameter settings) as in MOEA/D are used in this study. Moreover, all the three algorithms adopt the same repair operator as described in Section III-C. In this operator, the parameter τ is set to 0.9.

d) Performance metrics: To evaluate algorithms, we choose the hypervolume (HV) [63] and IGD+ [68] [69] as performance metrics, and both of them can simultaneously measure convergence and diversity. The HV metric is Pareto compliant [70], while the IGD+ metric is weakly Pareto compliant [69]. Due to the above good theoretical properties, both metrics have been widely used for performance evaluations in the evolutionary multi-objective optimization. Notice that a larger HV indicates a better approximation front. For IGD+, however, a small value is desired.

2) Results on 2-objective OSPS instances: Table II shows the average HV and IGD+ values obtained by MOEA/D-EoD (with TCHE1) and MOEA/D (with TCHE1) in some large-scale 2-objective OSPS instances. As seen, MOEA/D-EoD obtains better HV values than MOEA/D on all the models except for 2.6.28.6-icse11, freetz and 2.6.33.3-2var regarding

[5]The models are available at the LVAT repository: http://code.google.com/p/linux-variability-analysis-tools

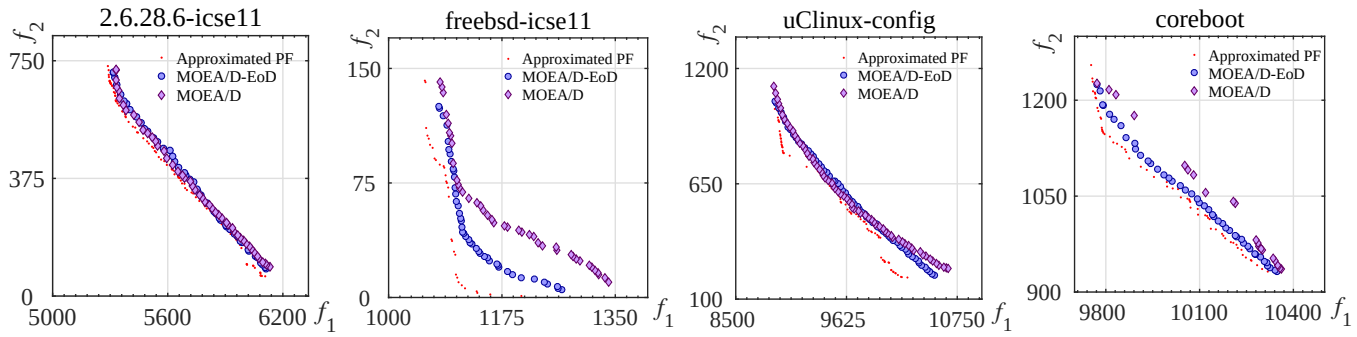


Fig. 3. Final solutions obtained by MOEA/D-EoD and MOEA/D on four representative feature models ($m = 2$).

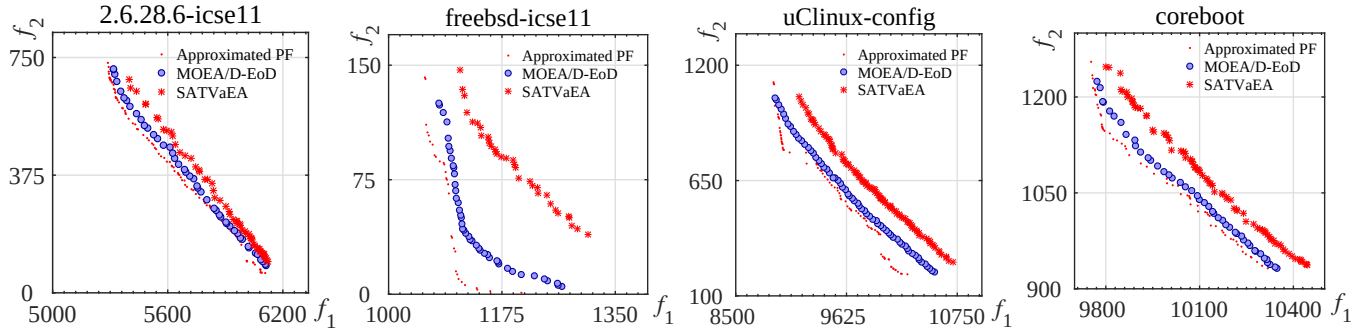


Fig. 4. Final solutions obtained by MOEA/D-EoD and SATVaEA on four representative feature models ($m = 2$).

both TCHE1 and TCHE2. As for PBI, the performance of MOEA/D-EoD is better than that of MOEA/D on 5 out of 7 models. Regarding the IGD+ values, as shown in Table II, they are in general consistent with the HV results, indicating again an improvement of MOEA/D-EoD over MOEA/D. To intuitively compare the two algorithms, we show the final solutions obtained by both algorithms on four representative FMs. As seen from Fig. 3, MOEA/D-EoD and MOEA/D perform similarly on 2.6.28.6-icse11 regarding both convergence and diversity. It is clear that the solutions of MOEA/D-EoD converge much better than those of MOEA/D on freebsd-icse11. For uClinux-config, the solutions found by MOEA/D-EoD are slightly closer to the PF than those obtained by MOEA/D, particulary at the bottom-right corner of the front. It is observed that MOEA/D-EoD significantly outperforms MOEA/D on the coreboot model concerning both convergence and diversity of the final solutions.

TABLE III
AVERAGE HV AND IGD+ OBTAINED BY MOEA/D-EOD (TCHE1) AND SATVAEA IN THE 2-OBJECTIVE OSPS INSTANCES.

FM	HV		IGD+	
	MOEA/D-EoD	SATVaEA	MOEA/D-EoD	SATVaEA
2.6.28.6-icse11	$7.215E-01$	$4.189E-01$	$3.120E-02$	$9.647E-02$
freebsd-icse11	$8.291E-01$	$1.797E-01$	$1.165E-01$	$5.100E-01$
uClinux-config	$7.204E-01$	$3.519E-01$	$6.689E-02$	$1.981E-01$
buildroot	$7.207E-01$	$4.273E-01$	$3.733E-02$	$1.012E-01$
freetz	$7.070E-01$	$4.712E-01$	$2.118E-02$	$4.390E-02$
coreboot	$7.711E-01$	$4.011E-01$	$5.884E-02$	$1.800E-01$
2.6.32-2var	$6.932E-01$	$3.222E-01$	$2.035E-02$	$9.943E-02$
2.6.33.3-2var	$6.784E-01$	$3.128E-01$	$2.480E-02$	$8.708E-02$

Next, the MOEA/D-EoD is compared with the latest SATVaEA. The average HV and IGD+ results are given in Table III, where we can find that MOEA/D-EoD performs much better than SATVaEA on all the models considered in terms of both HV and IGD+. As clearly shown in Fig. 4, the solutions of MOEA/D-EoD converge significantly better than those of SATVaEA on all the four models chosen.

TABLE IV
AVERAGE HV OBTAINED BY MOEA/D (WITH FOUR SCALARIZING FUNCTIONS) IN THE 3-OBJECTIVE OSPS INSTANCES.

	WS	TCHE1	TCHE2	PBI
toybox	$7.085E-01$	$8.232E-01$	$8.129E-01$	$8.281E-01$
axTLS	$7.120E-01$	$7.685E-01$	$7.607E-01$	$7.818E-01$
fiasco	$4.911E-01$	$7.105E-01$	$6.980E-01$	$7.053E-01$
uClinux	$7.969E-01$	$8.665E-01$	$8.635E-01$	$8.764E-01$
busybox-1.18.0	$5.151E-01$	$6.354E-01$	$6.156E-01$	$6.533E-01$
2.6.28.6-icse11	$3.127E-01$	$6.376E-01$	$6.073E-01$	$6.529E-01$
freebsd-icse11	$7.327E-01$	$8.034E-01$	$7.689E-01$	$7.847E-01$
uClinux-config	$5.267E-01$	$6.836E-01$	$6.618E-01$	$7.112E-01$
buildroot	$4.137E-01$	$6.477E-01$	$6.249E-01$	$6.839E-01$
freetz	$3.655E-01$	$5.636E-01$	$5.500E-01$	$5.907E-01$
coreboot	$4.907E-01$	$6.464E-01$	$6.305E-01$	$6.953E-01$
2.6.32-2var	$2.262E-01$	$5.466E-01$	$5.338E-01$	$5.448E-01$
2.6.33.3-2var	$2.163E-01$	$5.378E-01$	$5.314E-01$	$5.429E-01$

The above observations lead us to the following conclusion: The proposed MOEA/D-EoD performs in general better than MOEA/D in these large-scale 2-objective OSPS instances, regardless of the used scalarizing functions and reference points. In addition, MOEA/D-EoD overwhelmingly outperforms SATVaEA, particularly in terms of the convergence of the returned solutions (see Fig. 4).

3) Results on 3-objective OSPS instances: According to Table IV, MOEA/D using PBI can obtain better HV results than using WS, TCHE1 and TCHE2 in the 3-objective OSPS instances. Therefore, only PBI is used in the following comparative experiments, where MOEA/D-EoD is compared with

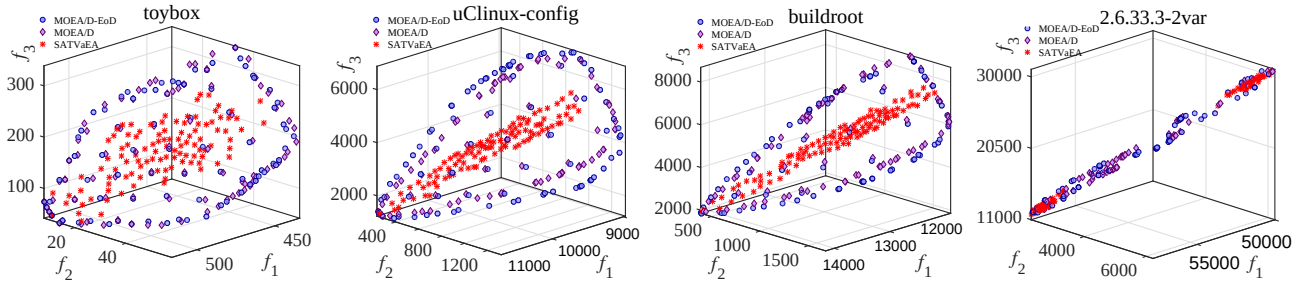


Fig. 5. Final solutions obtained by MOEA/D-EoD, MOEA/D and SATVaEA in four representative 3-objective OSPS instances.

both MOEA/D and SATVaEA. Table V gives the average HV results obtained by the three algorithms in all the 3-objective OSPS instances. Compared with MOEA/D, MOEA/D-EoD performs equivalently on 6 models, i.e., toybox, axTLS, fiasco, freetz, 2.6.32-2var and 2.6.33.3-2var, but significantly better on the remaining 7 FMs. In comparison with SATVaEA, however, MOEA/D-EoD performs significantly better on all the models considered.

As shown in Fig. 5, MOEA/D-EoD and MOEA/D can obtain similar solution sets on the small-scale toybox model. However, MOEA/D-EoD is able to find more solutions on the boundary for the two large-scale models, i.e., uClinux-config and buildroot. As for the model 2.6.33.3-2var, the solution sets found by MOEA/D-EoD and MOEA/D have no significant differences concerning the distribution of solutions. It is observed that the final solutions found by SATVaEA are mainly distributed either in the central region of the approximated PF (e.g., on toybox, uClinux-config and buildroot), or at some extreme parts (e.g., on 2.6.33.3-2var). The poor diversity of the solutions would be the main reason why SATVaEA is significantly worse than MOEA/D-EoD concerning the HV results as shown in Table V.

TABLE V
AVERAGE HV OBTAINED BY MOEA/D-EOD (PBI), MOEA/D (PBI) AND SATVAEA IN THE 3-OBJECTIVE OSPS INSTANCES.

FM	MOEA/D-EoD	MOEA/D	SATVaEA
toybox	$8.259E-01$	$8.281E-01\ddagger$	$6.930E-01\bullet$
axTLS	$7.840E-01$	$7.818E-01\ddagger$	$6.516E-01\bullet$
fiasco	$7.026E-01$	$7.053E-01\ddagger$	$6.440E-01\bullet$
uClinux	$8.797E-01$	$8.764E-01\bullet$	$7.027E-01\bullet$
busybox-1.18.0	$6.656E-01$	$6.533E-01\bullet$	$5.509E-01\bullet$
2.6.28.6-icse11	$6.593E-01$	$6.529E-01\bullet$	$5.913E-01\bullet$
freebsd-icse11	$8.231E-01$	$7.847E-01\bullet$	$6.523E-01\bullet$
uClinux-config	$7.357E-01$	$7.112E-01\bullet$	$5.995E-01\bullet$
buildroot	$6.970E-01$	$6.839E-01\bullet$	$6.042E-01\bullet$
freetz	$5.973E-01$	$5.907E-01\ddagger$	$5.535E-01\bullet$
coreboot	$7.684E-01$	$6.953E-01\bullet$	$6.612E-01\bullet$
2.6.32-2var	$5.488E-01$	$5.448E-01\ddagger$	$3.923E-01\bullet$
2.6.33.3-2var	$5.415E-01$	$5.429E-01\ddagger$	$3.803E-01\bullet$

We notice that MOEA/D can get comparable performance to MOEA/D-EoD on small-scale feature models, e.g., toybox, fiasco, and on some large-scale ones, e.g., 2.6.32-2var and 2.6.33.3-2var. In fact, these small-scale feature models are relatively easy, and can be well handled by both algorithms. For example, as shown in the first plot of Fig. 5, both MOEA/D-EoD and MOEA/D can find a high-quality solution set. For the large-scale 2.6.32-2var and 2.6.33.3-2var models, they both represent the Linux kernel configuration options for the x86 architecture[6]. Therefore, they share similarities in the feature model trees [66]. As shown in the fourth plot of Fig. 5, the PF of the OSPS problem constructed based on 2.6.33.3-2var may be narrow, which could make it difficult to distinguish between solution sets found by the two algorithms.

4) Results in 4-objective OSPS instances: Similarly, the HV and IGD+ results in the 4-objective OSPS instances are given in Table VI. Regarding HV, MOEA/D-EoD performs better than, equivalently to and worse than MOEA/D on 7, 1 and 5 out of 13 feature models, respectively. It is worth mentioning that comparable or even better results are obtained by MOEA/D mainly on the first three small-scale models, or on those large-scale models representing the Linux kernel configuration options for the x86 architecture (i.e., 2.6.28.6-icse11, 2.6.32-2var and 2.6.33.3-2var). Compared with SATVaEA, according to Table VI, we can find that MOEA/D-EoD performs significantly better on all the models except for coreboot.

Regarding IGD+, MOEA/D-EoD performs better than or equivalently to MOEA/D on all the feature models, except for 2.6.33.3-2var. Moreover, compared with SATVaEA, MOEA/D-EoD shows a significant improvement on all the models except for coreboot, on which the two algorithms yield similar performance.

V. FURTHER DISCUSSIONS

In the previous section, we have shown that the EoD operator can significantly improve MOEA/D on MOKP, mUBQP and OSPS problems, and that MOEA/D-EoD significantly outperforms SATVaEA on the OSPS problem considered in this study. In this section, we are going to answer the following three research questions (RQs).

- **RQ1**: How does the parameter τ affect the performance of the repair operator for the OSPS problem?
- **RQ2**: In the EoD operator, does historical information contribute more than neighboring information?
- **RQ3**: Can the EoD operator improve other decomposition-based algorithms in addition to MOEA/D?

A. Answers to RQ1

To answer this question, we choose MOEA/D-EoD (TCHE1+Ideal) as an exemplary implementation, and change

[6]In fact, the 2.6.28.6-icse11 model also represents the Linux kernel configuration options [30].

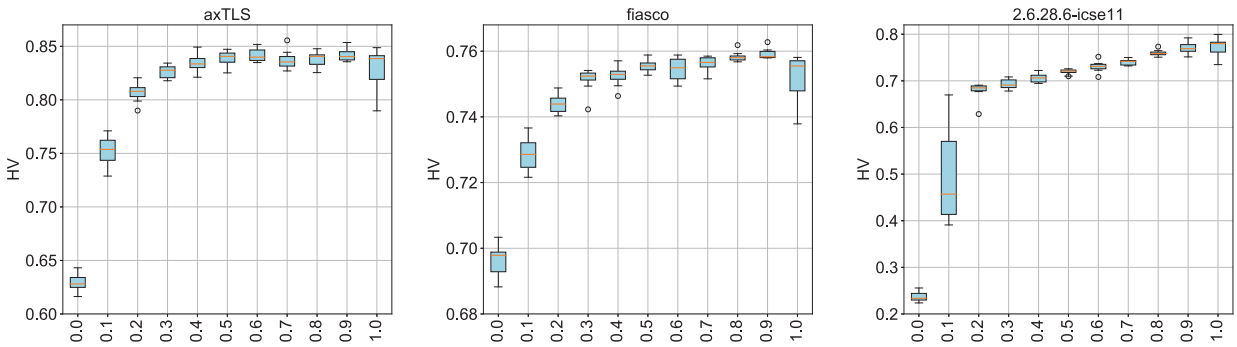


Fig. 6. The HV values, shown in boxplots, are obtained by MOEA/D-EoD (TCHE1+Ideal) under different values of τ.

TABLE VI
THE AVERAGE HV AND IGD+ FOR MOEA/D-EOD, MOEA/D AND SATVAEA ON THE 4-OBJECTIVE OSPS PROBLEMS.

FM	HV			IGD+		
	MOEA/D-EoD	MOEA/D	SATVaEA	MOEA/D-EoD	MOEA/D	SATVaEA
toybox	$6.479E-01$	$6.507E-01\circ$	$5.355E-01\bullet$	$3.220E-02$	$3.136E-02\ddagger$	$9.283E-02\bullet$
axTLS	$6.155E-01$	$6.203E-01\circ$	$4.938E-01\bullet$	$3.432E-02$	$3.445E-02\ddagger$	$1.057E-01\bullet$
fiasco	$5.661E-01$	$5.681E-01\circ$	$5.056E-01\bullet$	$2.021E-02$	$2.117E-02\ddagger$	$4.885E-02\bullet$
uClinux	$6.862E-01$	$6.802E-01\bullet$	$5.401E-01\bullet$	$1.728E-02$	$2.066E-02\bullet$	$1.334E-01\bullet$
busybox-1.18.0	$5.773E-01$	$5.647E-01\bullet$	$4.738E-01\bullet$	$2.970E-02$	$3.301E-02\bullet$	$9.412E-02\bullet$
2.6.28.6-icse11	$4.996E-01$	$5.060E-01\circ$	$4.809E-01\bullet$	$2.410E-02$	$2.283E-02\ddagger$	$4.607E-02\bullet$
freebsd-icse11	$6.170E-01$	$5.687E-01\bullet$	$4.115E-01\bullet$	$3.513E-02$	$5.130E-02\bullet$	$1.488E-01\bullet$
uClinux-config	$5.887E-01$	$5.611E-01\bullet$	$4.920E-01\bullet$	$2.108E-02$	$3.167E-02\bullet$	$8.394E-02\bullet$
buildroot	$5.437E-01$	$5.334E-01\bullet$	$4.895E-01\bullet$	$1.814E-02$	$2.190E-02\bullet$	$6.331E-02\bullet$
freetz	$5.136E-01$	$5.100E-01\bullet$	$4.917E-01\bullet$	$1.370E-02$	$1.504E-02\bullet$	$3.611E-02\bullet$
coreboot	$3.986E-01$	$2.309E-01\bullet$	$4.888E-01\circ$	$3.452E-02$	$5.064E-02\bullet$	$2.735E-02\ddagger$
2.6.32-2var	$4.669E-01$	$4.731E-01\ddagger$	$3.183E-01\bullet$	$1.501E-02$	$1.498E-02\ddagger$	$1.001E-01\bullet$
2.6.33.3-2var	$4.653E-01$	$4.760E-01\circ$	$3.191E-01\bullet$	$2.107E-02$	$1.571E-02\circ$	$1.080E-01\bullet$

τ from 0.0 to 1.0 with a step size 0.1. Each of the values is tested in three 2-objective OSPS instances, i.e., axTLS, fiasco and 2.6.28.6-icse11. The three models are picked because they have different termination conditions, being 6 seconds, 30 seconds and 200 seconds, respectively. Fig. 6 shows the HV results over 30 runs in the form of boxplots. According to Fig. 6 (a) and (b), the HV increases obviously as τ is changed from 0.0 to 0.5, and keeps steady when $\tau \in \{0.5, \ldots, 0.9\}$, but decreases when τ is switch from 0.9 to 1.0. For the 2.6.28.6-icse11 model, it is observed from Fig. 6 (c) that the median of the HV values tends to rise as τ increases. Compared with $\tau = 0.9$, however, the variation range for $\tau = 1.0$ is relatively larger.

According to the above discussions, large values of τ are preferred in the repair operator, suggesting that more computational resources should be given to the probSAT than to the SAT4J. This is in fact consistent with the findings in [30]. Since $\tau = 0.9$ performs well in general, we recommend this parameter setting in practice.

B. Answers to RQ2

In the EoD operator, the probability vector is learnt from historical information of each subproblem [see (14)]. In most of the previous works, however, the probability vector is constructed based on neighboring information. For example, the component of $\boldsymbol{p}_i$ in [27] is calculated as follows.

$$p_{i_k} = \frac{\sum_{j=1}^{T} x_k^{i_j} + \xi}{T + 2\xi}, \tag{19}$$

where $x_k^{i_j}$ denotes the k-th gene of the i_j subproblem (i.e., the j-th neighbor of the i-th subproblem), and ξ is a small value defined by $\xi = \frac{T \cdot s}{n-2s}$. Here, s is a control parameter, which is set to 0.4 [27]. According to (19), p_{i_k} is determined by the neighboring information of the i-th subproblem.

Table VII gives the IGD+ results obtained by MOEA/D-EoD (TCHE1+Ideal), in which the probability vector is learnt from either the historical information [(14)] or the neighboring information [(19)]. Compared with the MOEA/D-EoD using neighboring information, the algorithm using historical information performs significantly better in all the MOKP instances, all the 2-objective mUBQP instances, and 9 out of 13 OSPS instances. Note that historical information-based MOEA/D-EoD is inferior to the neighboring information-based algorithm in only three small-scale OSPS instances, i.e., toybox, axTLS and fiasco.

The above results clearly suggest that the historical information indeed contributes more than the neighboring information in the EoD operator. Since the neighbor size is relatively small, the probability vector constructed based on neighboring information is less likely to estimate accurately the real distribution of the solutions. This could be one possible reason for the ineffectiveness of using neighboring information.

C. Answers to RQ3

In Section III-B2, we have shown that EoD can be also integrated into NSGA-III. In this section, we will answer RQ3 by comparing NSGA-III-EoD with NSGA-III on both benchmark problems and OSPS problems. Due to space limitations, experimental results are not reported here, but in Section III

TABLE VII
AVERAGE IGD+ OBTAINED BY MOEA/D-EOD (TCHE1+IDEAL) USING HISTORICAL INFORMATION AND NEIGHBORING INFORMATION.

Problem	Instance	m	Historical-inf	Neighboring-inf
MOKP	2-500	2	$1.526E-02$	$7.519E-01\bullet$
	4-500	4	$8.701E-02$	$6.364E-01\bullet$
	6-500	6	$1.353E-01$	$7.192E-01\bullet$
	8-500	8	$1.383E-01$	$8.183E-01\bullet$
	10-500	10	$1.398E-01$	$1.063E+00\bullet$
mUBQP	(-0.5,1000)	2	$4.315E-02$	$3.461E-01\bullet$
	(-0.2,1000)	2	$6.846E-02$	$5.761E-01\bullet$
	(0.0,1000)	2	$6.291E-02$	$8.435E-01\bullet$
	(0.2,1000)	2	$7.879E-02$	$1.239E+00\bullet$
	(0.5,1000)	2	$1.089E-01$	$2.611E+00\bullet$
	(-0.5,2000)	2	$6.624E-02$	$3.552E-01\bullet$
	(-0.2,2000)	2	$7.258E-02$	$6.394E-01\bullet$
	(0.0,2000)	2	$1.125E-01$	$9.169E-01\bullet$
	(0.2,2000)	2	$1.801E-01$	$1.674E+00\bullet$
	(0.5,2000)	2	$3.370E-01$	$4.307E+00\bullet$
	(-0.5,3000)	2	$7.293E-02$	$3.927E-01\bullet$
	(-0.2,3000)	2	$9.749E-02$	$6.794E-01\bullet$
	(0.0,3000)	2	$1.227E-01$	$9.725E-01\bullet$
	(0.2,3000)	2	$1.844E-01$	$1.413E+00\bullet$
	(0.5,3000)	2	$5.188E-01$	$8.232E+00\bullet$
	(-0.5,4000)	2	$7.766E-02$	$3.486E-01\bullet$
	(-0.2,4000)	2	$1.108E-01$	$5.973E-01\bullet$
	(0.0,4000)	2	$1.233E-01$	$8.021E-01\bullet$
	(0.2,4000)	2	$2.609E-01$	$1.800E+00\bullet$
	(0.5,4000)	2	$5.840E-01$	$5.241E+00\bullet$
OSPS	toybox	3	$4.939E-02$	$4.031E-02\circ$
	axTLS	3	$6.195E-02$	$4.955E-02\circ$
	fiasco	3	$3.670E-02$	$3.251E-02\circ$
	uClinux	3	$4.217E-02$	$4.959E-02\bullet$
	busybox-1.18.0	3	$4.472E-02$	$4.587E-02\bullet$
	2.6.28.6-icse11	3	$3.099E-02$	$3.267E-02\bullet$
	freebsd-icse11	3	$5.418E-02$	$6.254E-02\bullet$
	uClinux-config	3	$4.428E-02$	$4.565E-02\bullet$
	buildroot	3	$3.695E-02$	$3.680E-02\ddagger$
	freetz	3	$2.390E-02$	$2.451E-02\bullet$
	coreboot	3	$4.385E-02$	$4.761E-02\bullet$
	2.6.32-2var	3	$2.285E-02$	$4.737E-02\bullet$
	2.6.33.3-2var	3	$2.056E-02$	$5.727E-02\bullet$

of the supplementary materials. It is observed from those results that NSGA-III-EoD performs better than, or at least equivalently to NSGA-III on all the MOKP, mUBQP and OSPS instances.

Therefore, the answer to RQ3 is clear: The EoD operator can indeed improve other decomposition-based algorithms in addition to MOEA/D. In fact, the NSGA-III is one of such algorithms.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we enhance the performance of decomposition-based multi-objective evolutionary algorithms by using an estimation of distribution update operator. In this operator, the probabilistic model (i.e., a probability vector) is learnt from historical information of each subproblem. This is different from most previous works where the probabilistic model is built based on neighboring solutions. Moreover, the learning factor in this model is adaptively adjusted during the evolutionary process, leading to the model encoding the uniform distribution initially and the distribution totally determined by the historical information eventually.

We show that the update operator can be easily integrated into two popular decomposition-based algorithms, MOEA/D and NSGA-III. Experimental results on benchmark problems indicate that this operator can indeed significantly improve the performance of the algorithms. Moreover, we apply the enhanced algorithms to the constrained optimal software product selection problem in the search-based software engineering. Different from prior works which handle only constraints related to the validity of software products, this paper takes into account those constraints specifying the demands of users. We give the formal definitions of the constrained optimal software product selection problem, for which a new SAT solvers-based repair operator is also suggested. The evaluations on a number of large-scale real-world instances demonstrate the effectiveness of the new algorithms on this practical problem. In addition, we find that it would be better to set the parameter in the repair operator to a relatively larger value, and that the historical information contributes more than neighboring information in the update operator.

Since the goal of this paper is to show that the performance gain can be achieved by using the estimation of distribution operator, we choose two popular decomposition-based algorithms (MOEA/D and NSGA-III) for exemplary implementations. It would be possible to integrate this operator into other decomposition-based algorithms. Currently, the basic versions of the two algorithms are adopted. In the future, it would be interesting to extend the new algorithms employing adaptive weight vectors [13], [50], as well as new mating selection and replacement principles [7], [51], [71]. Finally, it is of significance to apply the new algorithms to other similar practical problems in the area of search-based software engineering.

REFERENCES

[1] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[2] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450 – 455, 2014.

[3] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[4] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, Oct 2015.

[5] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[6] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 440–462, June 2017.

[7] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[8] S. Zapotecas-Martínez, B. Derbel, A. Liefooghe, H. E. Aguirre, and K. Tanaka, "Geometric Differential Evolution in MOEA/D: A Preliminary Study," in *Advances in Artificial Intelligence and Soft Computing*, G. Sidorov and S. N. Galicia-Haro, Eds. Cham: Springer International Publishing, 2015, pp. 364–376.

[9] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and Ant Colony," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, Dec 2013.

[10] W. Peng and Q. Zhang, "A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems," in *IEEE International Conference on Granular Computing*, Aug 2008, pp. 534–537.

[11] S. Z. Martinez and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in *The 13th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2011, pp. 69–76.

[12] N. A. Moubayed, A. Petrovski, and J. Mccall, "D^{2}MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces," *Evolutionary Computation*, vol. 22, no. 1, pp. 47–77, 2014.

[13] H. Li and D. Landa-Silva, "An adaptive evolutionary multi-objective approach based on simulated annealing," *Evolutionary Computation*, vol. 19, no. 4, pp. 561–595, Dec 2011.

[14] S. Zapotecas-Martínez, B. Derbel, A. Liefooghe, D. Brockhoff, H. E. Aguirre, and K. Tanaka, "Injecting CMA-ES into MOEA/D," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 783–790.

[15] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Pittsburgh, PA, USA, Tech. Rep., 1994.

[16] H. Mühlenbein and G. Paass, "From Recombination of Genes to the Estimation of Distributions I. Binary Parameters," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, ser. PPSN IV. London, UK, UK: Springer-Verlag, 1996, pp. 178–187.

[17] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111 – 128, 2011.

[18] R. Cheng, C. He, Y. Jin, and X. Yao, "Model-based evolutionary algorithms: a short survey," *Complex & Intelligent Systems*, vol. 4, no. 4, pp. 283–292, Dec 2018.

[19] A. Zhou, Q. Zhang, and G. Zhang, "A multiobjective evolutionary algorithm based on decomposition and probability model," in *2012 IEEE Congress on Evolutionary Computation*, June 2012, pp. 1–8.

[20] A. Zhou, Y. Zhang, G. Zhang, and W. Gong, "On neighborhood exploration and subproblem exploitation in decomposition based multiobjective evolutionary algorithms," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 1704–1711.

[21] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Generalized decomposition and cross entropy methods for many-objective optimization," *Information Sciences*, vol. 282, pp. 363–387, 2014.

[22] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, Dec 2015.

[23] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 682–691, Sept 2012.

[24] A. Zhou, F. Gao, and G. Zhang, "A decomposition based estimation of distribution algorithm for multiobjective traveling salesman problems," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1857 – 1868, 2013, iCNC-FSKD 2012.

[25] M. Zangari, A. Mendiburu, R. Santana, and A. Pozo, "Multiobjective decomposition-based Mallows Models estimation of distribution algorithm. A case of study for permutation flowshop scheduling problem," *Information Sciences*, vol. 397-398, pp. 137 – 154, 2017.

[26] Y. Li, A. Zhou, and G. Zhang, "A decomposition based estimation of distribution algorithm for multiobjective knapsack problems," in *2012 8th International Conference on Natural Computation*, May 2012, pp. 803–807.

[27] B. Wang, H. Xu, and Y. Yuan, "Scale adaptive reproduction operator for decomposition based estimation of distribution algorithm," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 2042–2049.

[28] M. Z. de Souza, R. Santana, A. T. R. Pozo, and A. Mendiburu, "MOEA/D-GM: using probabilistic graphical models in MOEA/D for solving combinatorial optimization problems," *CoRR*, vol. abs/1511.05625, 2015. [Online]. Available: http://arxiv.org/abs/1511.05625

[29] R. Santana, A. Ochoa, and M. R. Soto, "The mixture of trees factorized distribution algorithm," in *Genetic and Evolutionary Computation Conference (GECCO)*. San Francisco, CA: Morgan Kaufmann Publishers, 2001, pp. 543–550.

[30] Y. Xiang, Y. Zhou, Z. Zheng, and M. Li, "Configuring Software Product Lines by Combining Many-Objective Optimization and SAT Solvers," *ACM Transactions on Software Engineering and Methodology*, vol. 26, no. 4, pp. 14:1–14:46, Feb. 2018.

[31] A. S. Sayyad, T. Menzies, and H. Ammar, "On the value of user preferences in search-based software engineering: A case study in software product lines," in *2013 35th International Conference on Software Engineering (ICSE)*, May 2013, pp. 492–501.

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

[33] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Computer Engineering and Networks Laboratory,Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich,Switzerland, Tech. Rep., 2001.

[34] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th International Conference on Parallel Problem Solving from Nature, PPSN VIII*. Springer, 2004, pp. 832–842.

[35] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar, "Scalable product line configuration: A straw to break the camel's back," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2013, pp. 465–474.

[36] C. Henard, M. Papadakis, M. Harman, and Y. L. Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *The 37th International Conference on Software Engineering*, vol. 1, May 2015, pp. 517–528.

[37] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng, "SIP: Optimal Product Selection from Feature Models Using Many-Objective Evolutionary Optimization," *ACM Transactions on Software Engineering and Methodology*, vol. 25, no. 2, pp. 17:1–17:39, Apr. 2016.

[38] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 348–65, June 2014.

[39] Y. Xiang, Y. Zhou, M. Li, and Z. Chen, "A vector angle based evolutionary algorithm for unconstrained many-objective problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 131–152, Feb. 2017.

[40] P. Clements and L. Northrop, *Software product lines: practices and patterns*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[41] D. Batory, "Feature models, grammars, and propositional formulas," in *Proceedings of the 9th International Conference Software Product Lines, SPLC 2005*, H. Obbink and K. Pohl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 7–20.

[42] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," CMU/SEI-90-TR-21. SEI. Georgetown University, Tech. Rep., 1990.

[43] T. Berger, D. Lettner, J. Rubin, P. Grünbacher, A. Silva, M. Becker, M. Chechik, and K. Czarnecki, "What is a feature?: a qualitative study of features in industrial software product lines." in *Proceedings of the 19th International Conference on Software Product Line (SPLC)*, 2015, pp. 16–25.

[44] K. Czarnecki and U. Eisenecker, *Generative programming: Methods, tools, and applications*. Addison-Wesley, 2000.

[45] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[46] R. Saborido, A. B. Ruiz, and M. Luque, "Global WASF-GA: an evolutionary algorithm in multiobjective optimization to approximate the whole pareto optimal front," *Evolutionary computation*, 2016.

[47] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, April 2015.

[48] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[49] M. Z, *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[50] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, June 2014.

[51] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching based selection in evlutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2014.

[52] A. Balint and U. Schöning, *Choosing Probability Distributions for Stochastic Local Search and the Role of Make versus Break*. Berlin, Heidelberg: International Conference on Theory and Applications of Satisfiability Testing, 2012, pp. 16–29.

[53] D. L. Berre and A. Parrain, "The Sat4j library, release 2.2, system description," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59–64, 2010.

[54] Marques-Silva, J.P, Sakallah, and K.A, "GRASP: a search algorithm for propositional satisfiability," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[55] N. Eén and N. Sörensson, "An Extensible SAT-solver," in *International conference on theory and applications of satisfiability testing*. Springer Berlin Heidelberg, 2003, pp. 502–518.

[56] A. Biere, "PicoSAT Essentials," *Journal on Satisfiability Boolean Modeling & Computation*, vol. 4, no. 2C4, pp. 75–97, 2008.

[57] B. Selman, H. A. Kautz, and B. Cohen, "Noise strategies for improving local search," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, ser. AAAI '94. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1994, pp. 337–343.

[58] F. Lardeux, F. Saubion, and J. K. Hao, "GASAT: A Genetic Local Search Algorithm for the Satisfiability Problem," *Evolutionary Computation*, vol. 14, no. 2, pp. 223–253, 2006.

[59] C. Luo, S. Cai, K. Su, and W. Huang, "CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability," *Artificial Intelligence*, vol. 243, pp. 26 – 44, 2017.

[60] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon, "Multi-objective test generation for software product lines," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: ACM, 2013, pp. 62–71.

[61] T. H. Tan, Y. Xue, M. Chen, J. Sun, Y. Liu, and J. S. Dong, "Optimizing selection of competing features via feedback-directed evolutionary algorithms," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA 2015)*, 2015, pp. 246–256.

[62] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.

[63] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[64] H. Sato, "Inverted PBI in MOEA/D and its impact on the search performance on multi and many-objective optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 645–652.

[65] A. Liefooghe, S. Verel, and J.-K. Hao, "A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming," *Applied Soft Computing*, vol. 16, pp. 10 – 19, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613004079

[66] S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki, "Reverse engineering feature models," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 461–470.

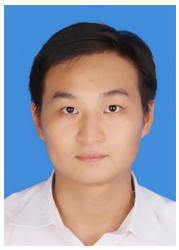
[67] R. Zabih and D. Mcallester, "A rearrangement search strategy for determining propositional satisfiability," in *National Conference on Artificial Intelligence*, 1988, pp. 155–160.

[68] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems," in *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*. IEEE, 2014, pp. 170–177.

[69] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified Distance Calculation in Generational Distance and Inverted Generational Distance," in *Evolutionary Multi-Criterion Optimization*. Cham: Springer International Publishing, 2015, pp. 110–125.

[70] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *Evolutionary Multi-Criterion Optimization*, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 862–876.

[71] S. Jiang and S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts," *IEEE transactions on cybernetics*, vol. 46, no. 2, pp. 421–437, 2016.



Yi Xiang 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 working with Prof. X. W. Yang within 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 designs and analyses of 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.

Yuren Zhou received the B. Sc. degree in mathematics from Peking University, Beijing, China, in 1988, the M.Sc. degree in the mathematics from Wuhan University, Wuhan, China, in 1991, and the Ph.D. degree in computer science from the same University in 2003. He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, P. R. China. His current research interests include design and analysis of algorithms, evolutionary computation, and social networks.

Han Huang (M'15) received the B.Man. degree in Information Management and Information System, in 2002, 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 evolutionary computation, swarm intelligence and their application. Dr. Huang is a senior member of CCF and member of IEEE.