Multi-Objective Integer Programming Approaches for Solving Optimal Feature Selection Problem

A New Perspective on Multi-Objective Optimization Problems in SBSE

Yinxing Xue
University of Science and Technology of China
Hefei, China
yinxingxue@gmail.com

Yan-Fu Li*
Department of Industrial Engineering, Tsinghua University
Beijing, China
liyanfu@tsinghua.edu.cn

ABSTRACT

The optimal feature selection problem in software product line is typically addressed by the approaches based on Indicator-based Evolutionary Algorithm (IBEA). In this study, we first expose the mathematical nature of this problem — multi-objective binary integer linear programming. Then, we implement/propose three mathematical programming approaches to solve this problem at different scales. For small-scale problems (roughly, less than 100 features), we implement two established approaches to find all exact solutions. For medium-to-large problems (roughly, more than 100 features), we propose one efficient approach that can generate a representation of the entire Pareto front in linear time complexity. The empirical results show that our proposed method can find significantly more non-dominated solutions in similar or less execution time, in comparison with IBEA and its recent enhancement (i.e., IBED that combines IBEA and Differential Evolution).

KEYWORDS

Optimal Feature Selection Problem, Multi-Objective Optimization (MOO), Multi-Objective Integer Programming (MOIP)

ACM Reference Format:

Yinxing Xue and Yan-Fu Li. 2018. Multi-Objective Integer Programming Approaches for Solving Optimal Feature Selection Problem. In *ICSE '18: ICSE '18: 40th International Conference on Software Engineering , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3180155.3180257

1 INTRODUCTION

In order to cater for the demands from various customers, many similar yet different products are developed and maintained (e.g., similar product variants from a software product family). Industries usually need to built the similar products in a systematic and reuse-based way — via a software product line (SPL) to reduce development costs, shorten product time to market, and improve product quality and flexibility [23].

SPLE is feature-oriented, as it adopts feature-oriented domain analysis [24] for requirements analysis and builds core assets architecture for reuse [9]. *Feature* refers to the modularizable program functionality in development. One fundamental task in SPL is to select features that meet the requirements of customers, avoid possible feature conflicts, and meanwhile optimize design goals in product configuration. Hence, given customer requirements, it is greatly helpful to guide the vendors to make automated decision on selecting optimal features.

In practice, as real-world SPLs contain thousands of features, manual feature selection for product configuration according to the given requirements and constraints is extremely difficult if not impossible. For example, Linux X86 kernel contains 6888 features, and 343944 constraints, as reported in [43]. Further, features are usually associated with quality attributes such as cost, defect and reliability. Owing to such complexity, it is extremely hard for the vendor to select a set of features that complies with requirement constraints yet optimizes the quality attributes according to user preferences. This is called the *optimal feature selection* problem [15].

In literatures, the optimal feature selection problem has been studied by several researchers. Two earlier studies adopted Filtered Cartesian Flattening (FCF) [48] or Genetic Algorithm (GA) [15]. In 2013, MOEAs were applied to solve this problem and IBEA was reported to be the best MOEA [42]. Along this line, studies of improvements on IBEA [41][45] and integration of IBEA with other techniques (e.g., constraint solving [18] and Differential Evolution (DE) [50]) are the latest progress. Among them, IBED [50] adopts a dual-population design (one population for IBEA and another for DE).Meanwhile, a trend is to combine IBEA with constraint solvers (e.g., SATIBEA [18] and SMTIBEA [14]) to improve the repair operator in the population evolution. Hence, *MOEAs (especially IBEA) become the indispensable parts of the state-of-the-art approaches.*

Generally, IBEA and other MOEAs are scalable. However, is IBEA really an indispensable part for a method to solve such SBSE problem? As one heuristic and non-deterministic algorithms, MOEA-based approaches at least face the following four drawbacks:

(1) Algorithm convergence. Customization of standard MOEA operators may fail the convergence of the MOEAs [27].
(2) Even on small problems, non-guarantee of solution set completeness.
(3) Non-guarantee of finding the true Pareto-optimal solutions.

*Yan-Fu Li is the corresponding author of the paper.

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05…$15.00
https://doi.org/10.1145/3180155.3180257

(4) Non-guarantee of finding evenly-distributed solutions [27], while retaining the correctness of solutions (e.g., Diversity-preferred NSGA-II cannot achieve correctness [42]).

All these drawbacks are inherent in the nature of heuristic algorithms. To circumvent them, is it feasible to solve the optimal feature selection problem from a different perspective? Let us revisit the nature of this SBSE problem — decision variables are binary and all constraints and objectives are linear (see §2.2 and §3). Hence, integer programming (IP), more specifically integer linear programming, is applicable for this problem as long as multiple objectives can be reduced to one.

In *epsilon*-constraint (ϵ-constraint) method [17], the straightforward idea is to convert the objectives 1 to $k-1$ into the range constraints and use the k-th objective as the objective function of IP. The operation procedure has several iterations, each increases (or decreases) the right-hand side of the constrained-objective by a step and runs IP method. The ϵ-constraint method and its improvement CWMOIP (Constraint Weighted Multi-objective Integer Programming) [37] are elaborated in §4. During the evaluation, we find that the brute-force iteration of all objectives' ranges will call IP too many times, make solving running forever. This indicates:

"Existing MOIP methods (ϵ-constraint and CWMOIP) are capable on small systems from SPLOT [31], but not on large ones"

Technical Innovation. For large problem, it is practical and meaningful to generate a good representation of the Pareto front. Therefore, we propose to combine the hit-and-run (H&R) method [44] with normal constraints (NC) method [32]. NC method, was originally proposed for generating Pareto front representation for continuous optimization problem. H&R is efficient in randomly generating uniformly-distributed points inside any bounded region. Due to the high dimensionality and complex boundaries of the utopia plane (see definition in §5), we propose to improve NC method by using H&R. The essential idea is to use uniformly-distributed reference points on the utopia plane to generate representative Pareto-optimal solutions (§5). The advantages of our method are three-fold: 1). guarantee the true optimal solution; 2). guarantee the spread of the Pareto-front; 3). computationally efficient. We call our method SolRep; 4) uniformity of the Pareto-front.

We compare the state-of-the-art MOEA for optimal feature selection (i.e., IBED [50]) with IP-based methods on SPLOT [31] and LVAT [1] repositories. Results show that, in most cases, SolRep finds significantly more non-dominated solutions than IBED in similar or less time (§6). Thus, we make *the first attempt* to develop and apply MOIP on large systems (e.g., Linux X86). A take-home message is

"The improved IP method (NC+H&R) can scale up to large systems (e.g., Linux X86 from LVAT), being better than IBEA and IBED."

Our main contributions are summarized below.

(1) By converting logical formulae into inequalities, we formulate this problem as a multiple-objective IP model.
(2) We apply ϵ-constraint method and its improvement CWMOIP to find complete solutions on small systems.
(3) We propose an improved IP-based method, SolRep, to achieve the scalability and effectiveness. SolRep is published at [3].
(4) The evaluation proves the effectiveness and efficiency of SolRep. Especially on *Linux X86*, most solutions of IBED are dominated by solutions of SolRep.

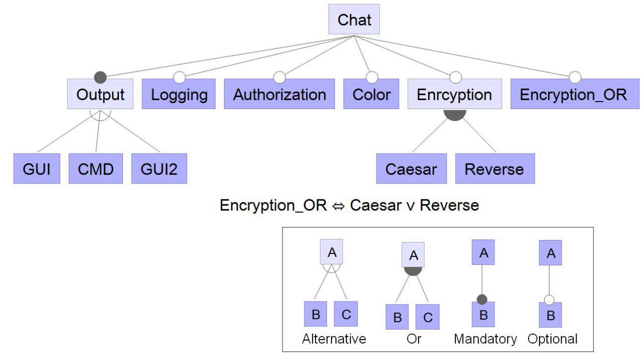

Figure 1: The feature model of $\mathcal{JCS}$ from [45]

2 OPTIMAL FEATURE SELECTION PROBLEM IN SPL

In this section, we briefly introduce the optimal feature selection problem and its formulation as a MOO problem.

2.1 Feature Model and the Constraints inside

In SE, *feature* is defined as "A distinguishing characteristic of a software item (e.g., performance or functionality, etc.)" in *IEEE Std* 829-1998. In SPL, *feature* further refers to the requirements and implementations of such a characteristic [24]. *Feature model* is a hierarchy of features within a product family as well as the *structural* and *semantic* constraints in between [10].

Feature model organizes the features inside in a tree-like structure. Given a parent (or compound) feature f' and its child features (or subfeatures) $\{f_1, ..., f_n\}$, four types of *tree-structure* constraints (TCs) exist and are defined as follow [4]:

- f_i is a *mandatory* subfeature — $f_i \Leftrightarrow f'$,
- f_i is an *optional* subfeature — $f_i \Rightarrow f'$,
- $\{f_1, ..., f_n\}$ is an *or* subfeature group — $f_1 \vee ... \vee f_n \Leftrightarrow f'$,
- $\{f_1, ..., f_n\}$ is an *alternative* subfeature group — $(f_1 \vee f_2 \vee ... \vee f_n \Leftrightarrow f') \wedge \bigwedge_{1 \le i < j \le n} (\neg(f_i \wedge f_j))$.

Given two features f_1 and f_2 that hold no TCs, three types of cross-tree *semantic* constraints (CTCs) may exist, i.e.,*iff*, *requires* and *excludes* [4]:

- f_1 *iff* f_2 — $f_1 \Leftrightarrow f_2$,
- f_1 *requires* f_2 — $f_1 \Rightarrow f_2$,
- f_1 *excludes* f_2 — $\neg(f_1 \wedge f_2)$.

The feature model of a Java Chat System ($\mathcal{JCS}$) is illustrated in Fig. 1 and the constraints of that model are listed in Table 1. Constraints c(1) – c(12) are TCs, as they are constraints on structural relation of features. The root (compulsory) feature of the feature model is *Chat*, which has a mandatory subfeature (*Output*) and several optional subfeatures (e.g., *Encryption*) — constraints c(1) – c(7). Since the feature *Output* is mandatory, exactly one of its subfeatures (*GUI*, *CMD*, and *GUI2*) must be selected — constraints c(8) – c(11). In addition, if feature *Encryption* is selected, at least one of its subfeatures (*Caesar* and *Reverse*) needs to be selected — constraints c(12). There is only one CTC for $\mathcal{JCS}$, which is of the form f_1 *iff* f_2, *Encryption_OR* is selected if and only if *Caesar* or *Reverse* is selected — constraints c(13).

Table 1: Constraints of $\mathcal{JCS}$

Chat	c(1)
$Output \iff Chat$	c(2)
$Logging \implies Chat$	c(3)
$Authorization \implies Chat$	c(4)
$Color \implies Chat$	c(5)
$Encryption \implies Chat$	c(6)
$Encryption\_OR \implies Chat$	c(7)
$(GUI \lor CMD \lor GUI2) \iff Output$	c(8)
$\neg(GUI \land CMD)$	c(9)
$\neg(GUI \land GUI2)$	c(10)
$\neg(CMD \land GUI2)$	c(11)
$(Caesar \lor Reverse) \iff Encryption$	c(12)
$Encryption\_OR \iff (Caesar \lor Reverse)$	c(13)

Given a feature model M, we refer TCs and CTCs of M as the conjunction of constraints of M, denoted by $conj(M)$. Let $Fea(M)$ denote the feature set of M. Thus, for the features in $\mathcal{JCS}$, we have $Fea(\mathcal{JCS}) = \{Chat, Output, Logging, Authorization, Color, Encryption, Encryption\_OR, GUI, CMD, GUI2, Caesar, Reverse\}$.

Definition 1 (Correct Solution). *Given a feature model M, a* correct solution *for M refers to a non-empty feature set $F \subseteq Fea(M)$ that satisfies the constraints of M.*

Take $\mathcal{JCS}$ as an example, $F = \{Chat, Output, GUI\}$ is a correct solution of feature selection for $\mathcal{JCS}$.

2.2 Multi-objective Optimization for SPL

2.2.1 Software Development Model and the Objectives. As the correct selection of a non-empty feature set F on feature model M is essentially a boolean satisfiability (SAT) problem [29][5]. Gradually, this problem evolves into a MOO problem — more than having a correct selection, multiple quality attributes of the product according to F need to be optimized .

According to [2], goals in software development have a positive impact for the products. Project managers control the software development according to four objective scores: project risk; development cost, development defects; and manpower (total months of development) [2]. According to [42], each feature is associated with three attributes, i.e., COST, DEFECT and USED_BEFORE. The optimal feature selection problem can be ideally modeled as a MOO problem, which requires trade-off among multiple design goals.

2.2.2 Formalization of the MOO Problem. Given a non-empty feature set F and the entire feature set $Fea(M)$ for M ($Fea(M) = \{f_1 ... f_n\}$. For $\forall f_i \in Fea(M)$, the binary value represented by the selection of f_i (denoted as $|f_i|$) is **1** if $f_i \in F$, else $|f_i|$ is **0**. Given a solution $\vec{x}$ (an encoded binary-value vector for $Fea(M)$), according to [42], five objective functions are defined as follows:

Obj1. **Correctness** means the extent of the compliance to $conj(M)$. We want to minimize the number of violated constraints: $\mathcal{F}_1(\vec{x}) = Viol(F)$, where $Viol(F)$ returns the number of constraints violated by F among $conj(M)$. Note that $Viol(F) = 0$ means a correct selection.

Obj2. **Richness of features** means the richness of the functionality of the product. We want to minimize the number of deselected features: $\mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n}(1-|f_i|)$, where $|f_i|$ returns 1 if $f_i \in F$, otherwise returns 0.

Obj3. **Feature used before** means the reliability of product, as features never used are more prone to have unknown defects. We want to minimize this: $\mathcal{F}_3(\vec{x}) = \sum_{i=1}^{n}(Used(f_i) \cdot |f_i|)$, where $Used(f_i)$ returns a boolean value of the attribute USED_BEFORE of feature f_i.

Obj4. **Defects** means the known bugs or errors contained in the product. We want to minimize this: $\mathcal{F}_4(\vec{x}) = \sum_{i=1}^{n}(Defect(f_i) \cdot |f_i|)$, where $Defect(f_i)$ returns the value of attribute DEFECT of feature f_i.

Obj5. **Cost** means the expense and efforts in developing the product. We want to minimize this: $\mathcal{F}_5(\vec{x}) = \sum_{i=1}^{n}(Cost(f_i) \cdot |f_i|)$, where $Cost(f_i)$ returns the value of the attribute COST of feature f_i.

Rather than encode all the objectives into one magic weighted fitness function, all the objectives are equally treated and solved by MOO using the Pareto dominance relation [22].

A k-objective optimization problem could be written in the following form (in our case, $k = 5$):

$$Minimize\ \vec{\mathcal{F}} = (\mathcal{F}_1(\vec{x}), \mathcal{F}_2(\vec{x}), ..., \mathcal{F}_k(\vec{x})) \tag{1}$$

where $\vec{\mathcal{F}}$ is a k-dimensional objective vector and $\mathcal{F}_i(\vec{x})$ is the value of $\vec{\mathcal{F}}$ for i-th objective.

Definition 2. *Given two correct solutions $\vec{x}, \vec{y} \in B^n$ and an objective vector $\vec{\mathcal{F}} : B^n \to R^k$, $\vec{x}$ **dominates** $\vec{y}$ ($\vec{x} \prec \vec{y}$) if*

$$\forall i \in \{1, ..., k\} \quad \mathcal{F}_i(\vec{x}) \le \mathcal{F}_i(\vec{y}) \tag{2}$$

$$\exists j \in \{1, ..., k\} \quad \mathcal{F}_j(\vec{x}) < \mathcal{F}_j(\vec{y}) \tag{3}$$

otherwise $\vec{x} \nprec \vec{y}$

Definition 3. *Given a correct solution $\vec{x}$ and a set of correct solutions $S_{\vec{x}}$, $\vec{x}$ is **non-dominated** iff*

$$\forall \vec{x}_i \in S_{\vec{x}} \quad \vec{x}_i \nprec \vec{x} \tag{4}$$

$\vec{x}$ is called a *Pareto optimal* solution if $\vec{x}$ is *correct* and *not dominated* by any other correct solutions. All Pareto-optimal solutions are called as the true Pareto front.

2.2.3 Problem Statement. Existing MOEAs (e.g., IBEA [52], NSGA-II [11], ssNSGA-II [12], MOCell [35], SPEA2 [53]) are used to find a set of non-dominated solutions that approximate the Pareto front for solving the MOO problems. As pointed by Sayyad *et.al* in [42], IBEA produces the solutions with the highest correctness rate. After that, more studies improve IBEA to search for correct solutions [40][45][18][19][50].

As these MOEA-based approaches suffer from the correctness, why not we use a method naturally assuring the solution correctness? As all constraints in $conj(M)$ and all objectives are *linear* (convertible to linear inequalities, see §3), Binary Integer Programming (BIP) can be applied. In theory, we can reduce the multiple objectives into one. The BIP-based analytic method is free from the drawbacks of MOEAs (see §1), but suffer from the scalability issue.

3 MATHEMATICAL FORMULATION

To apply IP methods, logical formulae shown in Table 1 are converted into inequalities to serve as linear constraints in IP.

3.1 Theory of Converting Logical Formulae to Inequalities

Converting logical formulae (LF) into inequalities is a typical problem of OR. By BIP upon the inequalities, it can be rapidly decided whether the original given LFs can be satisfied and how to be satisfied if so. In [20], Hooker proposed and proved that the satisfiability problem of a conjunctive normal form (CNF) can be directly reduced to a BIP problem. Though detailed steps are not formalized in [20], the basic idea of conversion was explained. Generally, two ways can convert an arbitrary LF as a CNF. Except for each atomic proposition, if no extra variables are introduced for operations on several atomic propositions, the running time and length of the resulting CNF can increase exponentially with the number of atomic propositions in the original formula in the worse case [7].

3.2 Fast Converting Logical Formulae from the SPL Models

As different types of TCs and CTCs are not arbitrary LFs, the conversion can be done in linear time without introducing intermediate CNFs and extra variables. Let $|f_i|$ denote whether the i-th feature (f_i) is selected in a solution (see §2.2). We deduce these lemmas:

Lemma 3.1. *If feature f is an Optional subfeature of feature f', the linear inequality for the Optional relationship is*

$$|f'| - |f| \geq 0 \tag{5}$$

Proof. $f \Rightarrow f'$ is *true* according to the optional relationship. Thus we can infer $\neg f \vee f'$ is *true*. As a CNF, it can be converted to the inequality $(1 - |f|) + |f'| \geq 1$, that is $|f'| - |f| \geq 0$. □

Lemma 3.2. *If feature f is a Mandatory subfeature of feature f', the linear equality for the Mandatory relationship is*

$$|f'| - |f| = 0 \tag{6}$$

Proof. $f \Leftrightarrow f'$ is *true* according to the mandatory relationship. Thus we infer the CNF $(\neg f \vee f') \wedge (\neg f' \vee f)$ is *true*. Two inequalities are deduced: $(1-|f|)+|f'| \geq 1$ and $(1-|f'|)+|f| \geq 1$. By unifying them, we infer $|f'| - |f| = 0$. □

Lemma 3.3. *If features f_1 ... f_n are the Or-subfeature of feature f', the linear inequalities for this relationship are*

$$\forall i \in \{1, ..., n\},\ |f_i| - |f'| \leq 0 \tag{7}$$

$$\sum\nolimits_{i=1}^{n} |f_i| - |f'| \geq 0 \tag{8}$$

Proof. $\bigvee_{i=1}^{n}(f_i) \Leftrightarrow f'$ is *true* according to the Or relationship. Here, $\bigvee_{i=1}^{n}(f_i)$ notates $f_1 \vee f_2 \vee ... \vee f_n$. Thus, the formulae $\bigvee_{i=1}^{n}(f_i) \Rightarrow f'$ and $f' \Rightarrow \bigvee_{i=1}^{n}(f_i)$ need to be *true*. For the first formula, we can represent it as the CNF and get the resulting formula $\bigwedge_{i=1}^{n}(\neg f_i \vee f')$ to be *true*. Note that for an indexed set of propositions $P = \{p_1, ..., p_n\}$, $\bigwedge_{i=1}^{n}(p_i)$ means that each proposition in P needs to be *true*. Thus, we can get n derived linear inequalities that are listed in the inequality (7). For the second formula $f' \Rightarrow \bigvee_{i=1}^{n}(f_i)$, we get $\neg f' \vee \bigvee_{i=1}^{n}(f_i)$ to be *true*, that is $\neg f' \vee f_1 \vee ... \vee f_n$ to be true. So the inequality (8) can be deduced from the second formula. □

Lemma 3.4. *If features f_1 ... f_n are the Alternative subfeature of feature f', the linear inequalities for this relationship are*

$$\forall i \in \{1, ..., n\},\ |f_i| - |f'| \leq 0 \tag{9}$$

$$\sum\nolimits_{i=1}^{n} |f_i| - |f'| \geq 0 \tag{10}$$

$$\sum\nolimits_{i=1}^{n} |f_i| \leq 1 \tag{11}$$

Proof. Essentially, alternative subfeatures are a special type of *or* features. In the subfeatures of an *or* relationship, at least one subfeature needs to be selected. Whereas, only and exactly one needs to be selected in the alternative subfeatures. Instead of using inequalities c(9)–c(11) in Table 1, the concise inequality (11) is used to assure the exclusiveness of all subfeatures. □

Except the above 4 types of TCs, there are 3 types of CTCs. The *requirement* of f' for f can be formulated as $f \Rightarrow f'$, and we can deduce $|f'| - |f| \geq 0$ according to Lemma 3.1. Similarly, the *iff* relationship between f' and f is formulated as $f \Leftrightarrow f'$, and we can deduce $|f'| - |f| = 0$ according to Lemma 3.2. Last, we have the last type of CTCs:

Lemma 3.5. *If feature f Excludes feature f', the linear equality for the Exclusion relationship is*

$$|f'| + |f| \leq 1 \tag{12}$$

Proof. $f \Rightarrow \neg f'$ is *true* according to the exclusion relationship. Thus we can infer $\neg f \vee \neg f'$ is *true*. As a CNF, it can be converted to $(1 - |f|) + (1 - |f'|) \geq 1$, that is $|f'| + |f| \leq 1$. □

3.3 The MOIP Model of Our Example

Let $\vec{x}$ be a solution, a binary variable vector $\vec{x} \in \{0,1\}^n$ where variable x_i denotes $|f_i|$. For each feature f_i, we denote its attributes $Used(f_i)$, $Defect(f_i)$ and $Cost(f_i)$ as coefficient a_i, b_i and c_i, respectively. We denote the objective function for obj_j as $\mathcal{F}_j(\vec{x})$, $j \in \{1, ..., k\}$. Subject to the converted linear inequalities, the example is formulated as a multi-objective BIP (MOBIP) problem:

$$\begin{array}{ll} \text{Min} & \mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n}(1 - x_i) \\ \text{Min} & \mathcal{F}_3(\vec{x}) = \sum_{i=1}^{n}(a_i \cdot x_i) \\ \text{Min} & \mathcal{F}_4(\vec{x}) = \sum_{i=1}^{n}(b_i \cdot x_i) \\ \text{Min} & \mathcal{F}_5(\vec{x}) = \sum_{i=1}^{n}(c_i \cdot x_i) \\ \text{s.t.} & \text{the inequalities for } conj(M) \text{ hold} \end{array} \tag{13}$$

4 ϵ-CONSTRAINT METHOD AND CWMOIP

When the problem size is relatively small, the ϵ-constraint method and CWMOIP are two feasible methods to find the complete non-dominated solutions (a.k.a., true Pareto front).

4.1 ϵ-constraint Method

The idea is to make $k - 1$ objectives as the range constraints and use the k-th one as the objective function in BIP [17]. As *obj1* is correctness, in BIP, all the constraints are satisfied, *obj1* is always 0 in theory and thus $\mathcal{F}_1(\vec{x})$ is not needed. So $\mathcal{F}_2(\vec{x})$, $\mathcal{F}_3(\vec{x})$, $\mathcal{F}_4(\vec{x})$ can be converted to range constraints. Each range constraint's upper bound will be iterated from 0 to the upper bound of the corresponding objective, by step size 1.

The detailed procedures are shown in Algorithm 1. Note that $getObjTheoBound(M, \mathcal{F}_2)$ at line 2 finds the theoretic upper bound

Algorithm 1: Function $EpsilonCont()$ for feature selection

```
Input: M: the feature model of the given system
Output: solutions: a non-dominated solution set for feature
        selection
1  E ← ∅;
2  f_2^TUB = getObjTheoBound(M, F_2), f_2^TLB = 0;
3  f_3^TUB = getObjTheoBound(M, F_3), f_3^TLB = 0;
4  f_4^TUB = getObjTheoBound(M, F_4), f_4^TLB = 0;
5  for p = f_2^TLB; p ≤ f_2^TUB; p = p+1 do
6      for q = f_3^TLB; q ≤ f_3^TUB; q = q+1 do
7          for t = f_4^TLB; t ≤ f_4^TUB; t = t+1 do
8              allCons = conj(M) ∪ {F_2 ≤ p} ∪ {F_3 ≤
                 q} ∪ {F_4 ≤ t};
9              ME = bintprog(allCons, F_5);
10             E = E ∪ ME;
11 return E;
```

f_2^{TUB} for $\mathcal{F}_2$ — for *JCS*, it is 12 when all $x_i = 0$ and $conj(M)$ is not considered. Similarly, f_3^{TUB} and f_4^{TUB} are $\sum_{i=1}^{n} a_i$ and $\sum_{i=1}^{n} b_i$ respectively, when the size of feature set $n = |Fea(JCS)| = 12$ for *JCS*. At line 8, *allCons* is the union of the original inequalities of the formula (13) and three new inequalities converted from other constrained objectives in formula (14). At line 9, $bintprog(allCons, \mathcal{F}_5)$ calls the BIP function for objective $\mathcal{F}_5$ such that *allCons* are satisfied.

$$
\begin{array}{ll}
\text{Min} & \mathcal{F}_5(\vec{x}) = \sum_{i=1}^{n}(c_i \cdot x_i) \\
\text{s.t.} & \sum_{i=1}^{n}(1 - x_i) \le p \\
 & \sum_{i=1}^{n}(a_i \cdot x_i) \le q \\
 & \sum_{i=1}^{n}(b_i \cdot x_i) \le t \\
 & \text{the inequalities for } conj(M) \text{ hold}
\end{array} \tag{14}
$$

Algorithm 1 is of the time complexity of $O(n^3)$, if considering BIP solving function *bintprog*() takes constant time — a time limit is set in its practical usage. Solving the MOBIP problem in formula (13) is reduced to solving the BIP problem in formula (14) by many times — a number of $(n+1)(\sum_{i=1}^{n} a_i + 1)(\sum_{i=1}^{n} b_i + 1)$ times.

4.2 CWMOIP

CWMOIP, proposed by Özlen *et al.* [37], is an objective-reduction technique for MOIP. CWMOIP is for generating *all* non-dominated solutions. It improves the ϵ-constraint method by two steps: (1) for each objective, the lower bound is not 0 and the upper bound is not the sum of feature attributes revelent to that objective. To be precise, BIP is applied to get the true lower and upper bounds for each objective (i.e., $\mathcal{F}_2$ to $\mathcal{F}_5$) separately, subject to the conjunction of constraints $conj(M)$. (2) objective-reduction is implemented via the constraint weight method, to avoid generating dominated solutions. The k-objective problem is first reduced to that of $k-1$, then $k-2$, iteratively, until the last objective.

Example for (1). In the precise calculation of the upper and lower bounds, for the example of *JCS* with 12 features, the true bounds f_2^{LB} and f_2^{UB} for $\mathcal{F}_2(\vec{x})$ are 2 (a maximum of 10 selected features) and 9 (a minimum of 3 selected) respectively, not 0 and 12 in the ϵ-constraint method. Hence, in ϵ-constraint method 13 (12-0+1) times of iteration is needed for the outermost loop at line 5 in Algorithm 1, while only 8 (9-2+1) times is needed for CWMOIP.

Algorithm 2: Function *CWMOIP()* for k-objective IP

```
Input: k: the number of objectives, l_k: constrained value for the
       weighted k-th obj, X: the set of linear constraints
Output: E: the set of non-dominated solutions
1  E ← ∅;
2  f_2^UB, f_2^LB = getObjTrueBound(X, f_2);
3  ...//get true bounds for other objective 3 to k − 1;
4  f_k^UB, f_k^LB = getObjTrueBound(X, f_k);
5  w_k = 1 / ((f_2^UB − f_2^LB + 1)(f_3^UB − f_3^LB + 1)...(f_k^UB − f_k^LB + 1));
6  if k = 1 then
7      E = E ∪ bintprog(X, f_k);
8  while true do
9      f_1 = addObjFuncSuffix(f_1, w_k · f_k);
10     X′ = X ∪ {f_k ≤ l_k};
11     ME = CWMOIP(k − 1, l_k, X′);
12     if ME = Null then
13         break;
14     E = E ∪ ME;
15     l_k = Max(f_k(x⃗), x⃗ ∈ ME) − 1;
16 return E;
```

$$
\begin{array}{ll}
\text{Min} & f_1(x) \\
\text{Min} & f_2(x) \\
\text{s.t.} & x \in X
\end{array}
\quad (15)
\quad \Longrightarrow \quad
\begin{array}{ll}
\text{Min} & f_1(x) + w_2 \cdot f_2(x) \\
\text{s.t.} & x \in X \\
 & f_2(x) \le l_2 \\
 & w_2 = 1/(f_2^{UB} - f_2^{LB} + 1)
\end{array}
\quad (16)
$$

Figure 2: An example of applying CWMOIP for a bi-obj problem

Example for (2). Fig. 2 shows an example for objective-reduction. Solving the *bi*-objective problem in formula (15) is reduced to solving the 1-objective problem in formula (16) by l_2 times. It is named "constraint weighted" because of the weight w_2 and the constraint $f_2(x) \le l_2$. Variable l_2 iterates from the lower bound f_2^{LB} to the upper bound f_2^{UB} of $f_2(x)$.

In Algorithm 2, we show the general steps of finding all non-dominated solutions for any given k-objective BIP problem [37] (not limited to the model of our problem). The initial invocation of Algorithm 2 is calling $CWMOIP(k, f_k^{UB}, X)$, and then $CWMOIP(k-1, l_k, X)$ at line 11, recursively, until calling $CWMOIP(1, l_2, X)$. Initially, lines 2 to 4 calculate the true upper and lower bounds of the 2-nd objective to the k-th, subject to the constraint set X (In practice, this can be done once and results are cached for reuse). Then w_k is calculated for the k-th objective at line 5. In the loop at line 8, the k-objective problem is reduced to a new *(k-1)*-objective problem (line 11), which has the new suffix $w_k \cdot f_k(\vec{x})$ for the objective function (line 9) and the new constraint $f_k(\vec{x}) \le l_k$ for the constraint set X (line 10). If no results are found (lines 12-13), the recursion process stops. If found, the constraint l_k is tightened to the value just smaller than the largest value of $f_k(\vec{x})$ for $\vec{x} \in ME$. Last, BIP solving function is called when only one objective ($k = 1$) is left at line 6 to 7.

The maximum number of recursion is $\frac{|E|(|E|+1)\ldots(|E|+k-2)}{2\cdot 3\cdot \ldots \cdot (k-1)}$, according to [37]. Note that in our example, $\mathcal{F}_1(\vec{x})$ is not needed as it is always 0 for BIP. Thus, it is a 4-objective ($\mathcal{F}_2(\vec{x})$ to $\mathcal{F}_5(\vec{x})$)

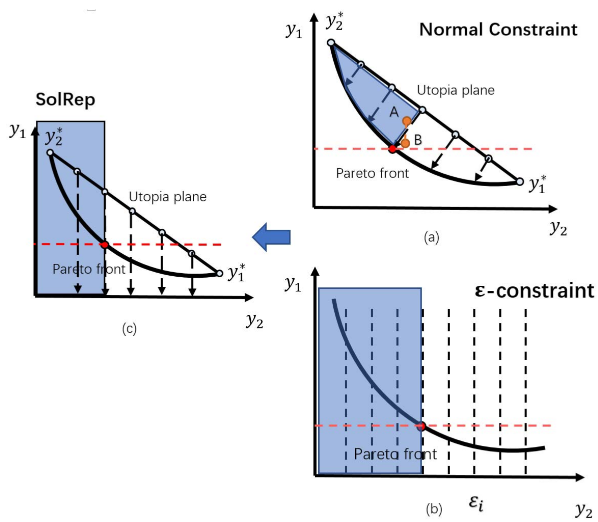

Figure 3: Two dimensional illustration of SolRep

BIP problem. *CWMOIP*() will reduce it to constrain-weighted 3-objective ($\mathcal{F}_2(\vec{x})$ to $\mathcal{F}_4(\vec{x})$); iteratively, until to a constrain-weighted 1-objective ($\mathcal{F}_2(\vec{x})$) problem.

5 OUR SOLUTION — SOLREP

The optimal feature selection problem is essentially a combinatorial problem. More often than not, the number of non-dominated solutions grows exponentially along the number of features and objectives. Given its NP-hardness, it is neither practical nor necessary to find the entire true Pareto front [25]. Instead, obtaining a set of solutions that evenly distributed over the true Pareto front is representative, pragmatic and computationally referrable.

The normal constraint (NC) method [32] is an effective method in the literature that guarantees generating evenly distributed solutions over the entire Pareto front. The main idea of NC method is to use the *utopia plane* to approximate the true Pareto front such that a set of evenly distributed reference points on *utopia plane* would result to a set of evenly distributed solutions on Pareto front, after the projection along the normal vector of *utopia plane*. Here, *utopia plane* refers to the hyper plane determined by all the anchor points, each of which individually optimizes a single objective (e.g. y_1^* for y_1-axis and y_2^* for y_2-axis in Fig 3.a). However, the original NC method was developed on continuous solution space. In the case of IP, it can generate dominated solutions (see Fig 3.a) where point A is a solution generated by NC, but there exists a point B outside the shaded feasible region, dominating A. So, we propose to combine the idea of *utopia plane* and the ϵ-constraint method such that all solutions are guaranteed to be non-dominated (see Fig 3.c).

Our proposed method consists of four main steps: 1) determine the utopia plane; 2) generate uniformly distributed reference points on the utopia plane; 3) for each reference point, use its $(k-1)$ coordinates to constrain the $(k-1)$ objectives; 4) optimize the k-th objective within the reduced solution space. Repeat steps 2) to 4) to achieve the representative Pareto front. In step 2), we resort to the hit-and-run (H&R) method [44] to sample the reference points. Its principle is straightforward: let p_t denote the current point then $p_{t+1} = p_t + \lambda * d$ is the next point, where d is a random direction vector and λ is the random length of the jump. As a random-walk algorithm, H&R is proven to generate uniformly distributed points inside any polyhedron after a sufficient number of runs [44].

Algorithm 3: Function *SolRep()* for k-objective IP

Input: k: the number of objectives, N: number of reference points, X: the set of linear constraints, f_i coefficient vector of the i-th objective function

Output: E: the set of representative non-dominated solutions

for $i = 1; i < k; i = i + 1$ **do**
  $y_{anch,i} = bintprog(X, f_i)$;
...//determine the vertexes of utopia plane;
$p_0 = randPoint(y_{anch})$;
...//generate a initial reference point p_0 on utopia plane;
$P \leftarrow \emptyset, P = P \cup \{p_0\}$;
for $i = 1; i < N; i = i + 1$ **do**
  $d = randDirect(y_{anch})$;
  $\lambda_{range} = linprog(X, d, p_0)$;
  $\lambda = unifrnd(\lambda_{range})$;
  $p_0 = p_0 + \lambda * d$;
  $P = P \cup \{p_0\}$;
...//find the other N-1 reference points;
$E \leftarrow \emptyset$;
for $i = 1; i \leq N; i = i + 1$ **do**
  $E = E \cup bintprog(X, f_k, P_i)$
return E;

Note that an obvious alternative is to equally divide the Pareto front along each objective direction and then traverse each grid. However, it might not evenly cut the Pareto front and the computational efforts grow exponentially along the number of objectives.

The proposed SolRep method is presented in Algorithm 3, which has the time complexity of $O(n)$ (consider *bintprog*(X, f_i) has $O(1)$, as a time limit is set for IP solving). The function *bintprog*(X, f_i) optimizes the i-th objective and returns one anchor point. Function $randPoint(y_{anch})$ generates a random initial reference point on utopia plane, $randDirect(y_{anch})$ returns a random direction vector, $linprog(X, d, p_0)$ returns the upper and lower bounds of the jumping length λ, $unifrnd(\lambda_{range})$ returns one random length within the bounds, $bintprog(X, f_k, P_i)$ optimizes the k-th objective with the constraints incurred by the reference point P_i.

6 EVALUATION

Implementation. To conduct a fair comparison between the MOIP methods and other MOEAs, we use the existing EAs from the open source tool of IBED [50]. We implement these MOIP methods in the IBED framework. These MOIP methods are mainly implemented in Java, only the IP solving method uses the APIs of Cplex [21]. Our tool and experimental data are published at [3].

Research Questions. For ϵ-constraint and CWMOIP that search for all non-dominated solutions, we answer the RQ1-RQ2. For SolRep that aims to efficiently find evenly-distributed non-dominated solutions, we compare it with the state-of-the-art MOEAs (i.e., IBED and IBEA) and answer the RQ3-RQ5:

RQ1. On small systems, can the ϵ-constraint and CWMOIP guarantee the *completeness* of non-dominated solutions?

RQ2. On small systems, what is the *performance* of the ϵ-constraint and CWMOIP?

RQ3. On medium-to-large systems, can SolRep find *many non-dominated* solutions?
RQ4. On medium-to-large systems, can SolRep find non-dominated solutions that are *evenly-distributed* ?
RQ5. On large industrial systems, is SolRep *scalable*?

6.1 Setup

6.1.1 Baseline Tools. Sayyad *et.al* [40, 42] reported that IBEA implemented in jMetal shows best results for the optimal feature selection. Recently, according to [50], on most systems (except *eCos*), IBED finds more non-dominated solutions than IBEA in similar time. IBEA and IBED represent the-state-of-art approaches and they are publicly available. So, we choose them as the baseline tools.

6.1.2 Baseline Tools Configurations. We adopt the version of IBEA and IBED that produce the best results according to [45][50] — actually the same enhancement for them, namely **IBED** $F + P$ and **IBEA** $F + P$. Here $F + P$ refers to the configuration that enables feedback-directed mechanism [45] and preprocessing for core feature encoding [19].

6.1.3 Parameter Settings. For the $F+P$ version of EAs, according to [45][50], the best setting of error mutation probability, mutation probability, and crossover probability are 1.0, 0.0000001, and 0.1 respectively. For SPLOT systems in Table 2, 25000 evaluations are used for IBEA and IBED. For larger LVAT systems, 100000 evaluations are used for both EAs. For any model, we generate 10 sets of attributes. For each set, we run each EA repeatedly for 30 times and use their union to compare with SolRep. Besides, the medium execution (the medium values of metrics) is compared with SolRep using 50 reference points. All other parameter settings for EAs are default settings of jMetal (e.g., population size is set to 100) are same as those used in [42][40][45][50].

In MOIP methods, one common parameter is needed — time limited for function *bintprog*. We set it as 0.1s for all system except for *Linux X86* (10s for this system due to its huge number of constraints). The setting for reference point number in SolRep is 1000 for SPLOT systems and 1500 for LVAT.

The experiments were performed on an Intel Core I7 4710HQ CPU with 8 GB RAM, running on Windows 8.1.

6.1.4 Quality Indicators. To measure the quality of Pareto front, we adopt four indicators in this work: percentage of correctness, hypervolume [54], spread [11] and Pareto dominance. We are using the same parameter setting (including objectives, cost values, etc.) and indicators as those in the earliest paper [42] on this topic. Currently, we are comparing HV, spread and time, and using the median value of 30 runs.

a) ***Percentage of Correctness* (%Cor)**: For MOEAs, not all solutions are correct, as correctness is an objective evolving over time. Thus, for a solution set, the percentage of correct solutions inside indicates its quality.
b) ***Hypervolume (HV)***: Hypervolume of the solution set S is the volume of the region that is dominated by S in the objective space. In jMetal, the Pareto front with a higher HV is preferred.
c) ***Spread (SP)***: Spread is used to measure the extend of spread in the obtained solutions.
d) ***Num. of Pareto non-dominance***: On the same set of feature attribute values, we can approximate the true Pareto front by the union of the two sets of solutions. On this Pareto front, we count the non-dominated solutions.

Table 2: Feature Models used in experiments

Repo.	Sys.	Fea.	Cons.	Ref.
SPLOT	JCS	12	12	[45]
	Web Portal	43	36	[30]
	E-Shop	290	186	[49]
LVAT	eCos	1244	3146	[43, 49]
	uClinux	1850	2468	[6]
	LinuxX86	6888	343944	[43]

6.2 Evaluation Systems

6.2.1 System Selection. Feature models of different systems are used in experiments. Table 2 shows repository name (*Repo.*), the model's system name (*Sys.*), number of features (*Fea.*), number of constraints (*Cons.*), and relevant literatures (*Ref.*).

Three models are from SPLOT, a repository used by many researchers as a benchmark [31]. *JCS* model is the running example of the paper. *Web Portal* model captures the configurations of Web portal product line, and *E-Shop* model captures a B2C system with fixed priced products. Industrial systems' models from the Linux Variability Analysis Tools (LVAT) repository [1] are used. LVAT's models (e.g., *Linux X86*) have much more features and constraints. All these models are chosen to facilitate the comparison with [42][45][50].

6.2.2 Feature Attribute. Each feature in each model has the following attributes, which are identical to attributes used in [42]:

1. ***Cost*** $\in \mathbb{R}$, records the price incurred to include the feature. For each feature, the *Cost* value is assigned with a real number that is normally distributed between 5.0 and 15.0.
2. ***Used_Before*** $\in \{true, false\}$, records whether this feature was used before — *true* for "yes" and *false* for "no". *Used_Before* values are uniformly distributed for features.
3. ***Defects*** $\in \mathbb{Z}$, records the number of defects in the feature. For each feature, the *Defects* value is assigned with an integer value normally distributed between 0 and 10.

6.3 Results

To accurately compare the quality of solutions found by two methods, we check Pareto dominance-relation and other indicators (HV, SP and time). In Table 3, 4, 5 and 6, we record the ***correct*** solutions found by the first method in column $|A|$, and those by the second one in column $|B|$. Column $|A \cap B|$ shows the number of ***correct solutions commonly found*** by both methods. After non-dominance sorting, column $|N_A \cup N_B|$ lists the number of ***the union of correct non-dominated*** solutions found by both methods. We also list the number of correct ***unique non-dominated*** solutions found only by the first (or the second) method in column $|N_A^U|$ (or $|N_B^U|$). To mitigate the bias due to one-time execution of EAs, we compare MOIP methods with them based on the union of non-dominated solutions found by 30 executions of EAs. $|Time(s)|$ refers to the total time —- for EAs, it is total time for 30 executions. To get $|HV|$ and $|SP|$, we approximate true Pareto front with $|N_A \cup N_B|$ and use that as reference points.

We first evaluate the ϵ-constraint and CWMOIP that aim to find all non-dominated solutions.

Table 3: Non-dominated solutions found by ϵ-constraint (A) and CWMOIP (B) on two small models, with 4 attribute sets for each model

System	A %Corr	B %Corr	A Time(s)	B Time(s)	A HV	B HV	A SP	B SP	$\|A\|$	$\|B\|$	$\|A \cap B\|$	$\|N_A \cup N_B\|$	$\|N_A^U\|$	$\|N_B^U\|$
JCS 1	100	100	5.11	1.66	0.29	0.29	0.69	0.69	39	39	39	39	0	0
JCS 2	100	100	6.84	0.67	0.27	0.27	0.39	0.39	9	9	9	9	0	0
JCS 3	100	100	3.54	0.96	0.29	0.29	0.43	0.43	22	22	22	22	0	0
JCS 4	100	100	7.92	1.97	0.25	0.25	0.56	0.56	31	31	31	31	0	0
Web Portal 1	100	100	570.68	212.22	0.34	0.34	0.87	0.87	451	451	451	451	0	0
Web Portal 2	100	100	692.88	256.38	0.31	0.31	0.98	0.98	768	768	768	768	0	0
Web Portal 3	100	100	696.83	237.75	0.32	0.32	0.92	0.92	859	859	859	859	0	0
Web Portal 4	100	100	564.82	163.97	0.33	0.33	0.96	0.96	713	713	713	713	0	0

Table 4: Non-dominated solutions found by CWMOIP (A) and IBED $F + P$ (B) on two small models, with 4 attribute sets for each model

System	A %Corr	B %Corr	A Time(s)	B Time(s)	A HV	B HV	A SP	B SP	$\|A\|$	$\|B\|$	$\|A \cap B\|$	$\|N_A \cup N_B\|$	$\|N_A^U\|$	$\|N_B^U\|$
JCS 1	100	92.37	1.66	150.66	0.29	0.28	0.69	0.28	39	19	19	39	20	0
JCS 2	100	91.72	0.67	131.19	0.27	0.27	0.39	0.39	9	9	9	9	0	0
JCS 3	100	85.80	0.96	135.38	0.29	0.28	0.43	0.31	22	17	17	22	5	0
JCS 4	100	84.77	1.97	135.20	0.25	0.24	0.56	0.54	31	20	20	31	11	0
Web Portal 1	100	98.00	212.22	162.80	0.34	0.33	0.87	0.45	451	232	97	451	354	0
Web Portal 2	100	98.73	256.38	166.67	0.31	0.30	0.98	0.63	768	277	116	768	652	0
Web Portal 3	100	99.63	237.75	156.52	0.32	0.31	0.92	0.66	859	386	184	859	675	0
Web Portal 4	100	96.93	163.97	163.69	0.33	0.32	0.96	0.73	713	327	167	713	546	0

6.3.1 RQ1—Solution Set Completeness on Small Systems. As shown in Table 3, on 4 different sets of feature attributes of each model, we can always find $|A|$=$|B|$= $|N_A \cup N_B|$, which means they found exactly same solutions. As the completeness of the ϵ-constraint and CWMOIP has been proven in [17][37], we confirm this by the experiments. Comparing CWMOIP with IBED in Table 4, we see **IBED cannot find all non-dominated solutions with 30 executions in most cases** (except *JCS2* that has only 9 non-dominated solutions). In most cases, IBED's results are unsatisfactory, covering averagely about 70% of the true Pareto front on 4 attribute sets of *JCS*, and about 20% of the true Pareto front on *Web Portal*. Also, we find $|N_A \cup N_B| = |A|$, which proves the completeness of $|A|$.

6.3.2 RQ2—Performance on Small Systems. In Table 4, we can see that it takes averagely about 5s for ϵ-constraint method and 1.5s for CWMOIP on 4 attribute sets of *JCS*, and averagely about 613s for ϵ-constraint method and 218s for CWMOIP on 4 attribute sets of *Web Portal*. In contrast, one medium execution of IBED (i.e., the one with the medium size of $|A|$ in 30 executions) on *JCS 1* takes about 4.5s to find 17 correct solutions. Actually, on *JCS 1*, 30 executions just brings several more solutions than 1 executions, as the search space is small and EAs can converge fast. Similarly, one medium execution of IBED on *Web Portal 1* takes about 5.4s to find 35 correct solutions, while 30 executions just produce 232 correct solutions, as many executions also share solutions.

In theory, the ϵ-constraint and CWMOIP both have exponential time complexity (§4). In Table 3 and 4, experimental results confirm that — **from 12 features (*JCS*) to 43 features (*Web Portal*), the time of ϵ-constraint and CWMOIP increases by at least 100-fold, respectively**. We run CWMOIP on any attribute set of *E-Shop* (290 features), and in 6 hours it cannot finish.

To make MOIP methods scalable and effective, we propose SolRep and evaluate this method in the following subsections.

6.3.3 RQ3—Richness and Non-dominance Solutions Found by SolRep. In Table 5 and 6, the richness of solutions refers to the number of unique solutions found only by one method, namely $|N_A^U|$ or $|N_B^U|$. The larger $|N_A^U|$ is than $|N_B^U|$, solution set A have better richness than B. In Table 5, we observe **in most cases, SolRep finds significantly more non-dominated solutions than IBED, except on *JCS 2* and *E-Shop 2***. On *JCS 2*, IBED finds all 9 non-dominated solutions because of the small search space, but SolRep fails to find all based on reference points method. On *E-Shop 2*, SolRep finds less solutions than IBED, as only 742 feasible solutions are found among 1000 reference points of the solution space — the true Pareto Front is not continuous, and on 258 points no feasible solutions are found by IP. **If relaxing the number of reference points to 1500, still on *E-Shop 2*, SolRep takes 123s to find 1099 non-dominated solutions**, more than IBED's 825 solutions in 201s. In Table 6, only results on system *eCos* are reported, where IBEA showed better results than IBED [50]. As Pareto-dominance relation is transitive, and cases where IBEA finds less than IBED are not necessary to be shown. SolRep also finds more non-dominated solutions than IBEA on *eCos*. Hence, richness of solutions found by SolRep is better than IBED and IBEA.

Regarding the non-dominance of solutions found by SolRep, we observe that $|A| = |A \cap B| + |N_A^U|$ **for all cases in Table 5 and 6**. The rationale is that if solutions in A are all non-dominated, after dominance-sorting for A and B, its solutions in A will be inside the intersection $|A \cap B|$ or be the unique solutions in A ($|N_A^U|$), but not dominated by B. After scrutiny, the non-dominance is guaranteed by SolRep. We find $|B| \gg |A \cap B| + |N_B^U|$ for IBED or IBEA. Especially on *Linux X86*, most solutions of IBED are dominated by SolRep's.

6.3.4 RQ4—Even-distribution of Solutions Found by SolRep. HV and SP can to some extent imply how evenly-distributed the solutions are. If the solution set A is with low HV and low SP, but high $|N_A^U|$ — meaning solutions in A are gathered in a small part of the true Pareto front. In Table 5 and 6, we fail to find such case. **In most cases, $|N_A^U|$, HV and SP of A are larger than B's, respectively.** Only in *JCS 2* and *Web Portal 1, 2*, SolRep's HV is just slightly worse than IBED's HV; still in the three cases, SolRep's SP is significantly better than IBED's SP. Hence, on small systems, SolRep is

Table 5: Non-dominated solutions found by SolRep (A) and IBED F+P (B) on SPLOT, LVAT models. On *LinuxX86*, IBED uses 3 seed solutions.

System	A %Corr	B %Corr	A Time(s)	B Time(s)	A HV	B HV	A SP	B SP	$\|A\|$	$\|B\|$	$\|A \cap B\|$	$\|N_A \cup N_B\|$	$\|N_A^U\|$	$\|N_B^U\|$
JCS 1	100	92.37	2.59	150.66	0.28	0.28	0.65	0.27	29	19	14	34	15	5
JCS 2	100	91.72	2.28	131.19	0.26	0.27	0.76	0.39	4	9	4	9	0	5
WebPortal 1	100	98.00	6.28	162.80	0.30	0.32	0.54	0.48	120	232	39	188	81	68
WebPortal 2	100	98.73	10.75	166.67	0.28	0.30	0.79	0.70	221	277	27	330	194	109
E-Shop 1	100	99.06	46.98	203.24	0.28	0.27	0.72	0.60	773	1491	0	1454	773	681
E-Shop 2	100	98.69	72.00	201.33	0.26	0.31	0.61	0.64	742	1441	0	1567	742	825
eCos 1	100	87.64	309.27	1444.74	0.29	0.25	0.68	0.55	1460	1306	0	2226	1460	766
eCos 2	100	88.78	308.95	1436.39	0.29	0.24	0.79	0.56	1461	1337	0	2214	1461	753
uClinux 1	100	100	83.49	1348.31	0.31	0.25	0.87	0.62	1111	1929	0	1988	1111	877
uClinux 2	100	100	130.53	1367.45	0.32	0.24	0.77	0.64	950	1875	0	1424	950	474
LinuxX86 1	100	28.64	47,414	74,863	0.33	0.12	0.70	1.04	1477	464	0	1505	1477	28
LinuxX86 2	100	20.27	63,750	78,339	0.33	0.06	0.77	1.01	1472	344	0	1483	1472	11

Table 6: Non-dominated solutions found by SolRep (A) and IBEA $F + P$ (A) on system $eCos$, where IBEA performs better than IBED. For other systems, IBEA finds significantly less non-dominated solutions than IBED [50]. Hence, results for other systems are omitted here.

System	A %Corr	B %Corr	A Time(s)	B Time(s)	A HV	B HV	A SP	B SP	$\|A\|$	$\|B\|$	$\|A \cap B\|$	$\|N_A \cup N_B\|$	$\|N_A^U\|$	$\|N_B^U\|$
eCos 1	100	92.27	309.27	1017.41	0.29	0.24	0.68	0.59	1460	1533	0	2269	1460	809
eCos 2	100	92.60	308.95	983.63	0.29	0.24	0.79	0.58	1461	1621	0	2363	1461	902

no worse than IBED in terms of even-distribution. From *E-Shop* onwards, the two methods share no commonly-found solutions. **On *E-Shop*, *eCos* and *uClinux*, SolRep's HV, SP and #unique non-dominated solutions are all clearly better than IBED's.**

On *Linux X86*, we find that: SolRep has much better HV (0.33) than IBED's HV (0.09 on average for two attribute sets), most of IBED solutions are dominated by SolRep ($|B| \gg |N_B^U|$). However, IBED's SP (above 1.0) is even higher than SolRep's (0.7-0.8) — implying the distance between IBED's solutions is wider than that of SolRep's solutions. Considering the quite few number of IBED's solutions ($|N_B^U|$ is only 28 and 11), it is reasonable. If IBED could find more non-dominated solutions near the current ones, SP would decrease. High SP (wide-distance) of IBED solutions is attributed to DE operators used in the dual-population evolution [50].

6.3.5 RQ5—Scalability of SolRep. As shown in Table 5, SolRep is fast on most systems, except *Linux X86*. We find both feature number and constraint number (more important) affect the execution time of SolRep. On three systems from SPLOT, at most 72s (on *E-Shop 2*) is needed with 1000 reference points (RPs). On LVAT models using 1500 RPs, the execution time is within 131s for *uClinux* and within 310s for *eCos*. More time is needed for *eCos* than *uClinux*, as *eCos* has 3146 constraints and *uClinux* has 2468. **In Table 5 and 6, SolRep takes less time than executing IBED 30 times, but find more non-dominated solutions in most cases.**

We study how the number of RPs affects the execution time. For *Linux X86 1*, calculating utopia plate takes 550s and solving the 1500 RFs takes 49864s. In Fig. 4, it shows the time complexity and the solution number increase linearly with the number of RPs. The time for calculating a non-dominated solution (according to a RP) is fixed on a certain system. The reason is the time complexity of Algorithm 3 is $O(n)$, if time limit for IP solving is set.

As 30 executions of EAs may share common solutions, we compare SolRep using 50 RPs with the medium execution of IBED, in terms of time and solution number. On *Linux X86 1*, SolRep using 50 RPs finds 50 non-dominated solutions in 2150s. It is better than the medium execution of IBED with 3 seeds (among 30 executions), which takes 2534s to find 24 correct solutions (only 4 are non-dominated). On *uClinux 1*, SolRep using 50 RPs finds 49 solutions in 6s, while IBED (the medium execution) finds 67 non-dominated solutions in 43s. On *eCos 1*, SolRep using 50 RPs find 48 solutions in 11s, and IBED (the medium execution) finds 42 non-dominated solutions in 48s. Hence, **on LVAT systems, SolRep using 50 RPs finds more non-dominated solutions than the medium execution of IBED, using similar or less time**.

Summary. On small systems, CWMOIP can find all non-dominated solutions — assuring the solution completeness. On medium-to-large systems, SolRep is scalable — no matter compared with EA's 30-executions union or only the medium execution, in most cases SolRep finds more non-dominated solutions than IBED and IBEA, with similar or less time. Hence, MOIP methods are effective and provide a new perspective on solving linear-constrained linear-objective MOO problems.

7 DISCUSSION

Threats to Validity. One threat to validity is about randomly generated values for feature attributes (i.e., Cost, Defects, and Used Before). To mitigate the effect of randomness, we generate 10 sets

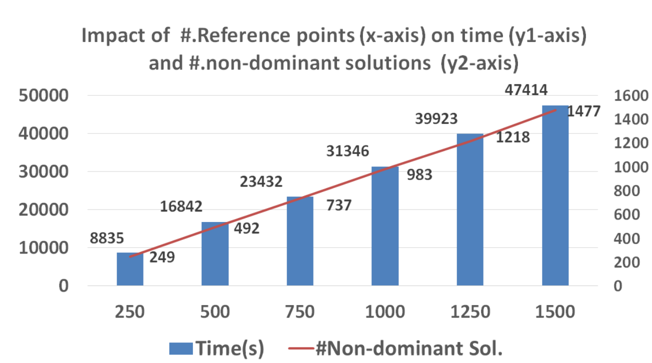

Figure 4: Impact of #. reference points (x-axis) on execution time (y1-axis) and #.non-dominated solutions (y2-axis) on *Linux X86 1*

of attributes for each model. Due to the page limit, we just report 2 or 4 representative attribute sets in the evaluation. According to [50] and our observation, impact of attribute values is minor to the results — if on two sets of attributes a method is better; then in general (like on ten sets) this method is better. The second threat is about the systems chosen. In future, the *EC2* feature model [13] and the *Drupal* model [39] need to be evaluated. The last threat is about the parameters of the EAs used as baseline tools. The best parameter setting of IBED (and also IBEA) reported by [50] is used.

MOIP or IBEA with Constraint Solving, or Others? In this study, we discuss the general theme — whether methods other than MOEAs could be applied to this SBSE problem. As illustrated in §6, we show that SolRep, the IP method, has been proven to be capable for this optimal feature selection problem. On large systems, SolRep takes similar time and finds significantly more non-dominated solutions than IBEA or IBED alone.

The next step is to conduct the detailed comparison on SolRep and the IBEA with constraint solving (i.e., SATIBEA [18] and SMTIBEA [14]). However, as IP-methods and IBEA with constraint solving are designed for different purposes, we need to be careful in selecting the evaluation metrics — IP-methods are designed for finding more non-dominated solutions that are as evenly-distributed as possible, while IBEA with constraint solving is designed to use SAT- or SMT- based repairing operator to find more valid solutions. Valid solutions are not necessary (in most case, they are not) to be global optimal (i.e., non-dominated by any other solutions in solution space). Now, we are working on finding proper metrics and setting up the benchmark to compare them fairly.

Actually, apart from MOEA-based and IP-based approaches, other attempts have been made to resolve the MOO problems in the SBSE. In recent two years, there are three pioneer studies on this directions. Zuluaga *et al.* [55] proposed the ϵ-Pareto Active Learning (ϵ-PAL). Krall *et al.* [26] propose GALE which is a near-linear time MOEA. Chen *et al.* [34] propose SWAY (Samplying WAY) to finds (near) optimal solutions by sampling. None of the above three systems have received much attention from SE community. To our best knowledge, none of these approaches have been applied to this optimal feature selection problem. Hence, the message of using approaches other than MOEAs is still unheard in SE. It will be interesting to conduct a throughout empirical study on investigating each approach's merits and drawbacks in solving the SBSE problem.

The Theoretic Limitation of SolRep. As explained in §1, MOEA-based approaches, as a heuristic and non-deterministic algorithm, face the four drawbacks on algorithm convergence, solution set completeness, the global optimality of solutions, and the solution set diversity (or evenly-distributiveness). In contrast, SolRep, as a analytic and mostly-deterministic algorithm, in theory, does not have the problem of algorithm convergence, and the global optimality of solutions. For solution set completeness, SolRep is not designed for solution set completeness (ϵ-constraint and CWMOIP are designed for this goal on small systems). Currently, the solution set diversity of SolRep is guaranteed by random sampling and random-walk mechanisms. The time complexity of SolRep is $O(n)$ (see Algorithm 3, Fig. 4 and §6.3.5), as long as the time for one time IP solving in Cplex has a constant upper-limit. Hence, SolRep is a mostly-deterministic algorithm, as it resorts to random sampling. The theoretic limit of SolRep is that it may fail if most non-dominated solutions are gathered in a small zone and H&R method cannot hit that zone. However, we did not find this case in our evaluated systems.

Generality of SolRep. *The important assumption of applying SolRep is that all the constraints and objectives need to be linear.* This assumption might narrow the application of SolRep in SBSE domain, since many real-world MOO problems in SBSE have non-linear objectives, e.g., the optimal refactoring step problem [33]. Still, one direct application of SolRep is feasible. Previously, Zhang *et al.* [51] propose a novel approach to time-aware test-case prioritization using IP. As in [51], the total time for selected test cases is the constraint, and the objective is to maximize the number of covered statements of selected test cases. Essentially, *the optimal test case priority problem can be reduced to a problem similar to the optimal feature selection* — one linear objective to minimize the number of selected test cases, one linear objective to minimize the total time for selected test cases, and one linear objective to maximize the number of covered statements of selected test cases.

8 RELATED WORK

The Optimal Feature Selection Problem. White *et al.* [48] first modeled the feature selection problem as a Multidimensional Multi-Choice Knapsack Problem (MMKP), and applied Filtered Cartesian Flattening (FCF) to derive an optimal feature configuration subject to resource constraints. Guo *et al.* [15] first proposed a genetic algorithm (GA) approach to tackle this problem. In [15], a repair operator is used to fix each candidate solution, and make it comply with the feature model during evolution. One limitation of [15] is aggregating all objectives into a single objective by weight.

To address the objective aggregation issue, Sayyad *et al.* [40, 42] first proposed to apply various MOEAs, and a range of optimal solutions (a.k.a., a Pareto front) is returned to the user as a result. As reported, IBEA [52] yields the best results among the seven tested EAs in terms of time, correctness and satisfaction to user preferences. In [41], they further used static method to prune features before execution of IBEA for reducing search space. They also introduced a "seeding method" by pre-computing a correct solution, which was later implanted the initial population of IBEA. Along this line, Tan *et al.* [45] improved these previous studies by using a novel feedback-directed mechanism to existing EAs. Similarly, to improve the correctness, Hierons *et al.* [19] proposed the $1 + n$ approach that prioritizes the number of failed constraints and considers the correctness objective first.

Recently, IBEA (including its variants) is integrated with other techniques for achieving better results. Henard *et al.* [18] integrated IBEA with constraint solving. They permuted different SAT parameters to maximize the diversity of SAT solutions in a cheap way by calling SAT solver hundreds of times. Similarly, along this line, Guo *et al.* [14] proposed to combine (IBEA) with the satisfiability modulo theories (SMT) solving. Xue *et al.* [50] integrated IBEA with differential evolution (DE) for achieving both correctness and diversity of solutions.

In comparison, many non-dominated solutions found by SolRep can be useful in practice, as one non-dominated solution can dominate many solutions found by other approaches. Furthermore, SolRep allows exploration of the entire solution space, which is difficult to realize by MOEAs.

Finding Exact Solution for This Problem. There are existing studies on finding exact solutions for this problem. Similar to our study, Guo *et al.* [16] applied several GAs with parallelization on the three small-to-medium systems, SAS, *Web Portal* [30] and *E-Shop*. They reported that Feature Split GIA (FS-GIA) achieves substantial (even super-linear) speedups that scale well up to 64 cores. Last, what we have summarized in this paper is consistent with the finding of the study [36]. Olaechea *et al.* proved that (1) it is feasible to use exact techniques (GIA) for small SPL systems, (2) approximate methods (IBEA) can be used for large problems which may takes much effort in finding the best setup of parameters for good approximation. In this paper, we proved that using exact IP-methods ϵ-constraint and CWMOIP can solve small SPL systems, and the proposed IP-method (SolRep, an approach adopting random sampling) can be applied for large systems. Note that only two parameters are needed for SolRep, the number of reference points and the constant upper-limit time for one time IP solving in Cplex.

OR and SBSE. IP has been used for the instantiation of products — the valid product feature selection problem [47]. In [47], only one single objective is considered, not addressing MOO via IP. Earlier than this study, requirement interdependencies were resolved via IP[8]. Then, flexible release planning was solved by IP [46]. Subsequently, requirements selection and scheduling for the release planning were integrated and optimized by IP to cater for budgetary constraints [28]. Further, IP was combined with computational intelligence and human negotiation to address conflicting objectives [38]. Recently, IP is not widely used due to its scalability issues and strict limitation on the application. Meanwhile, due to the emergence of MOO problems, MOEAs become the default method.

9 CONCLUSION

In this paper, we formulate the optimal feature selection problem in SBSE as a MOIP model. We try the ϵ-constraint and CWMOIP, and prove the completeness of their solutions on small systems. However, they are not scalable. To address this, we propose an innovative MOIP method — SolRep, which is scalable and finds more non-dominated solutions than IBED in most cases, using similar or less time. In future, we will compare SolRep with SATIBEA [18] and SMTIBEA [14] on effectiveness and efficiency. Last, to overcome the limitation of applicability to only linear objective functions, the proposed framework can be generalized into non-linear cases by linking it with the single-objective non-linear programming techniques, e.g. convex quadratic integer programming, to solve the problems of the respective nature. In future, we will extend this framework into quadratic objective functions.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their constructive comments. Professor Li's work was partially supported by the National Natural Science Foundation of China under a key project grant No. 71731008.

REFERENCES

[1] [n. d.]. Linux Variability Analysis Tools (LVAT) Repository. ([n. d.]). https://code.google.com/p/linux-variability-analysis-tools/source/browse/?repo=formulas.

[2] Tarek K. Abdel-Hamid, Kishore Sengupta, and Clint Swett. 1999. The Impact of Goals On Software Project Management: An Experimental Investigation. *MIS Quarterly* 23, 4 (1999), 531–555.

[3] Anonymous. [n. d.]. ([n. d.]). https://sites.google.com/view/ip-method-repsol.

[4] Don S. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Software Product Lines, 9th International Conference, SPLC 2005, Rennes, France, September 26-29, 2005, Proceedings*. 7–20.

[5] David Benavides, Pablo Trinidad Martín-Arroyo, and Antonio Ruiz Cortés. 2005. Automated Reasoning on Feature Models. In *CAiSE*. 491–503.

[6] Thorsten Berger, Steven She, Rafael Lotufo, Krzysztof Czarnecki, Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wasowski, and Krzysztof Czarnecki. 2012. *Variability Modeling in the Systems Software Domain*. Technical Report. University of Waterloo.

[7] Charles E. Blair, Robert G. Jeroslow, and James K. Lowe. 1986. Some results and experiments in programming techniques for propositional logic. *Computers & OR* 13, 5 (1986), 633–645. https://doi.org/10.1016/0305-0548(86)90056-0

[8] Pär Carlshamre. 2002. Release Planning in Market-Driven Software Product Development: Provoking an Understanding. *Requir. Eng.* 7, 3 (2002), 139–151. https://doi.org/10.1007/s007660200010

[9] Paul Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns* (3rd ed.). Addison-Wesley Professional. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0201703327

[10] Krzysztof Czarnecki and Ulrich W. Eisenecker. 2000. *Generative programming - methods, tools and applications*. Addison-Wesley. I–XXVI, 1–832 pages.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation* 6, 2 (2002), 182–197.

[12] Juan José Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. 2009. On the Effect of the Steady-State Selection Scheme in Multi-Objective Genetic Algorithms. In *EMO*. 183–197.

[13] Jesús García-Galán, Pablo Trinidad, Omer F. Rana, and Antonio Ruiz Cortés. 2016. Automated configuration support for infrastructure migration to the cloud. *Future Generation Comp. Syst.* 55 (2016), 200–212.

[14] Jianmei Guo, Jia Hui Liang, Kai Shi, Dingyu Yang, Jingsong Zhang, Krzysztof Czarnecki, Vijay Ganesh, and Huiqun Yu. 2017. SMTIBEA: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines. *Software & Systems Modeling* (22 Jul 2017).

[15] Jianmei Guo, Jules White, Guangxin Wang, Jian Li, and Yinglin Wang. 2011. A genetic algorithm for optimized feature selection with resource constraints in software product lines. *Journal of Systems and Software* 84, 12 (2011), 2208–2221.

[16] Jianmei Guo, Edward Zulkoski, Rafael Olaechea, Derek Rayside, Krzysztof Czarnecki, Sven Apel, and Joanne M. Atlee. 2014. Scaling exact multi-objective combinatorial optimization by parallelization. In *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*. 409–420. https://doi.org/10.1145/2642937.2642971

[17] Yacov Y. Haimes, Leon S. Lasdon, and David A. Wismer. 1971. On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1, 3 (July 1971), 296–297. https://doi.org/10.1109/TSMC.1971.4308298

[18] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 2015. Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*. 517–528. https://doi.org/10.1109/ICSE.2015.69

[19] Robert M. Hierons, Miqing Li, Xiaohui Liu, Sergio Segura, and Wei Zheng. 2016. SIP: Optimal Product Selection from Feature Models Using Many-Objective Evolutionary Optimization. *ACM Trans. Softw. Eng. Methodol.* 25, 2 (2016), 17:1–17:39. https://doi.org/10.1145/2897760

[20] J. N. Hooker. 1988. Generalized Resolution and Cutting Planes. *Ann. Oper. Res.* 12, 1-4 (Feb. 1988), 217–239. https://doi.org/10.1007/BF02186368

[21] IBM. [n. d.]. ([n. d.]). https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[22] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. 2008. Evolutionary Many-Objective Optimization: A Short Review. In *CEC*. 2419–2426.

[23] Ivar Jacobson, Martin L. Griss, and Patrik Jonsson. 1997. *Software reuse - architecture, process and organization for business*. Addison-Wesley-Longman. I–XXVIII, 1–497 pages.

[24] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-21. Carnegie Mellon University.

[25] Murat Köksalan. 2009. Multiobjective combinatorial optimization: some approaches. *Journal of Multi-Criteria Decision Analysis* 15, 3-4 (2009), 69–78.

[26] Joseph Krall, Tim Menzies, and Misty Davies. 2015. GALE: Geometric Active Learning for Search-Based Software Engineering. *IEEE Trans. Software Eng.* 41, 10 (2015), 1001–1018. https://doi.org/10.1109/TSE.2015.2432024

[27] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. 2002. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation* 10, 3 (2002), 263–282. https://doi.org/10.1162/106365602760234108

[28] C. Li, J. M. van den Akker, Sjaak Brinkkemper, and Guido Diepen. 2007. Integrated Requirement Selection and Scheduling for the Release Planning of a Software Product. In *Requirements Engineering: Foundation for Software Quality, 13th International Working Conference, REFSQ 2007, Trondheim, Norway, June 11-12, 2007, Proceedings.* 93–108.

[29] Mike Mannion and Javier Cámara. 2003. Theorem Proving for Product Line Model Verification. In *PFE.* 211–224.

[30] Marcílio Mendonça, Thiago T. Bartolomei, and Donald D. Cowan. 2008. Decision-making coordination in collaborative product configuration. In *SAC.* 108–113.

[31] Marcílio Mendonça, Moises Branco, and Donald D. Cowan. 2009. S.P.L.O.T.: software product lines online tools. In *OOPSLA Companion.* 761–762.

[32] Achille Messac and Christopher A. Mattson. 2004. Normal Constraint Method with Guarantee of Even Representation of Complete Pareto Frontier. *AIAA Journal* 42 (Oct. 2004), 2101–2111. https://doi.org/10.2514/1.8977

[33] Mohamed Wiem Mkaouer, Marouane Kessentini, Mel Ó Cinnéide, Shinpei Hayashi, and Kalyanmoy Deb. 2017. A robust multi-objective approach to balance severity and importance of refactoring opportunities. *Empirical Software Engineering* 22, 2 (2017), 894–927. https://doi.org/10.1007/s10664-016-9426-8

[34] Vivek Nair, Tim Menzies, and Jianfeng Chen. 2016. An (Accidental) Exploration of Alternatives to Evolutionary Algorithms for SBSE. In *Search Based Software Engineering - 8th International Symposium, SSBSE 2016, Raleigh, NC, USA, October 8-10, 2016, Proceedings.* 96–111. https://doi.org/10.1007/978-3-319-47106-8_7

[35] Antonio José Nebro, Juan José Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. 2009. MOCell: A cellular genetic algorithm for multiobjective optimization. *Int. J. Intell. Syst.* 24, 7 (2009), 726–746.

[36] Rafael Olaechea, Derek Rayside, Jianmei Guo, and Krzysztof Czarnecki. 2014. Comparison of exact and approximate multi-objective optimization for software product lines. In *18th International Software Product Line Conference, SPLC '14, Florence, Italy, September 15-19, 2014.* 92–101. https://doi.org/10.1145/2648511.2648521

[37] Melih Özlen and Meral Azizoglu. 2009. Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research* 199, 1 (2009), 25–35. https://doi.org/10.1016/j.ejor.2008.10.023

[38] Günther Ruhe and Moshood Omolade Saliu. 2005. The Art and Science of Software Release Planning. *IEEE Software* 22, 6 (2005), 47–53. https://doi.org/10.1109/MS.2005.164

[39] Ana B. Sánchez, Sergio Segura, José A. Parejo, and Antonio Ruiz-Cortés. 2015. Variability testing in the wild: the Drupal case study. *Software & Systems Modeling* (2015), 1–22. https://doi.org/10.1007/s10270-015-0459-z

[40] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. 2013. Optimum Feature Selection in Software Product Lines: Let Your Model and ValuesGuide Your Search. In *CMSBSE.* 22–27.

[41] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. 2013. Scalable product line configuration: A straw to break the camel's back.. In *ASE.*

[42] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. 2013. On the value of user preferences in search-based software engineering: a case study in software product lines. In *ICSE.* 492–501.

[43] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2011. Reverse engineering feature models. In *ICSE.* 461–470.

[44] Robert L. Smith. 1984. Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions. *Operations Research* 32, 6 (1984), 1296–1308. https://doi.org/10.1287/opre.32.6.1296

[45] Tian Huat Tan, Yinxing Xue, Manman Chen, Jun Sun, Yang Liu, and Jin Song Dong. 2015. Optimizing selection of competing features via feedback-directed evolutionary algorithms. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis, ISSTA 2015, Baltimore, MD, USA, July 12-17, 2015.* 246–256. https://doi.org/10.1145/2771783.2771808

[46] Marjan van den Akker, Sjaak Brinkkemper, Guido Diepen, and Johan Versendaal. 2005. Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming. In *The 17th Conference on Advanced Information Systems Engineering (CAiSE '05), Porto, Portugal, 13-17 June, 2005, CAiSE Forum, Short Paper Proceedings.* http://ceur-ws.org/Vol-161/FORUM_20.pdf

[47] Pim van den Broek. 2010. Optimization of Product Instantiation using Integer Programming. In *Software Product Lines - 14th International Conference, SPLC 2010, Jeju Island, South Korea, September 13-17, 2010. Workshop Proceedings (Volume 2 : Workshops, Industrial Track, Doctoral Symposium, Demonstrations and Tools).* 107–112. http://splc2010.postech.ac.kr/SPLC2010_second_volume.pdf

[48] Jules White, Brian Dougherty, and Douglas C. Schmidt. 2009. Selecting highly optimal architectural feature sets with Filtered Cartesian Flattening. *Journal of Systems and Software* 82, 8 (2009), 1268–1284.

[49] Yinxing Xue, Zhenchang Xing, and Stan Jarzabek. 2010. Understanding Feature Evolution in a Family of Product Variants. In *WCRE.* 109–118.

[50] Yinxing Xue, Jinghui Zhong, Tian Huat Tan, Yang Liu, Wentong Cai, Manman Chen, and Jun Sun. 2016. IBED: Combining IBEA and DE for optimal feature selection in software product line engineering. *Appl. Soft Comput.* 49 (2016), 1215–1231. https://doi.org/10.1016/j.asoc.2016.07.040

[51] Lu Zhang, Shan-Shan Hou, Chao Guo, Tao Xie, and Hong Mei. 2009. Time-aware test-case prioritization using integer linear programming. In *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis, ISSTA 2009, Chicago, IL, USA, July 19-23, 2009.* 213–224. https://doi.org/10.1145/1572272.1572297

[52] Eckart Zitzler and Simon Künzli. 2004. Indicator-Based Selection in Multiobjective Search. In *PPSN.* 832–842.

[53] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* Technical Report. Swiss Federal Institute of Technology (ETH) Zurich.

[54] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation* 3, 4 (1999), 257–271.

[55] Marcela Zuluaga, Andreas Krause, and Markus Püschel. 2016. e-PAL: An Active Learning Approach to the Multi-Objective Optimization Problem. *Journal of Machine Learning Research* 17 (2016), 104:1–104:32. http://jmlr.org/papers/v17/15-047.html