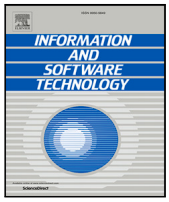
Information and Software Technology 147 (2022) 106825

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Multi-objective integer programming approaches to Next Release Problem — Enhancing exact methods for finding whole pareto front

Shi Dong [a], Yinxing Xue [a,*], Sjaak Brinkkemper [b], Yan-Fu Li [c]

[a] School of Computer Science and Technology, University of Science and Technology of China, China
[b] Department of Information and Computing Sciences of Utrecht University, Netherlands
[c] Department of Industrial Engineering, Tsinghua University, China

ARTICLE INFO

Keywords:
Next Release Problem
Multi-objective optimization
Integer linear programming
Search-based software engineering

ABSTRACT

Context: Project planning is a crucial part of software engineering, it involves selecting requirements to develop for the next release. How to make a good release plan is an optimization problem to maximize the goal of revenue under the condition of cost, time, or other aspects, namely Next Release Problem (NRP). Genetic and exact algorithms are used since it was proposed.

Objective: We model NRP as bi-objective (revenue, cost) and tri-objective (revenue, cost, urgency) form, and investigate whether exact methods could solve bi-objective and tri-objective instances more efficiently.

Methods: The state-of-art integer linear programming (ILP) approach to the bi-objective NRP is ϵ-constraint for finding all non-dominate solutions. To improve its efficiency, we employ CWMOIP (Constrained Weighted Multi-Objective Integer Programming) and I-EC (improved ϵ-constraint) for solving bi-objective instances. In tri-objective form, we introduce SolRep, an ILP method that optimizes the reference points from sampling, for finding solutions subset within a short time. NSGA-II is implemented as the evolutionary algorithm for the comparison with former methods and it adopts the seeding mechanism.

Results : I-EC can find all non-dominated solutions with better performance than both ϵ-constraint and CWMOIP on all instances except for one. I-EC reduces solving time by 19.7% (large instances) and 91.5% (small instances) on average separately compared with ϵ-constraint. SolRep can find evenly distributed solutions and exceed NSGA-II illustrated by several indicators (such as HyperVolume) on tri-objective instances. And each method has its merit in the aspect of speed and number of the solutions.

Conclusion: (1) The I-EC can solve all non-dominated solutions with better performance than the state-of-art exact method. (2) SolRep solves large tri-objective instances with more non-dominated solutions and solves small instances with less time compared with seeded NSGA-II. (3) Seeded NSGA-II shows its advantage on the number of non-dominated solutions on smaller tri-objective instances.

1. Introduction

Requirement engineering describes the needs and constraints of a real-world software project [1]. Understanding software requirements is an essential goal in requirement engineering. Poor or lack of understanding of users' requirements increases the risk of not meeting users' needs [2]. For a large project, selecting features for the next release plan could be complicated due to the increment of features and functions. It may involve revenue, budget, deadline, and many other aspects.

The Next Release Problem (NRP), proposed by Bagnal et al. [3], is modeled as a single objective problem with maximizing the revenue subject to a given budget. In single-objective form, revenue is the only thing to be discussed, other criteria such as cost, urgency, dependencies among requirements are seen as constraints. And for each constraint configuration, one optimization solution can be found. If trade-off solutions measured with several criteria are preferred, multi-objective NRP is naturally introduced, commonly bi-objective NRP with revenue and cost as objectives appear in many works [4–9].

Multi-Objective Evolutionary Algorithms (MOEA) such as NSGA-II [10] are adopted for MONRP (Multi-objective NRP) in many studies [4,5,7,8,11]. But methods mentioned above could hardly find all optimized solutions due to their randomness and approximation. In [9], an integer linear programming (ILP) method, named ϵ-constraint is introduced to solve bi-objective NRP. ϵ-constraint is an exact method

* Corresponding author.
E-mail addresses: dongshi@mail.ustc.edu.cn (S. Dong), yxxue@ustc.edu.cn (Y. Xue), s.brinkkemper@uu.nl (S. Brinkkemper), liyanfu@tsinghua.edu.cn (Y.-F. Li).

https://doi.org/10.1016/j.infsof.2022.106825
Received 4 May 2021; Received in revised form 30 December 2021; Accepted 3 January 2022
Available online 13 February 2022

Table 1
Requirements in a calculator project.

Requirements	Precedes	Coupling	Cost	Urgency
Basic operations (r1)	Base converter		3	5
Base converter (r2)			2	2
Buttons (r3)	GUI		4	3
Digital display (r4)	GUI		3	4
GUI (r5)		History dialog	5	4
History dialog (r6)		GUI	3	1
Logging (r7)	History dialog		2	2

Table 2
Requests in a calculator project.

Stakeholders	Requests list	Revenue
Alice (s1)	Base converter, logging	10
Bob (s2)	GUI, history dialogue	5

that means it can find optimized solutions rather than approximated ones.

However, the works mentioned above have the following defects:

(1) MOEA could hardly find all non-dominated solutions.
(2) In real-world cases, many factors would be involved in release plan selection. More than two objectives NRP are seldom discussed with exact methods.
(3) For a large NRP with a huge amount of requirements, stakeholders, objectives, and constraints, it is not easy for exact or evolutionary methods to find sufficient non-dominated solutions in a short time.

In our work, we introduce another criterion named *urgency* on measuring the importance of a requirement. Revenue and cost are the basic attributes in the requirement selection problem. Urgency describe how urgent it is that this feature be implemented and in use by the stakeholders [12]. In [4] the importance is treated as how much revenue a requirement could bring. In this paper, we adopt urgency as another objective along with the revenue to describe requirements importance, which some requirements are necessarily selected as soon as possible considering their eagerness. It is evaluated directly or estimated with the usage frequency by stakeholders.

Here we give an example of a requirement selection problem, in Table 1 we describe the requirements of a calculator project, and Table 2 shows stakeholders. Each requirement is associated with cost and urgency ranked from 1(lowest) to 5(highest) and for stakeholders requests list and revenue value are related to them. We also draw Fig. 1 for illustrating *dependencies* between requirements and *requests* from stakeholders. For example, the arrow which starts from "basic operations" towards "base converter" denotes requirement "basic operations" is a prerequisite for selecting requirement "base converter" in the next release plan. And the arrow from "base convertor" towards "Alice" stands for stakeholder "Alice" requests requirement "base converter". Besides the dependencies and requests, we also draw the *coupling* constraint with the dashed arrow. For example, the dashed arrow between "history dialog" and "GUI" denotes that if one is selected in the next release plan, another should also be selected at the same time.

We adopt CWMOIP (Constrained Weighted Multi-Objective Integer Programming) [13] on bi-objective NRP in this paper, innovated by Xue and Li's work [14]. According to their work, CWMOIP is a method for generating all non-dominated solutions on multi-objective feature selection optimization problems with considerable efficiency compared with ϵ-constraint. Then we propose an I-EC (improved "EC", where "EC" stands for ϵ-constraint) inspired by CWMOIP to be a trade-off between ϵ-constraint and CWMOIP (see Section 4.2). For tri-objective NRP, a new method SolRep [15] is adopted. We also compare these methods with an evolutionary algorithm NSGA-II on both bi-objective

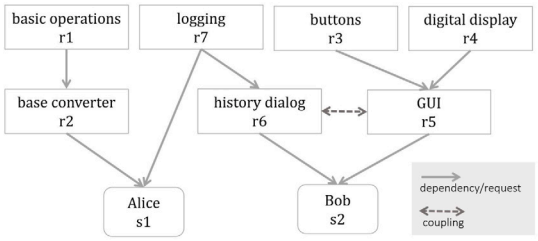


Fig. 1. Dependencies and requests.

NRP and tri-objective NRP. Furthermore, we implement a seeding mechanism for NSGA-II which is beneficial for solving NRP according to [9].

Briefly, our contributions could be summarized as follows:

(1) We adopt CWMOIP and propose I-EC for accelerating ϵ-constraint on bi-objective instances for whole Pareto front solving.
(2) Extend bi-objective NRP with another objective urgency and discuss both it is seen as a constraint and an objective.
(3) We adopt SolRep for tri-objective NRP to find optimized solutions subset rather than all solutions due to the considerable solution size of the tri-objective NRP. It successfully finds enough solutions subject to a given sampling size in large instances and solves faster than seeded NSGA-II in small instances.

In Section 2, we show the formulations of the Next Release Problem, general bi-objective NRP, bi-objective with an additional constraint NRP, and the tri-objective NRP. We also describe some transformation techniques for simplifying the model. Section 3 shows the current methods including ϵ-constraint and NSGA-II. Section 4 proposes CWMOIP, I-EC, and SolRep on solving multi-objective NRP. Research questions, experiments, and results are shown in Section 5. Related works and conclusion are in Sections 6 and 7.

2. Next release problem

In this section, we will show definitions and formulations of Next Release Problems in our study.

2.1. Problem modeling for NRP

Multi-objective optimization is a very important concept for real-world problem abstraction. In our work, given a model M, requirements set is denoted by $RQ(M)$ ($RQ = \{r_1 \dots r_n\}$). For each $r_i \in RQ(M)$, it is a binary value denotes that if the requirement r_i is chosen for next release ($r_i = 1$) or not ($r_i = 0$). We define stakeholders in the same way, stakeholders set $SH(M) = \{s_1 \dots s_m\}$. When employ a binary vector

$$\vec{x} = \{x_1, \dots, x_n, x_{n+1}, \dots, x_{m+n}\} \in \{0,1\}^{m+n}$$

as the next release plan or the solution of NRP. Note that $\forall i \in \{1, \dots, n\} \cdot x_i$ denotes how the requirement r_i evaluated, $x_i = 1$ when requirement r_i is selected. It is the same as $\forall i \in \{n+1, \dots, n+m\} \cdot x_{n+i}$ to s_i, which means s_i is satisfied.[1]

Furthermore, we can give a definition on cost which is relative to requirements as $C = \{c_1, \dots, c_n\}$, c_i is the cost of requirement r_i (x_i). The urgency is defined in a similar way, $U = \{u_1, \dots, u_n\}$. The revenue could be provided by either requirements or stakeholders. When the revenue is associated with the requirement, we can similarly define the revenue, $W = \{w_1, \dots, w_n\}$. When it is provided by stakeholders, we assume

[1] It is possible that revenue are related to requirements directly, for more details see Appendix A.

that the revenue brought by the stakeholder would be counted into the release plan when all his/her requested requirements are selected. And the revenue related to stakeholders: $W = \{w_1, \dots, w_m\}$ where w_j is the revenue provided by stakeholder s_j (x_{n+i} in $\vec{x}$).

Objectives. There are three objectives in our work:

Obj 1. **Revenue** means the revenue gained from a release plan. As revenue is associated with stakeholders and should be maximized, so it should be negative in minimum optimization.

$$\mathcal{F}_1(\vec{x}) = -\sum_{i=1}^{m} w_i x_{n+i} \tag{1}$$

Obj 2. **Cost** means the expense and effort in developing on the release plan which is associated with requirements.

$$\mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n} c_i x_i \tag{2}$$

Obj 3. **Urgency** describes the urgent need of each requirement. A requirement is urgent when its estimated usage frequency is high or very important for some stakeholders.

$$\mathcal{F}_3(\vec{x}) = -\sum_{i=1}^{n} u_i x_i \tag{3}$$

The revenue and urgency should be maximized, so we negate them for minimizing forms. Note that all three objectives are linear polynomials. Thus, the goal of optimization is to find $Min\ (\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3)$ solutions. We would discuss MONRP with two objectives and three objectives separately in later sections.

Constraints. Besides objectives, there are other attributes in a NRP. We use P for describing the *dependency* relationship between requirements. $\forall (i,j) \in P,\ i,j \in \{1,\dots,n\}$ shows when requirement x_j is in the next release, x_i should be selected as the prerequisite, in short, x_i precedes x_j.

Similarly, *requests* Q represents the relationship between requirements and stakeholders. For each request,

$$(i,j) \in Q,\ i \in \{1,\dots,n\}, j \in \{n+1,\dots,n+m\}$$

denotes that requirement x_i is requested by x_j. A feasible solution needs be subject to $x_j \le x_i$. And *coupling* constraints set R describes the relationship between requirements. $\forall (i,j) \in R,\ i,j \in \{1,\dots,n\}$ means that either both of them are selected or non of them are selected in the next release. For convenience, we will use a boolean value function indicating whether expression is violated ($y < x \Leftrightarrow y - x \le 0$).

$$\delta(x) = \begin{cases} 1 & , x > 0 \\ 0 & , x \le 0 \end{cases} \tag{4}$$

Cst 1. **Dependency** means there is a topological order among requirements.

$$\mathcal{G}_1(\vec{x}) = \sum_{(i,j)\in P} \delta(x_j - x_i) \tag{5}$$

Since $(i,j) \in P$ denotes $x_j \le x_i$, if $x_j - x_i > 0$ it means $x_j > x_i$ which leads to a violation. Then $\mathcal{G}_1(\vec{x})$ indicates the number of violated dependencies and $\mathcal{G}_1(\vec{x}) = 0$ when all dependencies are satisfied.

Cst 2. **Request** means the relationship between requirements and stakeholders, a stakeholder can request several requirements meanwhile a requirement can be requested by several stakeholders.

$$\mathcal{G}_2(\vec{x}) = \sum_{(i,j)\in Q} \delta(x_j - x_i) \tag{6}$$

It indicates how many requests are violated and $\mathcal{G}_2(\vec{x}) = 0$ when all requests are contented.

Cst 3. **Coupling** describes the relationship between two requirements that both of them should be always selected or not be selected at the same time.

$$\mathcal{G}_3(\vec{x}) = \sum_{(i,j)\in R} \delta(|x_i - x_j|) \tag{7}$$

$\mathcal{G}_3(\vec{x}) = 0$ when all couplings are not violated.[2]

We say a solution is *correct* when it does not violate any constraints. Still for example in Fig. 1, the dependency such as (buttons, GUI) which denotes buttons should be implemented as GUI is chosen for the next release, is a constraint of the calculator project. And the request (logging, Alice) is also a constraint as the stakeholder "Alice" requests the requirement "logging". As requirements "base converter" and "logging" are in the next release, the stakeholder "Alice" would be satisfied subject to the request constraints (base converter, Alice) and (logging, Alice), thus "Alice" would provide revenue as revenue is associated with stakeholders.

Dominate. For multi-objective optimization, commonly we cannot compare a pair of solutions with a specific order. Given two release plan $\vec{x}_1$ and $\vec{x}_2$, their k-dimension objectives are $\mathcal{F}_k(\vec{x}_1)$ and $\mathcal{F}_k(\vec{x}_2)$.

Definition 1. We say solution $\vec{x}_1$ **dominates** $\vec{x}_2$ denoted by ($\vec{x}_1 \prec \vec{x}_2$) if

$$\begin{aligned} &\forall i \in \{1,\dots,k\} \cdot \mathcal{F}_i(\vec{x}_1) \le \mathcal{F}_i(\vec{x}_2) \\ &\exists j \in \{1,\dots,k\} \cdot \mathcal{F}_i(\vec{x}_1) < \mathcal{F}_j(\vec{x}_2) \end{aligned} \tag{8}$$

otherwise $\vec{x}_1 \not\prec \vec{x}_2$, $\vec{x}_1$ cannot dominate $\vec{x}_2$.

Definition 2. A solution $\vec{x}$ and a set of solutions $S_{\vec{x}}$, $\vec{x}$ is **non-dominated** if

$$\forall \vec{x}_i \in S_{\vec{x}} \cdot \vec{x}_i \not\prec \vec{x} \tag{9}$$

Pareto front. Correct and non-dominated solutions are *Pareto-optimal* solutions. All Pareto-optimal solutions make the *true Pareto Front* (or just Pareto front). In our work, we also use "optimized solutions" and "Pareto solutions" indicating solutions on the Pareto front.

2.2. Transformations

We adopt several transformations in this work for simplifying problems. The main purpose of these transformations is to eliminate or reduce the number of explicit constraints for the benefit of NSGA-II since it cannot handle constraints natively [9]. Besides, these transformations reduce the number of decision variables which reduce the computational complexity for integer linear programming in theory.

Coupling removal. This removes the coupling constraints or logic AND constraints [9]. For each element x_i, it belongs to an equivalence set $\{x_i, x_{(1)}, x_{(2)}, \dots\}$, where $x_i = x_{(1)} = x_{(2)} = \dots$. We can use x_i to indicate the equivalence set, so these relations are no longer necessary. In Fig. 1, requirements "GUI" and "history dialog" are constrained by a coupling constraint, which means they will also share the same value. Thus, we can use a single "GUI (r5)" stands for "GUI (r5) and history dialog (r6)" and meanwhile the dependency "logging" precedes "history dialog" becomes the dependency "logging" precedes "GUI" and "Bob" requests "history dialog" is replaced by "Bob" requests "GUI", then the requirement "history dialog (r6)" is removed. The cost of the new "r5" becomes 8 and the urgency becomes 5 accordingly.

[2] Actually there is another kind of constraint in certain instance, see Appendix B.

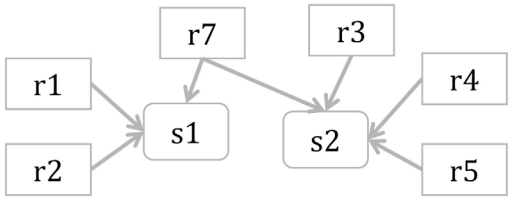


Fig. 2. Remove dependencies and couplings.

Dependency removal. This removes the dependencies [3]. For short, a requirement is requested by some stakeholders, its prerequisites are also requested by them. For a clear explanation, Fig. 2 shows requests including original requests and those generated from dependencies. In Fig. 1, requirement "digital display" and "buttons" precede "GUI". When "Bob" is satisfied with "GUI" is selected, "buttons" and "digital display" should be also selected. It is equivalent to "Bob" requests these two requirements and that is what this transformation does.

To find all prerequisites, we employ a set $Pre(x_j)$. Initially, it would contain all direct prerequisites of (x_j),

$$\forall (i,j) \in P \cdot x_i \in Pre(x_j)$$

Then, by recursively do this step for each element in $Pre(x_j)$,

$$Pre(x_j) = Pre(x_j) \bigcup_{x_i \in Pre(x_j)} Pre(x_i)$$

until $Pre(x_j)$ will not change anymore.

Hence, given a requirement x_i and its prerequisites $Pre(x_i) = \{x_{i,1}, x_{i,2}, \dots\}$, $\forall (i,j) \in Q$, update Q with new constraints $x_j \le x_{i,s}, x_{i,s} \in Pre(x_i)$. For example in former Fig. 1, though stakeholder "Alice" requests "base converter" and "logging", "Alice" also requests "basic operations" after this transformation because requirement "basic operations" precedes "base converter".

Stakeholder removal. This removes the requests for evolutionary methods. When a release plan is made, the requirements are chosen meanwhile the revenue is determinate. It could be simply written as $s_j = (r_{(1)} \wedge r_{(2)} \wedge \cdots \wedge r_{(i)} \wedge \dots)$, $\forall (j,(i)) \in Q$. When all requests for a stakeholder are satisfied, it contributes its revenue. Thus decision variables for stakeholders will not occur in an individual, or a "chromosome", revenue evaluation would be calculated with other objectives at the same time. Note that this transformation could not be applied with dependency removal transformation together.

Still using the example in Fig. 1, for an evolutionary method, decision variables would not contain "Alice" and "Bob". As requirements are decided in the next release, such as "basic operations", "base converter" and "logging", then "Alice" is satisfied as all requests are selected. It is unnecessary to use more variables to indicate whether "Alice" is satisfied but just add revenue provided by "Alice".

2.3. General bi-objective NRP

First, let us focus on two objectives: revenue and cost. The general bi-objective NRP should maximize revenue and minimize cost, meanwhile, a release plan should be correct which means it should not violate any dependencies or request constraints.

$$\begin{aligned} Min \quad & \mathcal{F}_1(\vec{x}) = -\sum_{i=1}^{m} w_i x_{n+i} \\ Min \quad & \mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n} c_i x_i \\ s.t. \quad & \mathcal{G}_1(\vec{x}) + \mathcal{G}_2(\vec{x}) + \mathcal{G}_3(\vec{x}) = 0 \end{aligned} \tag{10}$$

The example problem in Fig. 2 can be formed as:

$$\begin{aligned} Min \quad & -10s_1 - 5s_2 \\ Min \quad & 3r_1 + 2r_2 + 4r_3 + 3r_4 + 8r_5 + 2r_7 \\ s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\ & + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\ & + \delta(s_2 - r_7) = 0 \end{aligned} \tag{11}$$

2.4. Bi-objective NRP with additional constraint

Further, we adopt another criterion urgency in a formulation as a constraint but not an objective.

$$\begin{aligned} Min \quad & \mathcal{F}_1(\vec{x}) = -\sum_{i=1}^{m} w_i x_{n+i} \\ Min \quad & \mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n} c_i x_i \\ s.t. \quad & \mathcal{F}_3(\vec{x}) = -\sum_{i=1}^{n} u_i x_i \le L \\ & \mathcal{G}_1(\vec{x}) + \mathcal{G}_2(\vec{x}) + \mathcal{G}_3(\vec{x}) = 0 \end{aligned} \tag{12}$$

The constraint $\mathcal{F}_3(\vec{x}) = -\sum_{i=1}^{n} u_i x_i \le L$ is from the objective $\mathcal{F}_3$ mentioned before, it means the minimal urgency value that should be satisfied. Normally we can set $L = -l\ \Sigma_i u_i$ where $l \in (0.0, 1.0)$ is a scaling factor. Note that a higher urgency value is preferred as the value of urgency is negative, so $\mathcal{F}_3$ should be less or equal than that bound L.

Still use example in Fig. 2:

$$\begin{aligned} Min \quad & -10s_1 - 5s_2 \\ Min \quad & 3r_1 + 2r_2 + 4r_3 + 3r_4 + 8r_5 + 2r_7 \\ s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\ & + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\ & + \delta(s_2 - r_7) \\ & + \delta(5r_1 + 2r_2 + 3r_3 + 4r_4 + 5r_5 + 2r_7 - 21l) \\ & = 0 \end{aligned} \tag{13}$$

2.5. Tri-objective NRP

Bi-objective NRP are discussed in many former works [4–9]. For further discussion, we adopt revenue, cost, and urgency as objectives together and define tri-objective NRP.

$$\begin{aligned} Min \quad & \mathcal{F}_1(\vec{x}) = -\sum_{i=1}^{m} w_i x_{n+i} \\ Min \quad & \mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n} c_i x_i \\ Min \quad & \mathcal{F}_3(\vec{x}) = -\sum_{i=1}^{n} u_i x_i \\ s.t. \quad & \mathcal{G}_1(\vec{x}) + \mathcal{G}_2(\vec{x}) + \mathcal{G}_3(\vec{x}) = 0 \end{aligned} \tag{14}$$

And for example in Fig. 2, the tri-objective form is described as:

$$\begin{aligned} Min \quad & -10s_1 - 5s_2 \\ Min \quad & 3r_1 + 2r_2 + 4r_3 + 3r_4 + 8r_5 + 2r_7 \\ Min \quad & -5r_1 - 2r_2 - 3r_3 - 4r_4 - 5r_5 - 2r_7 \\ s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\ & + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\ & + \delta(s_2 - r_7) = 0 \end{aligned} \tag{15}$$

Algorithm 1: Multi-Objective *EConstraint()*

input : M: NRP Model, t: current objective, $Cons$: constraints
output: E: Solutions

$E \leftarrow \emptyset$;
// calculate the theory boundary of $\mathcal{F}_t$
$f_t^L, f_t^U \leftarrow$ `getObjTheoBound`$(M, \mathcal{F}_t)$;
if $t = 1$ **then**
  $E \leftarrow$ `bintprog`$(Cons, \mathcal{F}_t)$;
else
  // set $\mathcal{F}_t$ bound
  for $l \leftarrow f_t^U; l \geq f_t^L; l \leftarrow l - 1$ **do**
    $Cons \leftarrow Cons \cup \{\mathcal{F}_t \leq l\}$;
    $ME \leftarrow$ `EConstraint`$(M, t-1, Cons)$;
    $E \leftarrow E \cup ME$;
  end
end
return E;

3. Existing methods

In this section, we would describe methods that are applied on MONRP before, namely ϵ-constraint and NSGA-II. ϵ-constraint is an exact method aiming at solving the whole Pareto front, meanwhile, NSGA-II is a heuristic method aiming at finding solutions rapidly which hardly guarantees to get the whole Pareto front.

3.1. ϵ-constraint

The use of ϵ-constraint for solving bi-objective NRP is adopted in [9]. Its main idea is to maintain only one objective and turn another objective into constraints with given boundaries. Each constraint's upper bounds will iterate from maximum to minimum of the corresponding objective. The pseudocode is shown at Algorithm 1.

Notice that, `EConstraint` is defined as a recursive function. Function `getObjTheoBound`$(M, \mathcal{F}_t)$ calculates the theoretical boundaries of current objective with NRP model M and returns the lower bound and the upper bound separately. `bintprog`$(Cons, \mathcal{F}_t)$ calls BIP function for $\mathcal{F}_t$, where $Cons$ contain constraints from `Conj`(M)(constraints in M) and those from objectives.

The initial call of ϵ-constraint is `EConstraint`(k). Then for each iteration on bound, it recursive calls `EConstraint`$(k-1)$. When it is to `EConstraint`(1), BIP solves the optimization problem and return the solutions.

Algorithm 1 is of the time complexity of $O(n^{k-1})$ if we consider that `bintprog()` would costs constant time,[3] where k is the number of objectives and n stands for the estimated range of each objective. For example, it would solve $(max(\mathcal{F}_2) - min(\mathcal{F}_2))$times in general bi-objective NRP. Note that all coefficients in the problem we discussed are integers, l decreases by the step size 1 for finding the whole Pareto front.

Let we explain ϵ-constraint with example (11). It has two objectives, we reduce the second objective in this case. The bounds of the second objective are 0 and 22. Thus the problem is transformed into 22 single objective optimization problem,

$$\begin{aligned} Min \quad & -10s_1 - 5s_2 \\ s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\ & + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\ & + \delta(s_2 - r_7) \\ & + \delta(L - 3r_1 - 2r_2 - 4r_3 - 3r_4 - 8r_5 - 2r_7) \\ & = 0 \end{aligned} \tag{16}$$

where L is set from 22 to 0.

[3] The official document of Cplex suggests setting time limits for practical usage. That is due to the complexity of `bintprog()` is NP-hard, and it would cost much time for solving a complex optimization problem (such as on classic-2 or classic-4) according to our experience. Fortunately, it would not take over 1 s on average on other instances.

Algorithm 2: NSGA-II Main Loop in round t

input : M: NRP Model, *generation*: iteration times for generation, *population*: size of population
output: E: Solutions

$t \leftarrow 0$;
$P_t \leftarrow$ `initializePopulation`$(population)$;
$Q_t \leftarrow$ `makeNewPopulation`(M, P_t);
while $t \leq generation$ **do**
  $R_t \leftarrow P_t \cup Q_t$;
  $F \leftarrow$ `fastNonDominatedSort`(R_t) ;
  $P_{t+1} \leftarrow \emptyset, i \leftarrow 1$;
  while $|P_{t+1}| + |F_i| \leq N$ **do**
    `crowdingDistanceAssignment`(F_i) ;
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$;
    $i \leftarrow i + 1$;
  end
  `fill`$(P_{t+1}$, `sort`$(F_i))$;
  $Q_{t+1} \leftarrow$ `makeNewPopulation`(M, P_{t+1});
  $t \leftarrow t + 1$;
end
$E \leftarrow$ `fastNonDominatedSort`$(P_t \cup Q_t)$;
return E ;

3.2. NSGA-II

As the evolutionary algorithm used in our work, NSGA-II is proposed by [10] and solves bi-objective NRP in [4,5,7,11]. Algorithm 2 shows the description of the main loop in NSGA-II.

Algorithm 2 shows how NSGA-II updates its population. Initially, t is initialized to 0 and P_t is initialized randomly via function `initializePopulation`. Q_t is initialized according to the P_t. For the outer loop, each outer iteration shows how a new population is generated and how it updates the P_t. First, R_t is constructed from current population and offspring in generation t.

`fastNonDominatedSort` is called for sorting non-dominated fronts from R_t. Then it comes to the inner loop, distance calculation would be applied to front F_i for each inner iteration. When it leaves the inner loop, if the size of the last front F_i in the loop is bigger than vacancy, F_i would be sorted and fill the population. In the end, offspring population Q_{t+1} would be generated by the selection, crossover, and mutation operations, new individuals would be evaluated by the NRP Model M to find out whether they are feasible and the value of objectives. After the end of the last generation, E collects all non-dominated solutions by sorting P_t and Q_t.

The seeding mechanism is adopted in initializing the population and a repair mechanism is adopted both in population initiation and offspring generation in NSGA-II. As the algorithm creates the population randomly, it also adopts some "good" solutions in the population. Practically, we randomly select several solutions on the Pareto front solved by the exact method. It adopts the idea mentioned in the work [9]. We also repair the solution as an individual is evaluated both in initialization and generation. Only request constraints are repaired.

Algorithm 3: *CWMOIP()*

input : M: NRP Model, t: current objective, $Cons$: constraints, f: reduced objective
output: E: Solutions

$E \leftarrow \emptyset$;
$w_t \leftarrow 1$;
for $i \leftarrow 2; i \leq k; i \leftarrow i+1$ **do**
  $f_i^L, f_i^U \leftarrow$ `GetObjRealBound`$(M, \mathcal{F}_i)$;
  $w_t \leftarrow w_t/(f_i^U - f_i^L + 1)$;
end
if $t = 1$ **then**
  $ME \leftarrow$ `bintprog`$(Cons, f)$;
else
  $l \leftarrow f_t^U$;
  while *true* **do**
    $f \leftarrow$ `addObjFuncSuffix`$(f, w_t \cdot \mathcal{F}_t)$;
    $Cons \leftarrow Cons \cup \{\mathcal{F}_t \leq l\}$;
    $ME \leftarrow$ `CWMOIP`$(M, t-1, Cons, f)$;
    if $ME = \emptyset$ **then**
      break ;
    $E \leftarrow E \cup ME$;
    $l \leftarrow max(\mathcal{F}_t(\vec{x}), \vec{x} \in ME) - 1$
  end
end
return E;

For the constraint that x_i requests x_j and $x_i = 1, x_j = 0$, we repair x_j to 1. Note that the stakeholders are removed for NSGA-II in Section 2.2, x_i we use here is a logic variable rather than a decision variable.

4. Our approach

4.1. CWMOIP

CWMOIP is another objective reduction technique used in multi-objective optimization, proposed by Özlen et al.. It could also find the whole Pareto front as ϵ-constraint. Besides CWMOIP accelerate the solving time by two key improvements: (1) CWMOIP uses BIP for objectives boundaries, subject to the conjunction of constraints `conj(M)`. (2) CWMOIP reduces objectives by a weighted method to avoid generating dominated solutions. (3) When iterating bound l for each objective, CWMOIP will calculate the next bound with solutions instead of using a size-fixed step.

In Algorithm 3 we show the general steps of CWMOIP. The initial call of this procedure is `CWMOIP`$(M, k, \emptyset, \mathcal{F}_1)$. Inside the `CWMOIP`, boundaries of $\mathcal{F}_i, i \in 2, \ldots, k$ are solved first. We use them to get the weight of $\mathcal{F}_k$, $w_k = 1/(\prod_{i=2}^{k}(f_i^U - f_i^L + 1))$. Thus, $\mathcal{F}_k$ could be reduced as a weighted addend in f. For bound of $\mathcal{F}_k$, it is initially f_k^U. Then recursively call `CWMOIP`$(M, k-1, Cons, f)$ after updating constraints $Cons$. If none was found, there would not be solutions after this iteration, so break; otherwise, add solutions in E. Finally, update the new bound l by subtracting 1 from the upper bound of solutions in this iteration. After that, `CWMOIP`$(k-1)$ would call `CWMOIP`$(k-2)$, recursively to `CWMOIP`(1) and employ BIP to find a solution.

From Fig. 3, we can see how CWMOIP reduces the solving time in the bi-objective minimization problem. Assume we have two objectives f_1 and f_2 and the whole Pareto front is composed by four solutions E_1, E_2, E_3, E_4. ϵ-constraint would iterate from the upper bound l_0 to the lower bound l_6 and collect solutions. CWMOIP would use the maximum of solutions objective value minus 1 as the next bound for constraint $f_1 \leq l$. For example, the bound is initialized as l_0, and E_0 is found, so we can use $l_1 = f_1(E_0) - 1$ as the next bound and E_1 is found. Then l would be updated as $l_4 = f_1(E_1) - 1$ accordingly, until $l = l_6$ cannot let solver find more solutions.

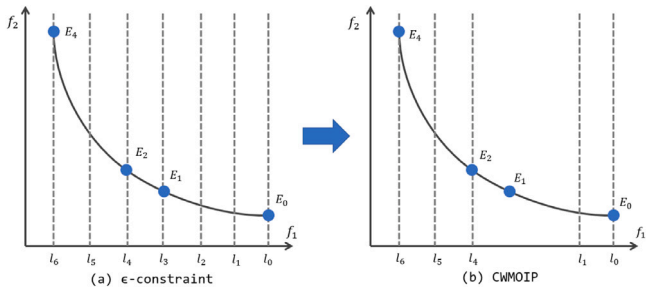


Fig. 3. Comparison between ϵ-constraint and CWMOIP.

According to [13], the maximum number of recursion is $\frac{|E|(|E|+1)\ldots(|E|+k-2)}{2\cdot 3\cdot\ldots(k-1)}$.

Still use Fig. 1 as an example, assume we are solving bi-objective form NRP on this instance. We choose cost as the reduced objective, the upper bound of cost is 22. So ϵ-constraint solves maximal revenue value subject to cost value less or equal to 22. Then cost bound is set as 22, 21, 20, ... till it touches the lowest bound 0. ϵ-constraint would solve 23 times. But CWMOIP would behave differently. When cost bound is set 22, the release plan is to select all requirements. For the next round, the cost value is set to 21 and the best release plan {basic operations, base converter, logging} is decided as its cost is 7 and revenue is 10. Then the cost bound is updated to $7 - 1 = 6$ and the solution with 0 revenue and 0 cost is found. Then the algorithm would terminate as it cannot find any more solutions. The number of solving times decreases from 23 to 4.

We would explain CWMOIP with an example (16).

$$
\begin{aligned}
Min \quad & -10s_1 - 5s_2 \\
& + (3r_1 + 2r_2 + 4r_3 + 3r_4 + 8r_5 + 2r_7)/22 \\
s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\
& + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\
& + \delta(s_2 - r_7) \\
& + \delta(L - 3r_1 - 2r_2 - 4r_3 - 3r_4 - 8r_5 - 2r_7) \\
& = 0
\end{aligned}
\tag{17}
$$

L is updated by the solutions found during the algorithm running.

4.2. I-EC

Inspired by CWMOIP, we can make ϵ-constraint another method simply replacing its fixed-step iteration with the way used by CWMOIP. But the reduction of the objective still maintains the ϵ-constraint way. That is based on the trade-off between eliminating weak dominated solution(very close to but not on the Pareto front) and objective polynomial complexity. As we can see in bi-objective NRP, the objective revenue and costs are constituted by stakeholders s_j and requirement r_i separately. ϵ-constraint would naturally use the last objective as the reduced objective as CWMOIP would use a weighted sum of each objective.

Here we would like to explain why the weighted objective can avoid weak dominated solutions. Assume there is a bi-objective optimization problem as shown in Fig. 4, the objectives are f_1 and f_2, reduced objective is f_1. In un-weighted situation (a), bound l_0 lead to the solution E_1, then bound is updated as l_1 and E_2 is found. If $f_2(E_1) = f_2(E_2)$, E_2 dominates E_1 (as $f_1(E_2) \leq l_1 < f_1(E_1)$). When we apply a weighted tactic, the objective become $f_2 + w_1 f_1$ (b). Still we assume $f_2(E_1') = f_2(E_1)$ in this case, the objective value of these two solutions are $f_2(E_1) + w_1 f_1(E_1) < f_2(E_1') + w_1 f_1(E_1')$. When $l = l_1$, E_1 is

Algorithm 4: I-EC *ImprEC()*

```
input : M: NRP Model, t: current objective, Cons: constraints,
        f: reduced objective
output: E: Solutions
E ← ∅ ;
f_i^L, f_i^U ← GetObjRealBound(M, F_i);
if t = 1 then
    ME ← bintprog(Cons, F_t);
else
    l ← f_t^U ;
    while true do
        Cons ← Cons ∪ {F_t ≤ l};
        ME ← ImprEC(M, t − 1, Cons, f);
        if ME = ∅ then
            break ;
        E ← E ∪ ME ;
        l ← max(F_t(x⃗), x⃗ ∈ ME) − 1
    end
end
return E;
```

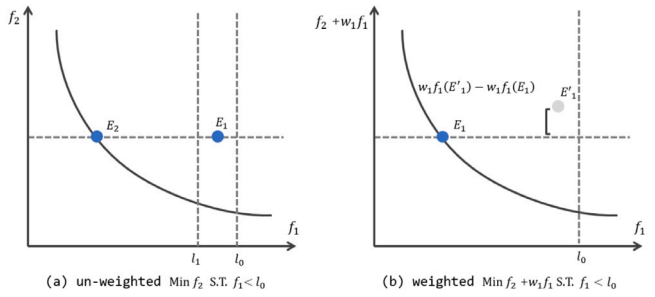


Fig. 4. Comparison between un-weighted objective and weighted objective.

solved but not E'_1. In I-EC weak dominated solutions are not ruled out due to its implementation as (b). For example, assume there are two solutions $(1,1)$ and $(1,2)$ in an optimization problem with two minimizing objectives with the second objectives is restricted to ≤ 2 and optimize the first objective. The solver might select $(1,1)$ or $(1,2)$ and could not figure out which one is on the Pareto front as they have the same value in the first dimension. As we apply a weighted tactic such as giving a factor 0.1 to the second objective, the optimization value comes to be 1.1 and 1.2 for solutions $(1,1)$ and $(1,2)$. The solver could find the solution $(1,2)$ is not a non-dominated solution in a single optimization.

For a clear explanation, we use example (16) shown in Section 3.1.

$$
\begin{aligned}
Min \quad & -10s_1 - 5s_2 \\
s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\
& + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\
& + \delta(s_2 - r_7) \\
& + \delta(L - 3r_1 - 2r_2 - 4r_3 - 3r_4 - 8r_5 - 2r_7) \\
& = 0
\end{aligned}
\tag{18}
$$

L is updated by the solutions found during the algorithm running. Note that the optimized objective is the same as the ϵ-constraint. It uses 2 decision variables but CWMOIP uses 8. But in a series of solving with the appliance of ϵ-constraint, the weak dominated solutions could be ruled out by solutions from other solving. Such as in Fig. 4, as the f_1 is restrict to l_1, it could find at least one solution (E_2) dominating E_1. As we restrict the f_1 from big to small, the weak dominated solutions would be ruled out, which avoids the weak dominated solutions problem.

4.3. Tri-objective SolRep

Inspired by normal constraint [16] and SolRep [15], we adopt another search-based method, each solves start from a sampled point on the *utopia plane*. The utopia plane is a hyperplane to approximate the Pareto front, a set of evenly distributed reference points on the utopia plane would result in a set of evenly distributed solutions on the Pareto front, after the projection along the normal vector of utopia plane [15]. In our tri-objective NRP, we can find two *anchor points* by finding each objective's optimal solution. Then the uniform sampling is applied on the line cross two anchors, which is the utopia plane in our problem. Thus we have points on the utopia plane and employ an ILP solver to find non-dominated solutions. We would explain how anchors are calculated with the pseudocode.

Algorithm 5: Tri-objective SolRep

```
1  y⃗_anchor,1 ← bintprog(Cons, F_1) ;
2  y⃗_anchor,2 ← bintprog(Cons, F_2) ;
3  d⃗ ← (y⃗_anchor,2 − y⃗_anchor,1)/(N + 1);
4  P ← ∅;
5  p⃗ ← y⃗_anchor,1 ;
6  while p⃗ ≠ y⃗_anchor,2 do
7      p⃗ ← p⃗ + d⃗;
8      P ← P ∪ {p⃗} ;
9  end
10 E ← ∅ ;
11 foreach p⃗ ∈ P do
12     if E contains p⃗ then
13         continue
14     end
15     Cons′ ← Cons ∪ {F_1 ≤ p⃗_1, F_2 ≤ p⃗_2};
16     ME ← bintprog(Cons′, F_3);
17     E ← E ∪ ME;
18 end
19 E ← non-dominated-sort(E);
```

The first two lines find the anchor points with `bintprog`, they are the optimization solutions related to each objective and ignoring others. In our bi-objective or tri-objective case, the anchors are "select every requirement" (maximal revenue and urgency) and "do not select anyone" (minimal cost). Then the direction $\vec{d}$ between anchors $\vec{y}_{anchor,1}, \vec{y}_{anchor,2}$ is calculated. Note that N is the number of sampling points. From lines 5 to 9, we initialize point $\vec{p}$ as one anchor point and iterate it to another anchor point with a fixed step $\vec{d}$. Then we get N uniform distribution points on the utopia plane. For each point, we reduce $\mathcal{F}_1$ and $\mathcal{F}_2$ by their value in these dimensions as bounds. Notice that E *contains* $\vec{p}$ denotes that current point $\vec{p}$ is contained in former searching space, thus we skip these points as it may lead to duplicated solutions.

We illustrate it with Fig. 5, there are 3-dimension objective space, namely x_1, x_2, x_3 for three objectives. y_1 and y_2 are two anchor points on x_1, x_2, Utopia plane is the line connected them together. Then we sample four points p_1 to p_4 on this line, with the same interval. For each point p_i, we would use its x_1 direction value p_{i,x_1} and p_{i,x_2} for x_2 to set bounds of objective constraints $x_1 \leq p_{i,x_1}$ and $x_2 \leq p_{i,x_2}$. Thus there are only one objective x_3 left, we apply an ILP solver on it and get E_1, E_2, E_3, E_4 subject to points p_1, p_2, p_3, p_4.

To explain much clearly, we use the former example mentioned in Tables 1 and 2. Requests are shown in Fig. 2 after transformations. We assume the objectives space (x, y, z) where x is for the value of

Table 3
Non-dominated solutions of general bi-objective NRP, found by ϵ-constraint (A), I-EC (B) and CWMOIP (C).

NRP	$\|A\|$	$\|B\|$	$\|C\|$	A Time (s)	B Time (s)	C Time (s)	A IGD	B IGD	C IGD
classic-1	465	465	465	29.81	22.27	23.85	0	0	0
classic-2	4540	4540	4540	48 869.28	48 560.56	49 328.06	0	0	0
classic-3	6296	6296	6296	1195.47	1152.29	1622.61	0	0	0
classic-4	13 489	13 489	13 489	33 594.48	31 526.31	60 617.34	0	0	0
classic-5	2898	2898	2898	259.58	223.38	285.66	0	0	0
realistic-e1	10 331	10 331	10 331	1360.17	1123.20	1515.93	0	0	0
realistic-e2	10 573	10 573	10 573	2261.23	1698.54	2371.83	0	0	0
realistic-e3	8344	8344	8344	773.49	654.21	887.82	0	0	0
realistic-e4	8303	8303	8303	1238.05	881.66	1387.60	0	0	0
realistic-g1	9280	9280	9280	714.27	539.18	701.26	0	0	0
realistic-g2	6393	6393	6393	741.00	450.57	556.38	0	0	0
realistic-g3	8457	8457	8457	619.10	453.86	597.05	0	0	0
realistic-g4	6171	6171	6171	487.49	316.42	395.87	0	0	0
realistic-m1	13 773	13 773	13 773	2884.71	2234.47	3336.91	0	0	0
realistic-m2	12 933	12 933	12 933	3413.91	2486.68	3568.52	0	0	0
realistic-m3	12 624	12 624	12 624	1759.64	1565.75	2034.87	0	0	0
realistic-m4	11 547	11 547	11 547	1746.29	2077.58	2702.83	0	0	0
Baan	793	793	793	22.37	3.94	4.90	0	0	0
Word	189	189	189	3.06	0.64	1.06	0	0	0
ReleasePlanner	71	71	71	51.53	0.16	0.24	0	0	0

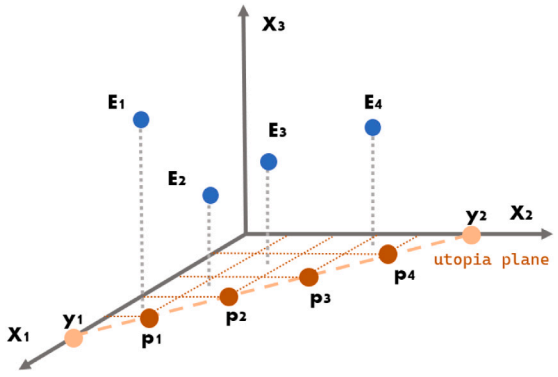


Fig. 5. Tri-objective SolRep Example.

revenue, y is for the value of cost and z is for the value of urgency. Two anchor points we can deduce from the problem are $(0, 0, 0)$ and $(15, 22, 21)$. Assume the sampling size is 2, the sampling points are $(5, 7.33, 7)$, $(10, 14.67, 14)$together with two anchors $(0, 0, 0)$, $(15, 22, 21)$ For each sampling point (x, y, z), there is an optimization problem:

$$\begin{aligned} Min \quad & -5r_1 - 2r_2 - 3r_3 - 4r_4 - 5r_5 - 2r_7 \\ s.t. \quad & \delta(s_1 - r_1) + \delta(s_1 - r_2) + \delta(s_1 - r_7) \\ & + \delta(s_2 - r_3) + \delta(s_2 - r_4) + \delta(s_2 - r_5) \\ & + \delta(s_2 - r_7) + \delta(10s_1 + 5s_2 - x) \\ & + \delta(y - 3r_1 - 2r_2 - 4r_3 - 3r_4 - 8r_5 - 2r_7) \\ & = 0 \end{aligned} \tag{19}$$

5. Evaluation

5.1. Research questions

1. **RQ1** Would I-EC and CWMOIP accelerate the Pareto front solving on bi-objective NRP?
2. **RQ2** How would I-EC and CWMOIP behave on bi-objective NRP with another constraint?
3. **RQ3** How efficient would exact methods and NSGA-II deal with problems with tri-objective NRP?

5.2. Setup

5.2.1. Datasets

Our work adopts synthetic and realistic NRP instances. There are 5 synthetic datasets (namely classic datasets) generated and 12 realistic datasets collected in [6]. They are all public on the Internet,[4] Realistic datasets are collected from bug repositories in three open source software projects: *Eclipse*, *Mozilla*, and *Gnome*. Each dataset is described as requirements cost, stakeholders revenue, dependencies among requirements, and requests between stakeholders and requirements.

Additionally, we also adopt realistic datasets from ReleasePlanner projects used in [17]: *Word* and *ReleasePlanner*. In these datasets, each stakeholder gives a revenue and urgency value for each requirement(for ReleasePlanner dataset, its value, and frequency of use). The cost is associated with the requirement. Each stakeholder s_j have an weight $weight_j$. In our work, we calculate revenue follows the way in [18]. Assume revenue for stakeholder s_j and requirement r_i is $w_{i,j}$, $w_i = \Sigma_j weight_j w_{i,j} / \Sigma_j weight_j$, it is the same to urgency value $u_i = \Sigma_j weight_j u_{i,j} / \Sigma_j weight_j$. (Floating-point numbers are rounded to integers.)

Another dataset Baan [19] is also adopted. In this dataset, revenue is provided by requirements, and costs are calculated with multiple teams' cooperation (details are in Appendix B).

5.2.2. Environment[5]

We use `jMetal` [20] for NSGA-II in Java as our evolutionary methods. And `Cplex(12)` [21] is employed as our ILP solver in exact methods.

The experiments are performed on Ubuntu 20.04 LTS, AMD Ryzen 5 3600 with 48 GB RAM.

The seeding mechanism is adopted and seeds are randomly selected non-dominated solutions found by the exact methods. Seeds are used in the progress during population initialization and offspring population creation [22,23].

The parameters of NSGA-II follow work in [9], population size is 500, crossover probability is 0.8, mutation probability is $1/n$ where n is the size of decision variables, maximum evaluation is 100 000 (200

[4] Classic and realistic datasets http://oscar-lab.org/people/jxuan/page/project/nrp/index.htm, MSWord, and ReleasePlanner https://sites.google.com/site/mrkarim/data-sets.

[5] Codes are published at https://github.com/Osinovsky/NRP_MOIP.

Table 4
Non-dominated solutions of general bi-objective NRP, found by I-EC (A), seeded NSGA-II (B).

NRP	$\|A\|$	$\|B\|$	$\|Pareto(A)\|$	$\|Pareto(B)\|$	A Time (s)	B Time (s)	A IGD	B IGD	A HV	B HV	A SP	B SP
classic-1	465	205.1	465	61.5	22.27	82.30	0	41.77	1.32_{10^6}	97%	2.70	5.77
classic-2*	4540	370.9	4540	12.0	48 560.56	545.95	0	268.79	3.70_{10^7}	94%	1.10	11.91
classic-3	6296	380.2	6296	9.4	1152.29	557.81	0	120.9	5.86_{10^7}	97%	1.05	11.37
classic-4*	13 489	394.4	13 489	7.3	31 526.31	1909.33	0	475.3	2.18_{10^8}	94%	0.79	17.18
classic-5	2898	419.0	2898	6.1	223.38	613.24	0	151.39	5.61_{10^7}	96%	7.47	25.09
realistic-e1	10 331	371.9	10 331	7.8	1123.20	725.54	0	222.16	1.34_{10^8}	97%	0.84	12.4
realistic-e2	10 572	362.4	10 572	9.2	1698.54	930.79	0	806.38	1.52_{10^8}	91%	1.07	10.71
realistic-e3	8344	364.7	8344	8.7	654.21	552.79	0	233.07	8.95_{10^7}	96%	0.85	10.01
realistic-e4	8303	372.3	8303	9.9	881.66	644.49	0	87.95	8.83_{10^7}	98%	0.98	12.25
realistic-g1	9280	357.6	9280	11.3	539.18	493.24	0	540.82	1.09_{10^8}	94%	0.86	10.4
realistic-g2	6393	380.93	6393	13.2	450.57	487.83	0	390.54	7.56_{10^7}	90%	1.83	10.1
realistic-g3	8457	366.7	8457	7.1	453.86	445.66	0	80.39	9.64_{10^7}	98%	1.14	11.73
realistic-g4	6171	369.3	6171	13.1	316.42	399.76	0	56.11	5.97_{10^7}	98%	1.21	10.63
realistic-m1	13 773	412.4	13 773	4.9	2234.47	965.65	0	288.81	2.25_{10^8}	96%	0.66	17.5
realistic-m2	12 933	382.8	12 933	7.0	2486.68	1073.91	0	236.94	1.96_{10^8}	97%	0.85	14.95
realistic-m3	12 624	396.9	12 624	4.2	1565.75	831.97	0	428.64	1.93_{10^8}	96%	0.80	13.97
realistic-m4	11 547	378.8	11 547	7.5	2077.58	841.28	0	384.18	1.49_{10^8}	95%	0.78	12.93
Baan	793	397.1	793	203.5	3.94	4.48	0	61.87	9.56_{10^7}	100%	43.21	60.03
MSWord	189	186.2	189	181.1	0.64	6.88	0	0.09	2.45_{10^5}	100%	4.13	4.16
ReleasePlanner	71	71	71	71	0.16	3.56	0	0	2.96_{10^6}	100%	313.91	313.91

Table 5
Non-dominated solutions of general bi-objective NRP with additional constraint on synthetic datasets, found by ϵ-constraint (A), I-EC (B) and CWMOIP (C).

NRP	$\|A\|$	$\|B\|$	$\|C\|$	A Time (s)	B Time (s)	C Time (s)	A IGD	B IGD	C IGD
classic-1(0.3)	404	404	404	34.54	25.69	27.68	0	0	0
classic-1(0.5)	340	340	340	32.58	23.47	24.60	0	0	0
classic-1(0.7)	267	267	267	23.94	18.27	18.56	0	0	0
classic-2(0.3)	3938	3938	3938	49 685.10	47 983.96	49 022.28	0	0	0
classic-2(0.5)	3346	3346	3346	15 674.65	15 508.49	17 022.07	0	0	0
classic-2(0.7)	2492	2492	2492	2486.46	2334.24	2915.44	0	0	0
classic-3(0.3)	5318	5318	5318	1658.59	1578.95	2550.44	0	0	0
classic-3(0.5)	4164	4164	4164	1354.55	1071.45	1825.01	0	0	0
classic-3(0.7)	2631	2631	2631	1045.07	629.78	831.59	0	0	0
classic-4(0.3)	11 611	11 611	9411*	34 963.99	30 422.60	61 907.13*	0	0	416.53*
classic-4(0.5)	9422	9422	5932*	25 498.69	21 570.85	36 947.51*	0	0	1331.18*
classic-4(0.7)	6628	6628	6628	8699.78	5711.87	11 181.94	0	0	0
classic-5(0.3)	2429	2429	2429	311.66	233.99	284.22	0	0	0
classic-5(0.5)	1931	1931	1931	328.72	207.16	267.76	0	0	0
classic-5(0.7)	1228	1228	1228	262.77	129.89	154.54	0	0	0
Baan(0.3)	672	672	672	18.29	3.01	3.6	0	0	0
Baan(0.5)	462	462	462	18.52	2.60	2.98	0	0	0
Baan(0.7)	235	235	235	20.04	2.15	2.27	0	0	0
MSWord(0.3)	147	147	147	5.57	0.57	0.89	0	0	0
MSWord(0.5)	111	111	111	6.01	0.44	0.65	0	0	0
MSWord(0.7)	66	66	66	7.35	0.26	0.29	0	0	0
ReleasePlanner(0.3)	50	50	50	46.64	0.11	0.17	0	0	0
ReleasePlanner(0.5)	34	34	34	39.74	0.08	0.10	0	0	0
ReleasePlanner(0.7)	22	22	22	212.47	0.06	0.09	0	0	0

generations) and for smaller instances (Baan, MSWord, and ReleasePlanner) maximum evaluation is 50 000 (100 generations). The seeding probability is set to 2%, repair method is applied for every individual as it is created in population initialization and offspring generation. For each instance, NSGA-II would run 30 times, and the results in the following sections are the average results of 30 runs. The sampling size in tri-objective SolRep is 2000 for Baan, classic instances, and 200 for MSWord and ReleasePlanner.

For ϵ-constraint, CWMOIP, and I-EC, the step of updating bound for each iteration is set as 1.

5.2.3. Quality indicators

Besides elapsed time and number of non-dominated solutions for each method finds, we adopt 3 other indicators for showing the quality of each method on all datasets. **HyperVolume** (HV) is wildly used for multi-objective solution set metric [9]. **Inverted Generational Distance** (IGD) and **Spacing** (SP) [24] are also widely used multi-objective performance indicators.

1. **HyperVolume(HV)** HyperVolume shows a region which is dominated by a solution set. We implement it with jmetalpy, higher HV is preferred.
2. **Inverted Generational Distance(IGD)** Inverted Generational Distance measures how a solution set approaching the Pareto front. We also use jmetalpy for this purpose, lower IGD is preferred.
3. **Spacing (SP)** Spacing calculates the standard deviation of the shortest distances from each solution to its nearest solution. Lower SP value is preferred.

Table 6
Non-dominated solutions of bi-objective NRP with additional constraint, found by I-EC (A) and seeded NSGA-II (B).

NRP	$\|A\|$	$\|B\|$	$\|Pareto(A)\|$	$\|Pareto(B)\|$	A Time (s)	B Time (s)	A IGD	B IGD	A HV	B HV	A SP	B SP
classic-1(0.3)	404	173.4	404	42.33	25.69	89.73	0	12.4	1.31_{10^6}	98%	3.09	8.19
classic-1(0.5)	340	146.83	340	41.97	23.47	89.22	0	17.39	1.01_{10^6}	98%	4.85	9.02
classic-1(0.7)	267	106.53	267	44.1	18.27	86.38	0	16.11	7.42_{10^5}	98%	6.37	17.38
classic-2(0.3)	3938	361.9	3938	12.33	47 983.96	581.81	0	226.13	3.43_{10^7}	94%	2.00	12.87
classic-2(0.5)	3346	351.87	3346	12.6	15 508.49	575.28	0	158.56	2.99_{10^7}	96%	2.53	11.82
classic-2(0.7)	2492	315.63	2492	12.3	2334.24	566.55	0	199.94	2.33_{10^7}	95%	4.79	10.65
classic-3(0.3)	5318	370.43	5318	12.23	1578.95	601.57	0	360.17	5.18_{10^7}	93%	2.12	9.18
classic-3(0.5)	4164	360.03	4164	12.1	1071.45	595.24	0	167.73	4.18_{10^7}	96%	2.56	9.7
classic-3(0.7)	2631	318.27	2631	11.03	629.78	564.46	0	129.94	2.77_{10^7}	97%	5.33	11.09
classic-4(0.3)	11 611	378.73	11 611	9.23	30 422.60	2030.73	0	570.39	1.99_{10^8}	94%	1.62	14.11
classic-4(0.5)	9422	378.33	9422	10.3	21 570.85	1996.99	0	425.95	1.66_{10^8}	95%	1.77	13.38
classic-4(0.7)	6628	347.47	6628	10.83	5711.87	1847.17	0	342.86	1.12_{10^8}	95%	2.44	16.4
classic-5(0.3)	2429	386.63	2429	9.87	233.99	651.64	0	375.86	4.41_{10^7}	93%	13.54	17.69
classic-5(0.5)	1931	356.43	1931	11.17	207.16	641	0	519.36	3.30_{10^7}	90%	17.58	18.67
classic-5(0.7)	1228	311.7	1228	14.53	129.89	608.02	0	430.15	1.87_{10^7}	91%	25.53	14.17
Baan(0.3)	672	373.67	672	239.03	3.01	4.56	0	45.04	5.68_{10^7}	100%	36.76	36.74
Baan(0.5)	462	322.27	462	211.73	2.60	4.65	0	23.2	3.72_{10^7}	100%	45.40	40.91
Baan(0.7)	235	171.43	235	106.23	2.15	4.34	0	38.73	1.82_{10^7}	100%	85.17	49.36
MSWord(0.3)	147	143.87	147	140.3	0.56	7.23	0	0.1	1.36_{10^5}	100%	4.57	4.64
MSWord(0.5)	111	110.9	111	110.77	0.47	7.3	0	0	8.04_{10^4}	100%	4.90	4.9
MSWord(0.7)	66	66	66	65.93	0.19	7.32	0	0	3.63_{10^4}	100%	5.75	5.75
ReleasePlanner(0.3)	51	49.87	51	49.43	0.15	3.51	0	4.18	1.79_{10^6}	100%	350.58	352.73
ReleasePlanner(0.5)	34	34	34	34	0.12	3.48	0	0	9.34_{10^5}	100%	409.52	409.52
ReleasePlanner(0.7)	26	25.87	26	25.77	0.08	3.49	0	1.17	5.25_{10^5}	100%	425.78	426.46

Table 7
Non-dominated solutions of triple-objective NRP, found by SolRep (A) and seeded NSGA-II (B).

NRP	$\|A\|$	$\|B\|$	$\|A \cap (A \cup B)\|$	$\|B \cap (A \cup B)\|$	A Time (s)	B Time (s)	A IGD	B IGD	A HV	B HV	A SP	B SP
Baan	739	479.2	739	445.27	13.58	9.32	144.66	165.63	93%	$3.27_{10^{11}}$	47.66	37.55
classic-1	472	477.43	469.77	458.47	81.75	85.27	70.94	83.26	7.78_{10^8}	99%	1.58	6.54
classic-2	1906	491.7	1906	440.47	13 424.16	571.96	163.61	596.33	$8.07_{10^{10}}$	91%	1.62	26.7
classic-3	1781	492.57	1750.93	445.2	456.34	584.25	429.54	471.29	$2.40_{10^{11}}$	97%	1.99	30.63
classic-4	1844	491.77	1842.57	422.73	4701.53	1936.96	772.83	865.71	$2.00_{10^{12}}$	96%	3.29	67.19
classic-5	1745	493.53	1736.77	428.67	193.45	613.64	397.74	572.33	$2.23_{10^{11}}$	99%	4.87	39.01
Word	138	265.97	130.87	225.67	0.47	7.29	2.02	1.01	4.85_{10^7}	100%	4.13	4.08
ReleasePlanner	60	214.87	56	214.87	0.28	3.84	149.43	0.11	95%	1.75_{10^8}	315.28	313.91

5.3. Answer to RQ1

To answer this question, we implement all instances in bi-objective form. We compare results of ϵ-constraint (A), I-EC (B) and CWMOIP (C) in Table 3. Note that $|A|$ denotes for non-dominated solutions found by method A which is ϵ-constraint, $|B|$ and $|C|$ are defined in the same way. Because all three methods can solve the whole Pareto front, it is unnecessary to adopt all indicators in this case. Thus we only show IGD in Table 3.

From the table can we observe that $|A| = |B| = |C|$ and IGD of three methods on all bi-objective instances are zero. These results indicate solutions found by the three methods are all the same. Both I-EC and CWMOIP can solve the whole Pareto front as ϵ-constraint. When it comes to elapsed time, I-EC spends less time than ϵ-constraint on all instances except for realistic-m4. I-EC reduced 17.8% time on average on large instances (classic and realistic instances) and it is 87.0% on small instances, namely Baan, MSWord, and ReleasePlanner. However, CWMOIP requires even more time than ϵ-constraint on most instances except for classic-1, realistic-g1 to realistic-g4, Baan, MSWord, and ReleasePlanner. In large instances, the ratio of time reduction is −11.3% on average. A negative value denotes that it fails to accelerate whole Pareto front solving. For small instances it is 80.9%. These results are likely to be related to the complexity and size of a problem.

According to our experience, an important factor associated with time consumption is the complexity of the objective. For ϵ-constraint and I-EC, the objective is simply set as the first or the last objective in light of implementation, as CWMOIP uses a weighted reduced objective. For example in the classic-2 instance, there are 620 requirements and 500 stakeholders. It would be 620 or 500 decision variables in the objective in ϵ-constraint but it comes to 1120 in CWMOIP.

Also, we implement seeded NSGA-II for these instances for comparison, and results are shown in Table 4. $|A|$ means non-dominated solutions found by method A.

$|Pareto(A)|$ denotes the number of solutions found by method A that are on the Pareto front, $|Pareto(B)|$ is defined similarly. Smaller HV scores would be written in a normalized way, which is a percentage with respect to the larger one. From Table 4 can we observe that seeded NSGA-II finds all non-dominated solutions on ReleasePlanner. For classic and realistic instances, seeded NSGA-II finds 200 to 420 non-dominated solutions, and less than 15 solutions are exactly on the Pareto front except for classic-1. In smaller instances (classic-1, Baan, MSWord, and ReleasePlanner), seeded NSGA-II successfully solves more than 60 Pareto solutions and in MSWord and ReleasePlanner it nearly finds all of them, but it spends much more time than I-EC. These conclusions are also supported by the IGD and HV scores. The SP scores of the NSGA-II results are greater or equal to ones achieved by the I-EC, which means solutions found by I-EC are much evenly distributed than solutions found by NSGA-II.

So the answer to RQ1 is that I-EC can solve the whole Pareto front with less time spent by ϵ-constraint. It accelerates the process of this solving process in most instances. But CWMOIP can hardly accelerate Pareto front solving on large instances.

5.4. Answer to RQ2

To answer this problem, we implement classic datasets, Baan, MSWord and ReleasePlanner in bi-objective with additional constraint form. Urgency objective is converted to a constraint and $l \in \{0.3, 0.5, 0.7\}$ as mentioned in Section 2.4. We compare results of ϵ-constraint (A), I-EC (B) and CWMOIP (C) in Table 5.[6]

From Table 5, I-EC methods can solve the whole Pareto front. CWMOIP solves the whole Pareto front except for classic-4 (0.3) and classic-4 (0.5) due to our time limit. Results of these two instances are not involved in the following discussion. On large instances (classic datasets), I-EC accelerates the whole Pareto front solving by 21.9% compared with ϵ-constraint. And it is 93.0% on smaller instances (Baan, MSWord, and ReleasePlanner). For CMWOIP, they are 1.09% and 91.2% (excluding classic-4 (0.3) and classic-4 (0.5)).

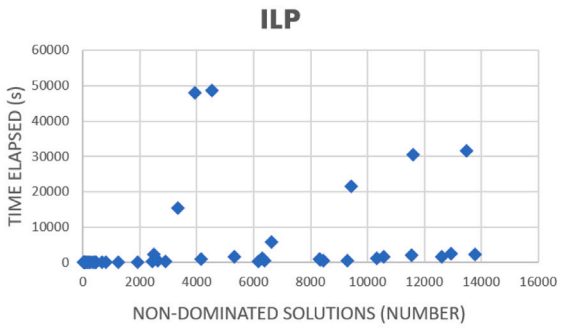
We also implement seeded NSGA-II for comparison and results are shown in Table 6. In small instances (MSWord and ReleasePlanner), seeded NSGA-II can solve the whole or almost the whole Pareto front but it spends much more time than I-EC does. In classic-1 and Baan with bounds 0.3, 0.5, and 0.7, I-EC also uses less time and finds more non-dominated solutions than seeded NSGA-II. IGD and HV indicators also corroborate these facts. In each instance with each bound factor, I-EC gains lower or equal SP scores than seeded NSGA-II, which denotes the much evenly distributed solutions sets.

The answer to RQ2 is that I-EC accelerates the whole Pareto front solving on bi-objective with an additional constraint.

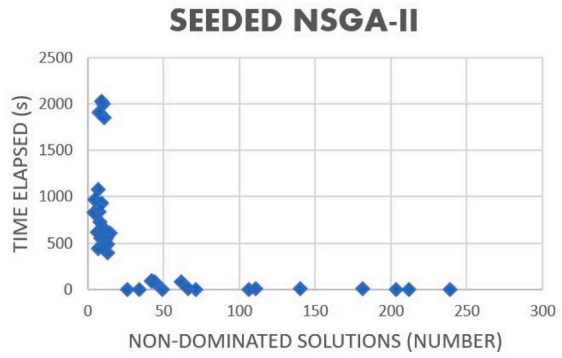
We can deduce from these results that in small instances (MSWord and ReleasePlanner), both I-EC and seeded NSGA-II behave well on finding solutions on the Pareto front. In these instances, solution space is rather small, less than 200 solutions. exact methods can find them within a second. For other instances, exact methods can also find all non-dominated solutions, but time increases as decision variables and constraints increase.

Fig. 6 are plotted to show the relationship between non-dominated solutions and elapsed time. Figs. 6a and 6b show it on all instances. As we mentioned in the former section, the elapsed time is associated with many factors in the practical study. Instances with many constraints and decision variables are hard to solve. Thus we remove results on classic-2 and classic-4 in Figs. 6c and 6d, and in Fig. 6c we add a dash line to indicate the trending. We can deduce from the figures that I-EC finds more non-dominated solutions with much time, meanwhile, it is not observed on seeded NSGA-II. When we investigate the quality of non-dominated solutions found by seeded NSGA-II (maybe not on Pareto front), IGD scores are mainly several hundred which shows its distance from Pareto front and SP values show most solutions sets are not as evenly distributed as those found by I-EC in Table 4, and most instances in Table 6 except for Baan and classic-5(0.7).

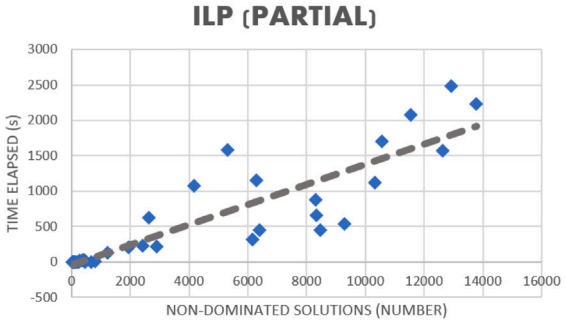
Due to the similarity between the formulations in RQ1 and RQ2, we can find out some commonalities from the results. I-EC could find all non-dominated solutions on both general bi-objective instance and achieve an acceleration compared with CWMOIP, both with or without the additional constraint. As compared with seeded NSGA-II in small instances (classic-1, Baan, MS Word, and ReleasePlanner), I-EC could find all non-dominated solutions in a short time. In other instances, the additional constraint makes seeded NSGA-II spends much time on finding non-dominated solutions, meanwhile, I-EC solves faster as the searching space is reduced by the additional constraint.



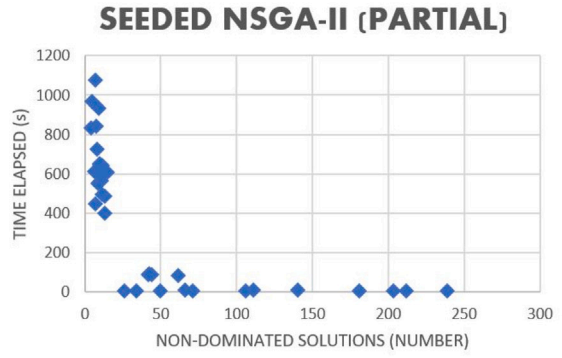
(a) I-EC on all instances



(b) Seeded NSGA-II on all instances



(c) I-EC on partial instances



(d) Seeded NSGA-II on partial instances

Fig. 6. Relation between non-dominated solutions and elapsed time.

[6] Restricted to memory limit, we run CWMOIP with a time limit of 1000 s for Cplex solver on classic-4 (0.3) and classic-4 (0.5), and it did not find the whole Pareto front which should be noticed. It already finds less non-dominated solutions in much time than other methods, thus it does not affect the conclusion.

5.5. Answer to RQ3

To answer this question, we implement tri-objective NRP with SolRep and seeded NSGA-II adopted. Results are shown in Table 7.

Notice that neither of the two methods can find the whole real Pareto front theoretically, but a "Pareto front" still can be made up from non-dominated solutions of the two methods. In this case, a non-dominated solution is on the Pareto front denotes we did not find any solutions to dominate it yet. Exact methods like ϵ-constraint can exam whether it is on the real Pareto front, but the expenditure of time is not afforded when searching space is considerably enlarged as another objective introduced. So in Table 7, we adopt $|A \cap (A \cup B)|$ to indicate how many non-dominated solutions are found by A.

As the third objective is introduced in Table 7, three exact methods can hardly solve the whole Pareto front, it would take hours for I-EC to solve the Pareto front even on the smallest synthetic instance classic-1. SolRep is introduced for finding non-dominated solutions subset. Note that all solutions found by SolRep are at least weakly dominated. As we explain the objective reducing mechanism in Fig. 4, SolRep does not adopt a weighted objective in consideration of efficiency. But we cannot prove whether a non-dominated solution found by seeded NSGA-II is on the real Pareto front as it is an approximate algorithm.

In smaller instances MSWord and ReleasePlanner, seeded NSGA-II find more non-dominated solutions than SolRep, indicated by the lower IGD score and the higher HV score. It also has a smaller SP score compared with SolRep. But SolRep spends less time even less than 0.5 s. For the Baan instance, SolRep finds more solutions that could not be dominated by the solutions by SolRep or NSGA-II, while NSGA-II achieves a better HV score and a better SP score. And for other instances (classic-1 to classic-5), SolRep successfully finds much more non-dominated solutions with better IGD, HV, and SP scores.

The answer to RQ3 is that for another more objective, exact algorithms aiming at finding all solutions fail to find enough solutions in a short time, meanwhile seeded NSGA-II shows its efficiency. SolRep is good at finding evenly distributed and much non-dominated solutions.

5.6. Threat to validity

Representative synthetic datasets. The urgency value is randomly generated with a uniform distribution in $[1, 9]$, which is not observed from the real world. It may be not representative or general. In RQ2, we construct the additional constraint with these values and let the urgency of the release plan be greater or equal than l multiple the sum of urgency value of all requirements (mentioned in Section 2.4). In RQ3, we use these urgency values as the coefficients of the third objective and apply the sampling-based ILP method SolRep and a genetic algorithm NSGA-II on it. Both of them do not be sensitive to what exactly the urgency value is of each requirement. Thus it may affect little to our study. To avoid this kind of problem, we should collect them from the real world in future works.

Problem instances timeliness. Datasets from [6] and [17] are mainly projects before 2010. Nowadays, the software is larger and much more complex than before. Correspondingly, their customers, stakeholders, requirements, and features might be quite far from 5000 requirements and stakeholders. For all projects of Mozilla, more than 10 000 new requirements(defects, enhancements) are still active.[7][8] For Firefox(no other platforms besides desktop version, only client) itself, it is still more than 10 000.[9] Relations for these requirements are more complex rather than logic-and, precedence, request, and so on. In the future, we should collect datasets from real-world projects and model the problem with more kinds of relations from practical usage.

Dedicated or generalized methods. We can hardly say what the "best" performance of evolutionary methods theoretically is due to their plenty of parameters. Though according to our study, the evolutionary algorithm NSGA-II with the seeding mechanism does not meet our exception on solving MONRP compared with exact methods. It is possible that for a certain NRP instance and certain modeling, some configurations are rather powerful since we cannot try every configuration. For future work, we may investigate more evolutionary algorithms and find a better way for seeding and repairing.

6. Related work

Since NRP is proposed in [3] there are many kinds of research discussing it and focusing on different aspects, such as extending the problem modeling to characterize real-world features, developing more efficient algorithms and so on.

Some works are focusing on the modeling of NRP. Zhang et al. [4] proposed MONRP and Pareto approach solving these problems. They adopted NSGA-II for MONRP and found it outperforms other weighted or Pareto approach algorithms (single-objective GA, Pareto GA, Random Search). In 2014, Harman et al. [25] developed a requirements sensitivity analysis for NRP and applied it with Nemhauser–Ullmann's (NU) algorithm for solving single-objective instances. Pitangueira et al. [18,26] model MONRP as risk-aware with three criteria, namely cost, revenue, and risk and they solved this problem with Z3, SMT and NSGA-II. Amaral et al. [27] model a risk-aware constrained bi-objective NRP, applies evolutionary algorithms on it. Mougouei and Powers [28] work on integrating value dependencies of requirements selection. The dependency could be positive, 0, or negative, where 0 denotes no dependency, negative and positive number denotes the quality and the strength of the explicit value dependency. They adopt a fuzzy-based optimization method for this problem.

For robust NRP, Paixao et al. [29] proposed scenario-based robust NRP formulation and solved it with simulated annealing and genetic algorithm. Li et al. [30] proposed the Monte-Carlo simulation robust MONRP dealing with uncertainty and developed two methods MCNRP-US (concerning NRP uncertainty size) and MCNRP-R (concerning NRP risk) for different aspects.

Aydemir et al. [31] proposed goal-oriented requirements engineering (GORE) to the next release problem and modeled the problem into SMT/OMT formulas. And for more objectives, Geng et al. [11] model many-objective NRP, up to five objectives and employs state-of-art genetic algorithms on those problems. In 2019, Etgar et al. [32] proposed the several-release-problem for f planning for the entire scope of product releases in the planning horizon. They introduced a clustering enhanced searching technique for solving this problem.

Many works tend to heuristic methods for solving NRP, such as evolutionary and genetic methods. In 2011, Durillo et al. [5] used NSGA-II, MOCell, and PAES for solving bi-objective NRP. And in 2013, Zhang et al. [7] improved NSGA-II with archive and repair method. Also, they proposed and, or, precedence, cost- and value-based constraints in MONRP modeling. Cai and Wei [33] employ a hybrid method of decomposition and domination-based evolutionary algorithm for multiple objectives NRP. Silva et al. [34] proposed a path relinking based method for generating initial population within the multi-objective genetic algorithm and found it outperformed the random initialized method on MoCell and NSGA-II.

Zhang et al. [35] proposed a two-phase external archive guided multi-objective evolutionary algorithm (2EAG-MOEA/D) for solving MONRP. The two-phase evolutionary process is composed of the convergence and diversity phase and it outperforms MOEA/D and NSGA-II. In 2016, Kumari et al. [36] proposed using quantum-inspired evolutionary algorithms namely QEMEA, QMDEA, and MQHDE for solving MONRP. Zhang et al. [19] investigated hyper- and meta-heuristic methods for solving MONRP and found hyper-heuristic NSGA-II outperformed others as a single method and suggested that it might be

[7] 10 000 is the maximum number for a single query in https://bugzilla.mozilla.org/.

[8] Query: Classification: Client Software, Developer Infrastructure, Components, Server Software, Other Status: NEW, till Dec 29 2020 PST as we queried.

[9] Query: Classification: Client Software Product: Firefox Resolution: —.

better for combining results found by other heuristic methods such as hyper-heuristic Hill Climbing and hyper-heuristic Simulated Annealing.

The ILP, anytime methods using exact solvers for MONRP are discussed in several works. In 2015, Veerapen et al. [9] start to use the exact method, namely ϵ-constraint, for capturing all non-dominated solutions, as the state-of-art search-based method. Domínguez-Ríos et al. [37] first use exact method in anytime methods. They could find enough supported and non-supported solutions for bi-objective NRP and be able to find the whole Pareto front in a sufficient time.

There are also many works on warm and colony optimization. In 2015, Botelho et al. [38] investigated Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) on three classic datasets. Hamdy and Mohamed [39] adopt a hybrid approach based on improved binary particle swarm optimization on bi-objective NRP. Zhang et al. [40] implement particle swarm optimization for finding optimal subset on multi-objective NRP. Balogun et al. [41] develop a hybrid of Ant Colony and Tabu Search for solving bi-objective NRP.

7. Conclusion

In our study, we formed NRP as a bi-objective optimization problem with maximizing revenue and minimizing cost. Further, we introduce another objective urgency as an additional constraint or the third objective. We use existing datasets and generate some synthetic datasets based on them before comparing algorithm performance on the large-scale problem. Several algorithms are applied on these MONRP instances, namely ϵ-constraint, I-EC, CWMOIP, SolRep, and NSGA-II as an evolutionary algorithm for comparison purposes. We found CWMOIP could hardly accelerate ϵ-constraint efficiency and I-EC behaves best on almost all datasets with two objectives. For tri-objective instances, seeded NSGA-II is good at finding non-dominated solutions in a short time. The quality of solutions found by SolRep is better on the large instance.

In conclusion, ILP is still an efficient method for generating non-dominated solutions. For more objectives, we may adopt other sampling methods to capture evenly distributed solutions on the Pareto front. Seeded NSGA-II shows its potentiality in solving smaller scale instances, other seeding mechanisms and repair methods evolutionary algorithm is worthy of further studying.

CRediT authorship contribution statement

Shi Dong: Software, Writing – original draft. **Yinxing Xue:** Supervision, Methodology, Writing – review & editing. **Sjaak Brinkkemper:** Data Curation, Writing - review & editing. **Yan-Fu Li:** Conceptualization, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No.61972373) and the Basic Research Program of Jiangsu Province (Grant No. BK20201192). Thank Prof. Nadarajen Veerapen for sharing source codes with us. Thank IBM for providing an academic license for using Cplex in our research. We appreciate a lot for Prof. Jifeng Xuan, and Dr. Muhammad Rezaul Karim sharing or publishing their datasets.

Appendix A. Revenue

Besides the case that revenue is associated with stakeholders, the revenue may be simply related to the requirements themselves. For classic and realistic instances, revenue is calculated with stakeholders. For Baan, MSWord, and ReleasePlanner they are associated requirements.

$$Min \quad \mathcal{F}_1(\vec{x}) = -\sum_{i=1}^{n} w_i x_i$$

$$Min \quad \mathcal{F}_2(\vec{x}) = \sum_{i=1}^{n} c_i x_i$$

$$s.t. \quad \mathcal{G}_1(\vec{x}) + \mathcal{G}_2(\vec{x}) + \mathcal{G}_3(\vec{x}) = 0$$

And for bi-objective with another constraint and tri-objective, they are defined in the same way. For convenience, we just address the way that classic and realistic instances are modeled in the discussion above.

Appendix B. Constraint

We inform that there are three kinds of constraints which are dependency, request, and coupling in our work. There is another kind of constraint we used for the dataset Baan. In this dataset, the cost of a requirement is calculated as the cooperation of multiple teams and each team works on several requirements. Thus there is a constraint that workload of each team is limited. Assume teams are $\{t_1, t_2, \ldots, t_r\}, t_j \in \{0, 1\}$ and capacity for team t_j is $Cap(t_j)$. Assume requirement r_i would cost $l_{i,1}t_1 + l_{i,2}t_2 + \cdots + l_{i,r}t_r$ where $l_{i,j}$ is the cost of team t_j working on r_i. So the additional constraints used by Baan are shown below.

$$\forall j \cdot \sum_{i=0}^{n} l_{i,j} t_i \leq Cap(t_j)$$

References

[1] A.M. Pitangueira, R.S.P. Maciel, M. Barros, Software requirements selection and prioritization using sbse approaches: A systematic review and mapping of the literature, J. Syst. Softw. 103 (2015) 267–280, http://dx.doi.org/10.1016/j.jss.2014.09.038, URL http://www.sciencedirect.com/science/article/pii/S0164121214002118.

[2] W.W. Sim, P.S. Brouse, Empowering requirements engineering activities with personas, Procedia Comput. Sci. 28 (2014) 237–246, http://dx.doi.org/10.1016/j.procs.2014.03.030, URL http://www.sciencedirect.com/science/article/pii/S1877050914000933. 2014 Conference on Systems Engineering Research.

[3] A.J. Bagnall, V.J. Rayward-Smith, I.M. Whittley, The next release problem, Inf. Softw. Technol. 43 (14) (2001) 883–890, http://dx.doi.org/10.1016/S0950-5849(01)00194-X.

[4] Y. Zhang, M. Harman, S.A. Mansouri, The multi-objective next release problem, in: Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference, 2007, pp. 1129–1137, http://dx.doi.org/10.1145/1276958.1277179.

[5] J.J. Durillo, Y. Zhang, E. Alba, M. Harman, A.J. Nebro, A study of the bi-objective next release problem, in: Empirical Software Engineering, Vol. 16, 2011, pp. 29–60, http://dx.doi.org/10.1007/s10664-010-9147-3.

[6] J. Xuan, H. Jiang, Z. Ren, Z. Luo, Solving the large scale next release problem with a backbone-based multilevel algorithm, IEEE Trans. Softw. Eng. 38 (5) (2012) 1195–1212, http://dx.doi.org/10.1109/TSE.2011.92.

[7] Y. Zhang, M. Harman, S.L. Lim, Empirical evaluation of search based requirements interaction management, Inf. Softw. Technol. 55 (1) (2013) 126–152, http://dx.doi.org/10.1016/j.infsof.2012.03.007.

[8] M.R. Karim, G. Ruhe, Bi-objective genetic search for release planning in support of themes, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNCS, vol. 8636, 2014, pp. 123–137, http://dx.doi.org/10.1007/978-3-319-09940-8_9.

[9] N. Veerapen, G. Ochoa, M. Harman, E.K. Burke, An integer linear programming approach to the single and bi-objective next release problem, Inf. Softw. Technol. 65 (2015) 1–13, http://dx.doi.org/10.1016/j.infsof.2015.03.008, URL http://www.sciencedirect.com/science/article/pii/S0950584915000658.

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[11] J. Geng, S. Ying, X. Jia, T. Zhang, X. Liu, L. Guo, J. Xuan, Supporting many-objective software requirements decision: An exploratory study on the next release problem, IEEE Access 6 (2018) 60547–60558.

[12] D. Olson, Matrix prioritization, 2014, URL http://www.bawiki.com/wiki/Matrix-Prioritization.html.

[13] M. Özlen, M. Azizoğlu, Multi-objective integer programming: A general approach for generating all non-dominated solutions, European J. Oper. Res. 199 (1) (2009) 25–35, http://dx.doi.org/10.1016/j.ejor.2008.10.023.

[14] Y. Xue, Multi-objective integer programming approaches for solving, in: 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), 2018, pp. 1231–1242.

[15] Y. Xue, Y.-F. Li, Multi-objective integer programming approaches for solving the multi-criteria test-suite minimization problem, ACM Trans. Softw. Eng. Methodol. 29 (3) (2020) 1–50, http://dx.doi.org/10.1145/3392031, URL https://dl.acm.org/doi/10.1145/3392031.

[16] A. Messac, C.A. Mattson, Normal constraint method with guarantee of even representation of complete Pareto frontier, AIAA J. 42 (10) (2004) 2101–2111, http://dx.doi.org/10.2514/1.8977.

[17] M.R. Karim, G. Ruhe, Bi-objective genetic search for release planning in support of themes, in: C. Le Goues, S. Yoo (Eds.), Search-Based Software Engineering, Springer International Publishing, Cham, 2014, pp. 123–137.

[18] A. Pitangueira, P. Tonella, A. Susi, R. Maciel, M. Barros, Minimizing the stakeholder dissatisfaction risk in requirement selection for next release planning, Inf. Softw. Technol. 87 (2017) 104–118, http://dx.doi.org/10.1016/j.infsof.2017.03.001, URL http://www.sciencedirect.com/science/article/pii/S0950584917301829.

[19] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, S. Brinkkemper, An empirical study of meta- and hyper-heuristic search for multi-objective release planning, ACM Trans. Softw. Eng. Methodol. 27 (1) (2018) 1–32, http://dx.doi.org/10.1145/3196831, URL https://dl.acm.org/doi/10.1145/3196831.

[20] A.J. Nebro, J.J. Durillo, M. Vergne, Redesigning the jmetal multi-objective optimization framework, in: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, in: GECCO Companion '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1093–1100, http://dx.doi.org/10.1145/2739482.2768462.

[21] I.I. Cplex, V12. 1: User's manual for CPLEX, Int. Bus. Mach. Corp. 46 (53) (2009) 157.

[22] T. Friedrich, M. Wagner, Seeding the initial population of multi-objective evolutionary algorithms: A computational study, Appl. Soft Comput. 33 (2015) 223–230, http://dx.doi.org/10.1016/j.asoc.2015.04.043, URL http://www.sciencedirect.com/science/article/pii/S1568494615002707.

[23] T. Chen, M. Li, X. Yao, On the effects of seeding strategies: A case for search-based multi-objective service composition, in: Proceedings of the Genetic and Evolutionary Computation Conference, in: GECCO '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1419–1426, http://dx.doi.org/10.1145/3205455.3205513.

[24] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, L. Salomon, Performance indicators in multiobjective optimization, European J. Oper. Res. 292 (2) (2021) 397–422, http://dx.doi.org/10.1016/j.ejor.2020.11.016, URL https://www.sciencedirect.com/science/article/pii/S0377221720309620.

[25] HarmanMark, KrinkeJens, Medina-BuloInmaculada, Palomo-LozanoFrancisco, Renjian, YooShin, Exact scalable sensitivity analysis for the next release problem, 2014, Undefined.

[26] A.M. Pitangueira, P. Tonella, A. Susi, R.S. Maciel, M. Barros, Risk-aware multi-stakeholder next release planning using multi-objective optimization, in: Lecture Notes In Computer Science (Including Subseries Lecture Notes In Artificial Intelligence And Lecture Notes In Bioinformatics), Vol. 9619, Springer Verlag, 2016, pp. 3–18, http://dx.doi.org/10.1007/978-3-319-30282-9_1.

[27] A. Amaral, G. Elias, A risk-driven multi-objective evolutionary approach for selecting software requirements, Evol. Intell. 12 (3) (2019) 421–444, http://dx.doi.org/10.1007/s12065-019-00237-2.

[28] D. Mougouei, D.M.W. Powers, Dependency-aware software release planning through mining user preferences, Soft Comput. 24 (15) (2020) 11673–11693, http://dx.doi.org/10.1007/s00500-019-04630-y.

[29] M. Paixão, J. Souza, A scenario-based robust model for the next release problem, in: GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference, 2013, pp. 1469–1476, http://dx.doi.org/10.1145/2463372.2463547.

[30] L. Li, M. Harman, E. Letier, Y. Zhang, Robust next release problem: Handling uncertainty during optimization, in: GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference, Association for Computing Machinery, 2014, pp. 1247–1254, http://dx.doi.org/10.1145/2576768.2598334.

[31] F.B. Aydemir, F. Dalpiaz, S. Brinkkemper, P. Giorgini, J. Mylopoulos, The next release problem revisited: A new avenue for goal models, in: Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 5–16, http://dx.doi.org/10.1109/RE.2018.00-56.

[32] R. Etgar, R. Gelbard, Y. Cohen, Presenting the several-release-problem and its cluster-based solution accelartion, Int. J. Prod. Res. 57 (14) (2019) 4413–4434, http://dx.doi.org/10.1080/00207543.2017.1404657.

[33] X. Cai, O. Wei, A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem, in: 2013 10th IEEE International Conference on Control and Automation (ICCA), 2013, pp. 412–417, http://dx.doi.org/10.1109/ICCA.2013.6565143.

[34] T.G.N. Da Silva, L.S. Rocha, J.E.B. Maia, An effective method for MOGAs initialization to solve the multi-objective next release problem, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8857, Springer Verlag, 2014, pp. 25–37, http://dx.doi.org/10.1007/978-3-319-13650-9_3.

[35] N. Zhang, Y. Huang, X. Cai, A two-phase external archive guided multiobjective evolutionary algorithm for the software next release problem, in: Communications in Computer and Information Science, Vol. 562, Springer Verlag, 2015, pp. 664–675, http://dx.doi.org/10.1007/978-3-662-49014-3_59.

[36] A.C. Kumari, K. Srinivas, Comparing the performance of quantum-inspired evolutionary algorithms for the solution of software requirements selection problem, Inf. Softw. Technol. 76 (2016) 31–64, http://dx.doi.org/10.1016/j.infsof.2016.04.010.

[37] M.A. Domínguez-Ríos, F. Chicano, E. Alba, I.M.D. Águila, J. Sagrado, Efficient anytime algorithms to solve the bi-objective next release problem, J. Syst. Softw. 156 (2019) 217–231.

[38] G. Botelho, A. Rocha, A. Britto, L. Silva, Investigating bioinspired strategies to solve large scale next release problem, 2015, Undefined.

[39] A. Hamdy, A. Mohamed, Greedy binary particle swarm optimization for multi-objective constrained next release problem, Int. J. Mach. Learn. Comput. 9 (5) (2019) 561–568.

[40] Y. Zhang, H. Li, R. Bu, C. Song, T. Li, Y. Kang, T. Chen, Fuzzy multi-objective requirements for NRP based on particle swarm optimization, in: X. Sun, J. Wang, E. Bertino (Eds.), Artificial Intelligence and Security, Springer International Publishing, Cham, 2020, pp. 143–155.

[41] B.A. Oluwagbemiga, S.J.A. Basri Shuib, G. Mariam, A.A. Thabeb, A Hybrid ant Colony Tabu Search Algorithm for Solving Next Release Problems.