# The Value of Exact Analysis in Requirements Selection

Lingbo Li, Mark Harman, Fan Wu, and Yuanyuan Zhang

**Abstract**—Uncertainty is characterised by incomplete understanding. It is inevitable in the early phase of requirements engineering, and can lead to unsound requirement decisions. Inappropriate requirement choices may result in products that fail to satisfy stakeholders' needs, and might cause loss of revenue. To overcome uncertainty, requirements engineering decision support needs uncertainty management. In this research, we develop a decision support framework *METRO* for the Next Release Problem (NRP) to manage algorithmic uncertainty and requirements uncertainty. An exact NRP solver (*NSGDP*) lies at the heart of *METRO*. *NSGDP*'s exactness eliminates interference caused by approximate existing NRP solvers. We apply *NSGDP* to three NRP instances, derived from a real world NRP instance, RALIC, and compare with NSGA-II, a widely-used approximate (inexact) technique. We find the randomness of NSGA-II results in decision makers missing up to 99.95 percent of the optimal solutions and obtaining up to 36.48 percent inexact requirement selection decisions. The chance of getting an inexact decision using existing approximate approaches is negatively correlated with the implementation cost of a requirement (Spearman $\rho$ up to $-0.72$). Compared to the inexact existing approach, *NSGDP* saves 15.21 percent lost revenue, on average, for the RALIC dataset.

**Index Terms**—Software engineering, exact multi-objective optimisation, simulation optimisation, next release problem

## 1 Introduction

Determining an appropriate subset of requirements to be delivered in the next release of a software system is a critical aspect in software engineering. In 1996, Karlsson developed an *Analytical Hierarchy Process* for supporting software requirements selection and prioritisation [1]. It was subsequently formulated as the Next Release Problem (NRP) by Bagnall et al. [2] in 2001. The NRP models stakeholders' objectives quantitatively, and employs optimisation techniques (i.e., meta-heuristic algorithms, dynamic programming) to identify a subset of requirements that is both feasible and well-suited to stakeholders' requirements. The NRP is a non-trivial problem, known to be *NP-hard* [2], [3], [4]. The search space of this problem increases exponentially with the number of requirements. Nevertheless, there are exact solutions to the classic NRP (which simply balances cost and value) that scale reasonably [4], [5], [6].

Unfortunately, such exisiting exact solution approaches fail to cater for uncertainty. Uncertainty is an inherent characteristic of software engineering [7]. The essence of uncertainty is the lack of complete knowledge at the time a decision must be made [8]. In software engineering, the requirements of a new system are incomplete before the users have started to use it [9], and may remain the subject of change and uncertainty thereinafter. Nevertheless, decision makers have to make decisions under such uncertainties. The uncertainties include uncertainty about development resource availability, the impact of dynamic and frequent changes in the overall software development life cycle, and the accuracy of the software project estimation. Underestimated or ignored uncertainties may elevate risk, and might even result in project failure [9].

Previous NRP work was concerned with point-based estimation. Point based esitmates are specific values (rather than intervals of "confidence"), estimated by human requirements engineers [2], [10], [11], [12]. In this previous work, the attributes of requirements and stakeholders are quantified as explicit values, and requirements uncertainty is either underestimated or completely overlooked [13]. For example, given a set of quantified requirements, although point-based estimation approaches can provide optimal solutions in terms of expected cost and revenue, they fail to offer an assessment of the confidence of such results. Thus, they may mislead the decision maker and amplify the consequences of risk.

The impact of uncertainty could be mitigated by performing mathematical measurement methods and scientific risk management methods. Sensitivity analysis is one such risk management method, and has been applied previously in search-based requirements engineering [4], [14]. In this previous work, sensitivity analysis was performed on solutions generated by an approximate algorithm to capture the sensitivities of requirements attributes and their effect on candidate solution uncertainty. However, sensitivity analysis can only provide information on the sensitivities of parameters. It does not generate robust solutions that are 'robust' in the sense that they can tolerate the changes in circumstances that uncertainty makes inevitable.

Instead of discovering the sensitivity characteristics of the problem, robust optimisation is an operational research framework that explores the solution space and takes

• *The authors are with the CREST, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom. E-mail: {lingbo.li, m.harman, fan.wu, yuanyuan.zhang}@cs.ucl.ac.uk.*

*Manuscript received 29 Jan. 2016; revised 18 July 2016; accepted 30 Sept. 2016. Date of publication 3 Oct. 2016; date of current version 20 June 2017. Recommended for acceptance by R. Lutz.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TSE.2016.2615100*

uncertainty into account simultaneously [15], [16]. Paixão et al. and Li et al. introduced concepts from the literature on robust optimisation to cater for the uncertainty in NRP requirement decisions [17], [18]. They reported robust NRP solutions, but did not guide decision makers to select solutions from thousands of candidates offered. Moreover, the algorithms they adopted are inexact, with a potential consequence of information loss due to suboptimal solutions. In order to aid decision support in the early stages of software engineering, we develop a decision support framework for Next Release Problem (called *METRO*), which utilises both a simulation-based robust optimisation approach and in parallel with an alternative traditional point-based optimisation approach. Our approach involves a novel exact algorithm combined with a Monte-Carlo Simulation (MCS) to deal with algorithm non-determinism and to capture requirements uncertainty. Our approach eliminates algorithmic uncertainty, and explicitly helps decision makers to understand and make the trade-off between uncertainty/risk and conventional objectives of cost & revenue.

To handle requirement uncertainty, *METRO* takes into account the quantified cost and revenue of requirements as well as the Probability Density Function (PDF) of uncertainties associated with these requirement attributes (cost and revenue). With the aid of a PDF of uncertainties, *METRO* uses MCS to simulate uncertainties in terms of their impact on specific objectives. A set of solutions is subsequently picked by *METRO*'s exact optimisation technique. *METRO* quantitatively analyses the outcomes of its own optimisation phase, and interprets the findings through a set of visualisations. These visualisations depict the tension between two different objectives in the presence of uncertainty, and illustrate the characteristics of requirements regarding the design space. This information allows decision makers to understand the impact of uncertainty and to determine each requirement's priority.

The paper's primary contribution is to introduce exact multi-objective dependence-respecting NRP solver to deal with algorithmic uncertainty and requirements uncertainty, although we also believe that *METRO*'s visualisations are a potentially useful secondary contribution. More specifically, the following contributions are made:

1) The first contribution is about eliminating algorithmic uncertainty. We develop an exact NRP optimisation solver: **N**on-dominated **S**orting Conflict **G**raph based **D**ynamic **P**rogramming algorithm (*NSGDP*) for our framework *METRO*. Our experimental studies reveal that, with the aid of *NSGDP*, the decision maker can avoid information loss (without which we show that, for an example real world requirements problem, he or she will lose up to 99.95 percent of the optimal solutions and will make up to 36.48 percent inexact requirement selection decisions as a result). Furthermore, for the RALIC study, the execution time of *NSGDP* is better than NSGA-II which is currently the best performing NRP solver according to a recent empirical study [19]. However, this is only a result from a single set of requirements, and so we cannot claim overall superior performance, based on this single study.
2) The second contribution is our introduction of an approach to cater for requirements uncertainty. *METRO* investigates the difference between the optimal-yet-risky solutions and robust-yet-suboptimal solutions. Two indicators are used: expected risk premium and risk reduction. Our experimental results show that, for RALIC NRP instances, developing a software project based on an optimal-yet-risky release plan rather than robust-yet-suboptimal release plan, increases by 10 percent the probability of overrunning (more than 150 percent budget) while gaining less than 0.39 utility in return "the expected risk premium".
3) The third contribution is that the proposed framework can better support decision makers in understanding the requirements selection problem. A series of quantitative techniques is provided for highlighting the characteristics of requirements and solutions. The difference of requirement selection probability between two NRP approaches is analysed and presented in a stacked bar plot. We found that risk-aware simulation-based NRP model (*simulation-NRP*) is more likely to include requirements with low uncertainty than point-based estimated NRP model (*point-NRP*) does.

The structure of the rest of the paper is organised as follows: Section 2 briefly describes the background of this paper, including the problem statement of NRP in general, the *simulation-NRP*, Requirements Interaction Management: Conflict Graph, and core NRP solver: Nemhauser-Ullmann Algorithm. Section 3 formally defines the NRP decision analysis framework which implements our approach. Section 4 presents the research questions, and answers these research questions by applying the proposed framework on three synthetic NRP instances. Section 5 evaluates the threats to validity for our framework, and Section 6 discusses the related work in which our work is located. Section 7 concludes the paper and suggests future work.

## 2 Background

Before introducing our NRP decision analysis framework, we first describe the problem statement of the NRP in general, as well as the improved NRP model: simulation-based NRP. Subsequently, a Requirements Interaction Management model: Conflict Graph is presented to construct requirement dependencies, followed by introducing Nemhauser-Ullmann Algorithm, which is the NRP solver used in our framework.

### 2.1 Next Release Problem Statement

In this section, we depict the problem statement of classic NRP (we use a standard formulation of the problem). In requirements analysis and optimisation, deciding which requirements should be included in the next release of a software system is critical for the software project. In the context of Search-based Software Engineering (SBSE), the term NRP was suggested by Bagnall et al. [2] in 2001. The aim of NRP is to search the feasible and (near) ideal combinations of requirements to balance the requests from different stakeholders, and the constraints, by applying various meta-

heuristic algorithms. The set of stakeholders is denoted by Eq. (1) and the set of possible requirements is denoted as

$$C = \{c_1, \ldots, c_m\} \tag{1}$$

$$R = \{r_1, \ldots, r_n\}, \tag{2}$$

where $m$ is the number of stakeholders, and $n$ is the number of requirements.

During software development, some resources (e.g., human resources and facility resources) need to be allocated to satisfy each requirement. In the context of cost-value based NRP, cost is used to measure the amount of resource needed to fulfil the requirements as given as

$$Cost = \{cost_1, \ldots, cost_n\}. \tag{3}$$

There is a weight vector that reflects the degree of importance of each stakeholder for the company. The relative weight vector related to each stakeholder $c$ $(1 \leq j \leq m)$ is denoted as

$$Weight = \{w_1, \ldots, w_m\}, \tag{4}$$

subject to: $w_j \in [0, 1]$, and $\sum_{j=1}^{m} w_j = 1$.

It is assumed that the importance of each requirement for each stakeholder is different. Given a stakeholder, the level of satisfaction of this stakeholder is based on the requirements that are satisfied in the next release of the software system. Based on this assumption, each requirement $r_i$ $(1 \leq i \leq n)$ is assigned a *value* $(r_i, c_j)$ by each stakeholder $c_j$ $(1 \leq j \leq m)$. The overall revenue of a given requirement $r_i$ $(1 \leq i \leq n)$ for the company is denoted as

$$Revenue_i = \sum_{j=1}^{m} (w_j \cdot value(r_i, c_j)). \tag{5}$$

In NRP, the solution is presented as a decision vector $\vec{x} = \{x_1, \cdots, x_n\} \in \{0, 1\}^n$ to determine the requirements that are to be selected in the next release. In this vector, $x_i$ is 1 if requirement $i$ is selected and 0 otherwise.

Then, a single objective NRP, which intends to maximise commercial profits within a limited cost budget, is formulated as

$$Maximise\ Objective(\vec{x}) = \sum_{i=1}^{n} (x_i \cdot Revenue_i) \tag{6}$$

$$Subject\ to\ Constraint(\vec{x}) = \sum_{i=1}^{n} (x_i \cdot Cost_i) \leq b, \tag{7}$$

where $b$ is the project budget.

To allow the decision makers to understand the trade-off between two conflicting objectives, the Multi-Objective version of NRP (MONRP) treats the budget as another objective for optimisation instead of a constraint [12]. Thus, the formulation of MONRP can be represented as follows:

$$Minimise\ Cost(\vec{x}) = \sum_{i=1}^{n} (x_i \cdot Cost_i) \tag{8}$$

$$Maximise\ Revenue(\vec{x}) = \sum_{i=1}^{n} (x_i \cdot Revenue_i). \tag{9}$$

### 2.2 Simulation-Based Next Release Problem Statement

The traditional formulation of NRP computes the model parameters and objectives using point-based estimated values (exact numbers). As a result traditional NRP models (abbreviated to *point-NRP* hereinafter) overlook any concealed uncertainties under the expected parameter values. To overcome this limitation, *simulation-NRP*, a robust NRP model, is suggested by Li et al. [18] in 2014.

In *simulation-NRP*, the uncertainties concerning the model parameters are extracted as probability distributions. Besides, the solutions on Pareto-front contain, not only the expected quality of objectives, but also the probability of achieving the quality. This approach offers a probabilistic point of view for decision makers, and allows decision makers to flexibly balance the trade-off between the quality of software release plan and its probabilistic robustness.

Before performing *simulation-NRP*, one should ensure that human domain experts have elicited probability distribution of model parameters by a prior risk analysis. Once the probability distribution of parameters is determined, *simulation-NRP* utilises Monte-Carlo Simulation [20] to sample a large number of simulation scenarios of model parameters by their underlying probability distribution. The requirements scenarios are presented as a matrix $S$ as

$$S = \begin{pmatrix} (R,C)_{1,1} & (R,C)_{2,1} & \ldots & (R,C)_{n,1} \\ (R,C)_{1,2} & (R,C)_{2,2} & \ldots & (R,C)_{n,2} \\ \ldots & \ldots & \ldots & \ldots \\ (R,C)_{1,p} & (R,C)_{2,p} & \ldots & (R,C)_{n,p} \end{pmatrix}, \tag{10}$$

where $n$ is the number of requirements, and $p$ is the number of scenarios. The tuple $(R,C)_{i,k}$ denotes the revenue $R$ and cost $C$ of the $i$th $(1 \leq i \leq n)$ requirement in the $k$th scenarios $(1 \leq k \leq p)$.

Then according to the generated simulated scenarios, decision makers can use *simulation-NRP* to estimate the expected value of objectives as well as other interesting measures, such as the expected revenue of alternative (Eq. (11)), the expected cost of alternatives (Eq. (12)), and probability of budget overrun as

$$Exp\_Revenue(\vec{x}) = \sum_{i=1}^{n} \left( x_i \cdot \frac{\sum_{k=1}^{p} Revenue_{i,k}}{p} \right) \tag{11}$$

$$Exp\_Cost(\vec{x}) = \sum_{i=1}^{n} \left( x_i \cdot \frac{\sum_{k=1}^{p} Cost_{i,k}}{p} \right) \tag{12}$$

$$Risk(\vec{x}) = \mathbb{P}(actual\_cost(\vec{x}) > \theta \cdot Exp\_Cost(\vec{x})), \tag{13}$$

where $p$ is the number of scenarios, $\theta$ is the extent of budget overrun assigned by the decision maker (e.g., $\theta = 150$ percent), and $\mathbb{P}$ measures probability.

Drawing support from such probabilistic objective formulations, *simulation-NRP* can simply compute arbitrarily complicated parameter probability distributions and explore the feasible solutions in terms of these intuitive objectives as

$$Objective_1(\vec{x}) = Minimise\ Exp\_Cost(\vec{x}) \tag{14}$$

$$Objective_2(\vec{x}) = Maximise\ Exp\_Revenue(\vec{x}) \tag{15}$$

$$Objective_3(\vec{x}) = Minimise\ Risk(\vec{x}). \quad (16)$$

### 2.3 Next Release Problem with Conflict Graphs

In practice, there may be different constraints between the requirements in NRP. These constraints describe the relationships between the various requirements [21]. Mutual exclusion is a typical constraint, which denotes at most one of the two mutually exclusive requirements can be selected simultaneously. In graph theory, conflict graphs are usually used to construct such logical relations between objects. More precisely, a conflict graph $G$ contains a set of vertices and edges between two vertices as

$$\begin{aligned} &G = (V, E) \\ &V = \{v_i\} \\ &E = \{(v_i, v_j) | \text{The } v_i \text{ and } v_j \text{ is mutually exclusive}\}, \end{aligned} \quad (17)$$

where $V$ is the vertex set, in which each vertex represents a distinct object, and $E$ is the edge set, in which each edge means two connected vertices exclude each other (thus cannot be selected at the same time). The isolated vertices denote that those vertices can be selected with every other isolated vertex at the same time.

Conflict graphs have been successfully applied to Knapsack-like Problems with Conflicts [22], [23], [24], which are strongly *NP-hard* in general. Moreover, in 2009, Pferschy and Schauer [25] proved that forming Knapsack-like Problem with Conflict Graph (KCG) in the search tree can carry forward fully polynomial time approximation schemes (FPTAS). Accordingly, it is promising to model NRP in the form of the conflict graph, and then reconstruct it to search tree. We would interpret how to construct NRP with conflict graph to the search tree exemplified by the general knapsack-like problem.

To reconstruct a knapsack-like problem from a conflict graph data structure to a search tree data structure, the first step is processing $G$ in depth-first-search. Then picking a constrained vertex $v_i$ (conflicting with vertex $v_j$) to distinguish the problem into two sub-problems from top-down:

- Necessarily including $v_i$ in the sub-problem, and excluding $v_j$
- Always excluding $v_i$ in the sub-problem, and keeping the decision concerning $v_j$ open.

Mathematically, the process of constructing problem tree is presented as follows.

**Definition 1.** *$G \setminus v$ means subtracting a vertex $v \in V$ from graph $G$: $G \setminus v = (V', E')$, where $V' = V - \{v\}$ and $E' = \{(v_i, v_j) \,|\, (v_i, v_j) \in E, v_i \in V', v_j \in V'\}$.*

**Definition 2.** *For graph $G = (V, E)$ and a vertex $v \in V$, $C(v)$ represents a set of objects including $v$ and those have constraints with $v$: $C(v) = \{u \in V \,|\, u = v \text{ or } (u, v) \in E\}$.*

When all leaves of the root problem tree have no edge at all ($|E| = 0$), the problem is solved bottom up. The procedure of solving KCG is described in Algorithm 1. In Algorithm 1, if $G$ has no constraints at all ($|E| = 0$), then solve the problem using dynamic programming, otherwise the problem $G$ is divided into two sub-problems $G \setminus v$ and $G \setminus C(v)$ with respect to a chosen constraint $v$. The former one assumes $v$ is not selected in all of the solutions and the latter one assumes $v$ selected, thus all the objects that conflict with it cannot be selected in the final solutions and are removed from the problem as well ($C(v)$ contains the objects that have connections with $v$). After these two sub-problems are solved recursively, the algorithm sets $x_v = 1$ in all of the Pareto solutions for the second sub-problem, since $v$ is assumed to be selected in the second sub-problem. At last the algorithm merges these two sets of non-dominated solutions together and removes those being dominated to form the Pareto solution set for the problem $G$.

**Algorithm 1.** Solve KCG $S = Solve(G)$

**Require:** conflict graph $G = (V, E)$
  **if** $E = \emptyset$ **then**
    **return** $KnapsackProblemSolver(G)$
  **end if**
  Pick $v \in V$ that has an edge in $E$
  $S_0 = Solve(G \setminus v)$
  $S_1 = Solve(G \setminus C(v))$
  **for all** $\vec{s} \in S_1$ **do**
    set $v$ selected: $s_v = 1$
  **end for**
  **return** $S = Merge(S_0, S_1)$

### 2.4 Nemhauser-Ullmann Algorithm

To solve NRP exactly, we build an exact NRP solver *NSGDP* using the Nemhauser-Ullmann algorithm to solve specific instances in a decision tree solution space. The Nemhauser-Ullmann algorithm is a dynamic programming algorithm proposed by Nemhauser and Ullmann in 1969 [26]. It is a non-dominated sorting based multi-objective exact optimisation algorithm on enumerating the Pareto set of knapsack-like problems [27]. However, it has an obvious drawback. It cannot deal with knapsack-like problems with constraints. Harman et al. [4] employed it to materialise an exact NRP solver that focuses on NRP with the independent requirement.

For a given NRP problem with $n$ requirements, the Nemhauser-Ullmann algorithm starts with considering 0 requirements, and then iteratively inserts the next requirement $i$ into the every solution in the Pareto-front $P(i-1)$, where $P(i-1)$ denotes the Pareto-front of first $i-1$ requirements. After merging two solutions to set $P'(i) = P(i-1) \cup (P(i-1) + i)$, the Nemhauser-Ullmann algorithm uses non-dominated sorting (the so-called *staircase* function) to compute the Pareto-front of first $i$ requirements $P(i) = \text{Non-dominated-Sorting}(P'(i))$. $P(i-1) + i$ denotes the set of solutions that is obtained by setting the $i$th requirement to be selected for all solutions from $P(i-1)$. Following these steps, the final result $P(n)$ is computed inductively. Summarising, the Nemhauser-Ullmann algorithm is formalised by Algorithm 2:

**Algorithm 2.** Nemhauser-Ullmann Algorithm for NRP

**Require:** A set of $n$ requirements
  **for** $i = 1, \ldots, n$ **do**
    $P'(i) = P(i-1) \cup (P(i-1) + i)$
    $P(i) = \text{Non-dominated-Sorting}(P'(i))$
  **end for**
  **return** $P(n)$

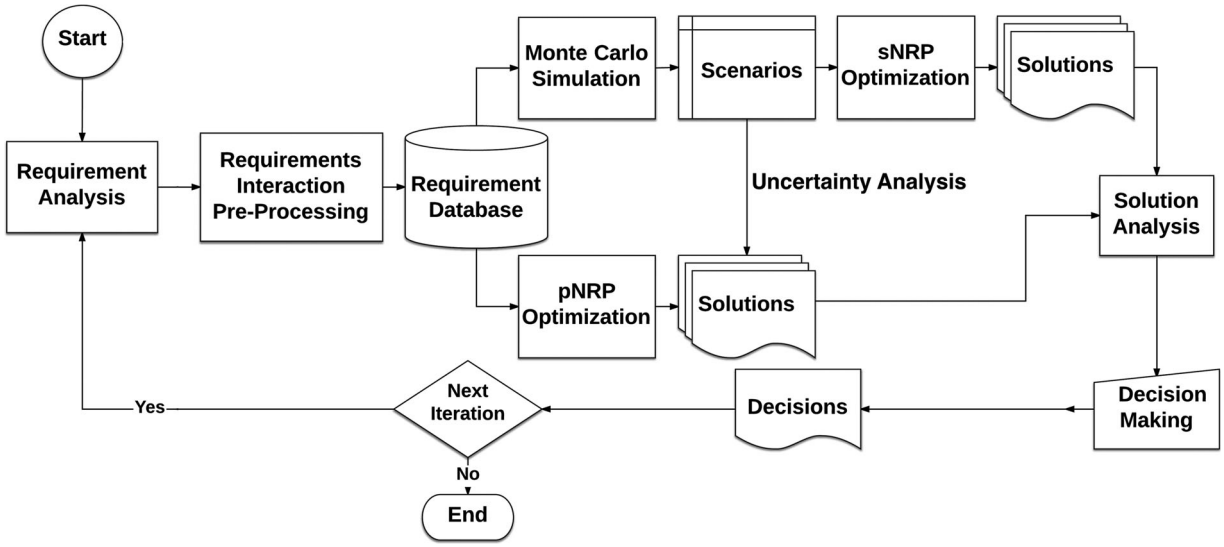



Fig. 1. NRP decision analysis framework: *METRO*.

## 3 Simulation NRP Decision Analysis Framework METRO

Multi-Objective NRP approaches produce a Pareto-front which may contain a large number of solutions. It is laborious for engineers to understand and identify one solution from thousands of candidate solutions, especially taking uncertainty into account. To aid decision makers to tackle the latent information within optimal solutions, this paper proposes a simulation NRP decision analysis framework, *METRO*. Instead of merely generating the optimal solutions themselves, *METRO* statistically analyses the optimal solutions, mines information from them, and provides the insight of these solutions. The main processes of *METRO* (Fig. 1) are:

1) Pre-processing the requirements dependencies.
2) Adopting *simulation-NRP* and *point-NRP* to model the requirements analysis problem separately, and then using exact optimisation solver (*NSGDP*) to produce the optimal solutions.
3) Statistically analysing results of two solutions, and visualising the refined information as well as the implicit requirement pattern for decision processes.
4) Performing this analysis in the next iteration.

### 3.1 Requirements Interaction Pre-Processing

Requirements may depend on each other [28]. Some requirements may interact with other requirements due to the constraints or limitations that come from techniques, or business related issues. Requirement implementations may be mutually exclusive, or should be fulfilled together on the basis of their interactions. Failure to consider requirement interactions, may yield infeasible decisions.

Requirements Interaction Management has been proposed to analyse and manage the dependences among requirements [21], [29]. In NRP, Requirements Interaction Management involves at least two types of interactions (*And*, and *Or*). The *And* dependence between two requirements means the selections of requirements have to be in the same release. On the other hand, the selection of two requirements which have *Or* dependence is "repelling" each other because these two requirements are mutually exclusive. Table 1 presents the mathematical expressions of these interactions.

Although the original Requirements Interaction Management defines the dependencies between requirements, Requirements Interaction Management can be simplified to enable fast execution and better convergence. In our proposed approach, the *And* dependence satisfies $\forall(i,j) \in \xi, \quad x_i = x_j$. By transitivity, if $(i,j) \in \xi$ and $(j,k) \in \xi$, then $x_i = x_j = x_k$. Therefore, a super-requirement $Req_{i,j,k}$ can be used to represent requirement $i$, $j$, and $k$ in a single decision variable. This simplification, reduces the computational cost for requirements constraint handling and the search space within which we seek solutions.

### 3.2 Exact NRP Optimisation Solver

After requirement data pre-processing, decision makers have to decide which requirements are critical and should be included in the next release of system under budget constraints. For this step, the objectives and formulations should be clearly defined. The conventional criteria for NRP are maximising the expected revenue and minimising the expected release cost. Decision makers can also define other criteria, such as the satisfaction degree of customers, the fairness level among different stakeholders [10], and the utility of release.

Taking uncertainty into account, project risk could be an extra objective to optimise. In a software project, the project risk is related to future events that may have undesired

TABLE 1
Requirement Interactions

| | |
|---|---|
| ***And*** | $\forall(i,j) \in \xi, x_i = x_j$ |
| **Or** | $\forall(i,j) \in \varphi, x_i \wedge x_j = 0$ |

*The sets $\xi$, and $\varphi$ present the interaction types* And, *and* Or, *respectively. The set $\xi \cap \varphi = \emptyset$.*

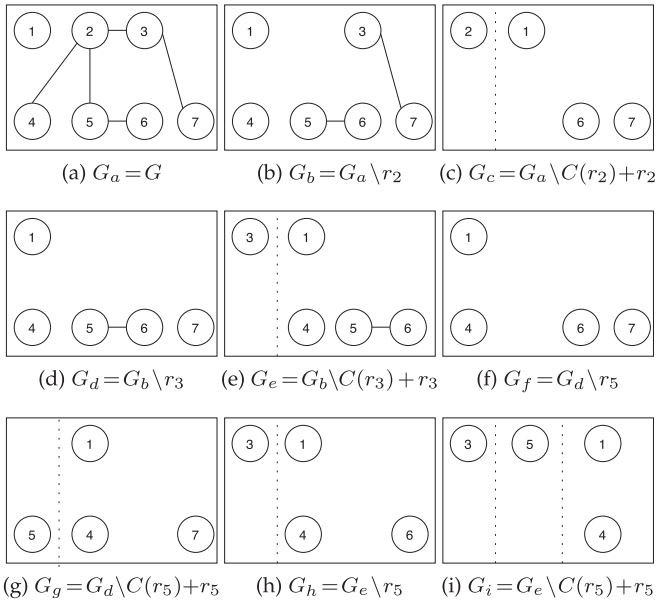
consequences for the project [30]. Project risk could include budget overrun, the number of requirements becoming inflated, departure of a key person, and productivity failing to meet expected estimates [31], [32]. There are existing risk analysis methods that identify and elicit these software project uncertainties quantitatively and use probability distributions to represent the uncertainty [33]. After these uncertainties have been elicited, our framework formulates the fitness function to optimise project risk.

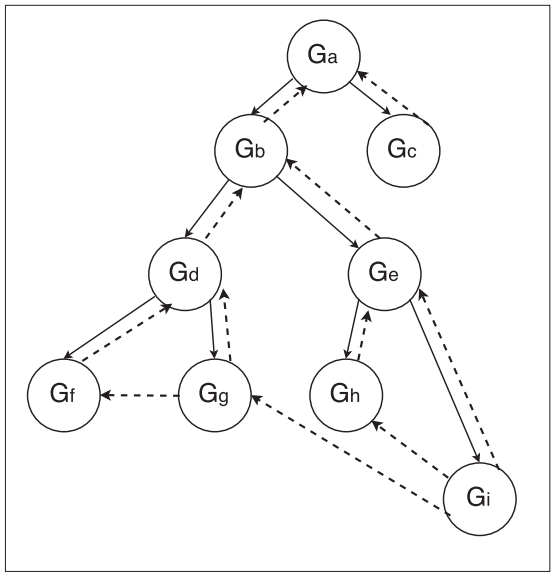
In NRP, such multi-objective decision support problems can be investigated using a multi-objective optimisation algorithm. In order to ensure that the variations in results do not come from the stochastic nature of the algorithm, we design an exact NRP optimisation solver *NSGDP*. The *NSGDP* uses the Nemhauser-Ullmann algorithm, an exact dynamic programming algorithm, as the core NRP solver, and augmented by Conflict Graph to deal with the requirements interaction. First, the NRP problem with constraints is modelled into Conflict Graph. Then, the root problem is broken down into sub-problems according to Algorithm 1 until there is no constraint in sub-problems. Lastly, Algorithm 2 is used to solve NRP without constraint directly It is worth mentioning that, our algorithm is applicable to, not only the case we study in this paper, but also any kind of knapsack-like problems with exclusive conditions.

To further improve the performance of our algorithm, we introduce an array to store the processing order. This is because, when a graph $G$ is divided to two graphs $G \setminus v = (V_0, E_0)$ and $G \setminus C(v) = (V_1, E_1)$, $G \setminus C(v)$ is a sub-graph of $G \setminus v$ ($V_0 \supset V_1$ and $E_0 \supset E_1$). If further divided, the 'offspring'(s) of $G \setminus v$ may be exactly the same as $G \setminus C(v)$, thus does not need to be solved multiple times.

There is no strict rule of which $v$ should be chosen as long as it has at least one constraint on it. In our algorithm, we always choose the vertex $v$ with the biggest degree (has the biggest number of edges connecting it), thus the number of edges in $G_1$ is minimised to have a minimal depth of subsequent dividing.

Fig. 2 illustrates the breakdown process of our *NSGDP* algorithm with a simple problem instance (Fig. 2a). There are seven requirements and five conflicting interactions in this instance. The edge connects two requirements means these two requirements are conflicting with each other. So the expressions of this instance is $V = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$ and $E = \{(r_2, r_3), (r_2, r_4), (r_2, r_5), (r_5, r_6), (r_3, r_7)\}$. For this instance, the problem is divided based on requirement $r_2$ first. The reason is that $r_2$ conflicts with most requirements ($r_3$, $r_4$, and $r_5$), so its degree is biggest (degree 3). Then the problem is broke down into two sub-problem. The $r_2$ is not selected in the first sub-problem $G_b = G_a \setminus r_2$ (Fig. 2b), so $r_2$ and the edges connected to $r_2$ are removed. In the second sub-problem $G_c = G_a \setminus Ext(r_2)$ (Fig. 2c), $req_2$ is selected. Accordingly, the requirements have connection with $r_2$ are removed. The dashed line in Fig. 2c denotes that, in order to solve the problem $G_c = G_a \setminus Ext(r_2)$, *NSGDP* first solves the right part, and then computes the optimal frontier of whole problem $G_c$ by merging the consideration of left part requirements.




(j) The breakdown procedure (solid line) generated by *NSGDP* algorithm, and the solving procedure (dashed line) for problem $G_a$

Fig. 2. The illustration of the subdivision process of the *NSGDP* for an instance with 7 requirements and 5 conflicting interactions. Figs. 2a to 2i are each generated sub-problem in the subdivision phase. Fig. 2j illustrates the generated sub-problems and the solution path of *NSGDP* algorithm.

Subsequently, *NSGDP* further divides these two generated sub-problems. Because there is no edge in $G_c$ (Fig. 2c), no further breakdown would be performed on $G_c$. Since there are two conflicts in $G_b$ ($E_1 = \{(r_3, r_7), (r_5, r_6)\}$), and each conflicted requirement has same degree (degree 1), *NSGDP* picks $r_3$ by requirement id order. Thus, sub-problem $G_b$ is divided into sub-problems $G_d = G_b \setminus r_3$ (Fig. 2b) and $G_e = G_b \setminus Ext(r_3)$ (Fig. 2e). *NSGDP* continues to breakdown the problem until there is no further conflict that can be subdivided. In this instance, there are five leaf node sub-problems generated.

After the breakdown process is terminated, the *NSGDP* solves the problem from the bottom up. Fig. 2j illustrates the procedure by a dashed line. According to the composition of problems $G_i$, $G_h$, and $G_g$, the algorithm solves the $G_i$

first. Then the results of $G_i$ can be used for solving the other two leaf nodes sub-problem $G_h$ and $G_g$. Thus, the re-computation can be avoided by storing previous steps' results.

### 3.3 Results Analysis & visualisation

The last step of *METRO* is to analyse the solutions on two Pareto-fronts, one of which is produced by *point-NRP*, and the other by *simulation-NRP*. The shape of generated Pareto-frontier exposes the possible trade-off among all conflicting objectives.

The shape of Pareto-front helps decision makers to understand the possible trade-off among all conflicting objectives, yet it does not provide other intelligible information to interpret the variations among the solutions as well as the characteristics of requirements. In particular, the number of solutions on Pareto-front maybe large, thereby requiring further analysis support to help the decision maker understand the implication for requirement release decisions. By contrast, *METRO* performs a series of posterior analysis procedures to help decision makers to concentrate on the impacts of requirements uncertainty, most interesting solutions, and most urgent and worthwhile requirements.

**Algorithm 3.** Generate Solution Compare Pairs

**Require:** *simulation-NRP* solution set $S_1$, and *point-NRP* $S_2$.
  set $Pairs = \emptyset$
  **for all** $\vec{s_1} \in S_1$ **do**
    $\vec{s} = S_2[0]$
    **for all** $\vec{s_2} \in S_2 \wedge Cost(\vec{s_2}) \geq Cost(\vec{s_1})$ **do**
      **if** $Cost(\vec{s_2}) < Cost(\vec{s})$ **then**
        $\vec{s} = \vec{s_2}$
      **end if**
    **end for**
    $Pairs = Pairs \cup Pair(\vec{s_1}, \vec{s})$
  **end for**
**return** $Pairs$

In order to assess the impact of requirements uncertainty, we introduced the concept of the *expected risk premium*, which is a variant of the *risk premium* [34]. This measures the difference between robust-yet-suboptimal solutions and optimal-yet-risky solutions. The robust-yet-suboptimal solution is simply that which has the lowest uncertainty variance in the distribution of possible values. Suppose we use the point based method to find a particular optimal-yet-risky solution (a set of requirements), $\vec{a}$, with given cost, $cost(\vec{a})$ and value, $value(\vec{a})$. We can find the robust-yet-suboptimal solution, $\vec{r}$ with cost $cost(\vec{r})$ closest to $cost(\vec{a})$ that does not exceed $cost(\vec{a})$. This is the greatest lower bound, on robust-yet-suboptimal solutions, bounded by the cost of $\vec{a}$. Because the robust-yet-suboptimal solution takes account of uncertainty, it has a range of possible values, of which the expected value, $value(\vec{r})$, is simply the most probable. The *expected risk premium* is simply the difference between $(value(\vec{a}) - cost(\vec{a}))$ and $(value(\vec{r}) - cost(\vec{r}))$. It is an 'expected' assessment of the return that will be lost by maximally reducing uncertainty. It is thus a way of understanding the penalty that is paid for reducing uncertainty in terms of reduced expected return.

To compute the *expected risk premium*, a *solution compare pair* which contains an optimal-yet-risky solution and a robust-yet-suboptimal solution should be determined first.

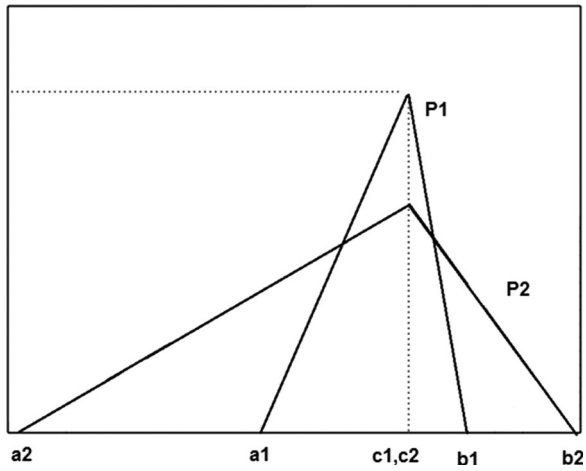



Fig. 3. Illustration of the triangle probability distribution [41]. $c_1$ and $c_2$ are the mode (expected) value of probability distribution $P_1$ and probability distribution $P_2$, respectively. $a_1$ and $b_1$ is the lowest value and highest of $P_1$ respectively while $a_2$ and $b_2$ is the lowest value and highest of $P_2$.

For each robust-yet-suboptimal solution $\vec{r}$, the optimal-yet-risky solution from *point-NRP*, which has the closest cost to $\vec{r}$ and the cost is not lower than the cost of $\vec{r}$, is chosen as the paired solution $\vec{a}$. Thus, the *solution compare pair* is expressed as $Pair(\vec{a}, \vec{r})$. An example is illustrated in Algorithm 3.

## 4 Applying Our Approach to the RALIC Dataset

In this section, we illustrate the insights that can be obtained by applying the proposed framework on a large real-world example: the RALIC dataset.

### 4.1 Experimental Set Up

The detail of dataset, and the targeted objectives of the experimental study are presented as follows.

#### 4.1.1 Dataset

The RALIC project is an access control system developed at University College London, UK. This project was established in 2009 and deployed in 2011. The requirement data was collected by using the StakeNet stakeholder analysis method and StakeRare requirement elicitation method [35]. The implementation cost of each requirement was derived from the RALIC posterior implementation report. The cost is represented as the total man-hours spent on the requirement during the whole project development life cycle. The detail information of RALIC data is publicly published at http://soolinglim.wordpress.com/datasets/.

Because there is no uncertainty information about the attributes of requirements in RALIC dataset, we synthetically simulated these uncertainties following guidelines from the literature [6] which advocate a triangle probability distribution (illustrated in Fig. 3). In the early requirements engineering phase, due to the lack of definition or understanding of the requirements to be done, the level of software cost estimation accuracy ranged from 25 to 400 percent [36], [37], [38]. According to Jørgensen and Moløkken-Østvold's review [38], the Standish Group CHAOS Report [39] indicates that 52.7 percent of software projects will overrun the 89 percent of their original budget

estimation. Therefore, in our study, we define the range of uncertainty for requirement cost as $[25, 400]$ percent. There have also been studies on the accuracy of the software profit estimation or the satisfaction of stakeholders. Michael Bloch et al. study large-scale IT projects and report that the average benefit shortfall of IT projects is 56 percent, but no range is reported. As Fogelstrom et al. [40] pointed out in 2009, business risk-related uncertainty has received little attention, which means that we have little guidance as to the likely bounds we should place on uncertainty. Therefore, we have allowed for potential boundary scenarios in choosing our uncertainty bounds. That is, the range of uncertainty for satisfaction of stakeholder is defined as $[10, 300]$ percent. We believe that the true uncertainty value for any realistic project is likely to lie within this extreme range.

There are two versions of RALIC datasets: 'PointP' and 'RankP'. In this paper, we empirically studied our framework on the 'PointP' dataset, which consists of $143$ requirements, $86$ *And* dependencies, and $23$ *Or* dependencies. According to literature [42], the size and complexity of RALIC NRP instance are representative among the real world projects. However, it should be noted that there may still be more elaborate requirements and dependencies in the real world. Therefore, we cannot claim that RALIC NRP instances are fully representative of the real world. To generalise the study, three NRP instances are derived. There are two boundary scenarios, in which the uncertainty of a requirement is estimated, either highly optimistically or pessimistically, and one 'in-between' scenario. In highly optimistic scenarios, the requirements uncertainty is totally underestimated (mode value equals to the lowest value). By contrast, the requirements uncertainty is overestimated in highly pessimist scenarios (mode value equals to the highest value). After the pre-processing described in Section 3.1, there are $57$ refined requirements, and $4$ *Or* dependencies.

#### 4.1.2 NRP Objective Formulation

In our experiment, three attributes of software release planning were considered as the optimisation objectives: cost, satisfaction level, and the probability of budget overrun. The objective cost and satisfaction level were viewed as the utility of software release attainment and expressed as normalisation functions. We assumed that these two objectives were aggregatable. Thus, the expressions of the objective cost and satisfaction can be defined as

$$U(\vec{x}, cost) = \frac{\sum_{i=1}^{n}(x_i \cdot Cost_i)}{\sum_{i=1}^{n} Cost_i} \tag{18}$$

$$U(\vec{x}, satisfaction) = \frac{\sum_{i=1}^{n}(x_i \cdot Satisfaction_i)}{\sum_{i=1}^{n} Satisfaction_i}. \tag{19}$$

The quality of the solution is measured as the utility score of the solution as

$$Quality(\vec{x}) = U(\vec{x}, satisfaction) - U(\vec{x}, cost). \tag{20}$$

The expression of the probability of budget overrun remains the same (Eq. (13)). The extent $\theta$ is set as 150 percent. To reduce the simulations errors introduced by Monte-Carol Simulation, in our experimental study, the number of simulations is set as 10,000.

### 4.2 Research Questions

To evaluate the *METRO* framework, we carried out an experimental study to assess the ability of this approach to manage the algorithmic uncertainty and capture the impact of requirements uncertainties. In the experiment, we demonstrate why the requirements optimisation community should take care with algorithmic uncertainty, and how to employ *METRO* as a tool to assist decision makers to comprehend the results, thus raising three main research questions:

**RQ1:** How effective is *NSGDP* with respect to eliminating algorithmic uncertainty compared to NSGA-II?

We investigate how much difference can be observed between the solutions found by NSGA-II and *NSGDP*. This research question is a foundation for applying *NSGDP*. We compare the solutions found by NSGA-II with the benchmarks which are found by *NSGDP*. The differences between NSGA-II solutions and benchmarks reveal additional (unnecessary & unhelpful) uncertainty introduced by NSGA-II.

**RQ1.1:** How close are the solutions found by NSGA-II to the ones found by *NSGDP* in objective space?

**RQ1.2:** Comparing the solutions provided by *NSGDP* and NSGA-II, how much difference can be observed in design space?

The remaining research questions are more concerned with scrutinising the impact of uncertainty that came from requirement itself.

**RQ2:** After eliminating the algorithmic uncertainty by using *NSGDP*, what is the impact of the requirements uncertainty?

This question can be expressed in a quantified manner as to how much expected risk premium can be obtained when a decision moves from an optimal-yet-risky solution to a robust-yet-suboptimal one under the same budget.

**RQ3:** After eliminating the algorithmic uncertainty by using *NSGDP*, is there any pattern between the requirements characteristics and requirements inclusions? If so, what kind of pattern can be observed?

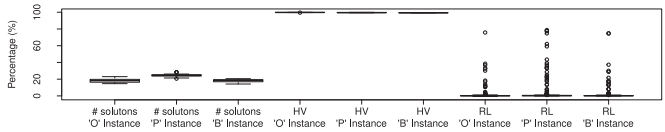
The third research question investigates the possible insight of the requirement characteristics, which may help decision makers to concentrate on the most interesting property of requirements. This question is composed of two more detailed sub-questions (*RQ3.1*, and *RQ3.2*):

**RQ3.1:** Which requirements are the most sensitive, so require the closest attention from the decision makers?

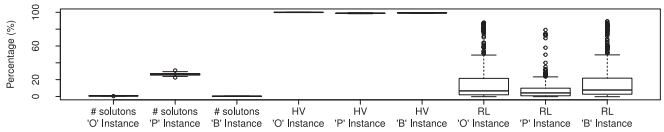
**RQ3.2:** Which requirements have the same inclusion behaviours, and can thus be clustered together?

### 4.3 Experiment Results

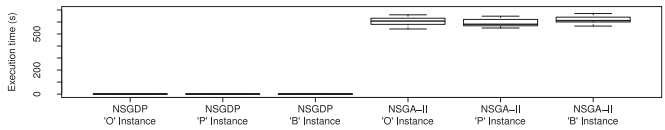
In this sub-section, we present the results of the experimental study, and provide a decision analysis guidance for decision makers by interpreting the research questions sequentially and separately.



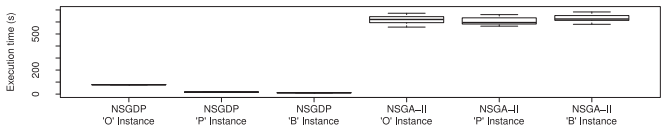

(a) # solution and HV of the results found by *point-NRP*




(b) # solution and HV of the results found by *simulation-NRP*




(c) The time for executing each approach (*point-NRP*)




(d) The time for executing each approach (*simulation-NRP*)

Fig. 4. **Answers** *RQ1.1*. These figures illustrate the differences between the solutions found by NSGA-II and *NSGDP*. Figs. 4a and 4b present the differences, based on three quality indicators (number of optimal solutions found, relative hypervolume, and relative revenue loss). The *NSGDP* results always correspond to the 100 percent line, by definition. Figs. 4c and 4d present execution time differences. '# solution' denotes the percentage of optimal solutions, 'HV' stands for relative hypervolume, and 'RL' indicates the percentage of revenue loss. The names of instance 'O', 'P', and 'B' stands for the highly Optimistic RALIC instance, and highly Pessimistic RALIC instance, and 'in-Between' RALIC instance, respectively.

#### 4.3.1 *RQ1: How Effective Is* NSGDP *with Respect to Eliminating Algorithmic Uncertainty Compared to NSGA-II?*

**RQ1.1** How close are the solutions found by NSGA-II to the ones found by *NSGDP* in objective space?

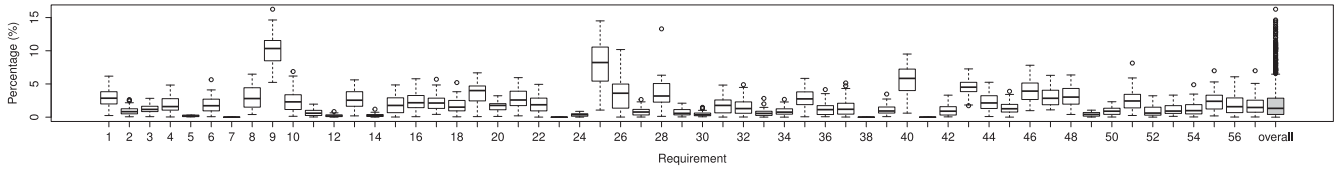
We answer this question by comparing the quality of solutions found by NSGA-II and *NSGDP*. Four quality indicators are used: the percentage of optimal solutions found; the relative hypervolume of the solution set; the percentage of revenue loss that is measured by comparing the revenue between NSGA-II solution and the *NSGDP* solution ($1 - \frac{Revenue(\vec{s}_{NSGA-II})}{Revenue(\vec{s}_{NSGDP})}$), and the execution time. Figs. 4a and 4b present three quality indicators of the solutions generated by NSGA-II in objective space. The execution times of NSGA-II and *NSGDP* are reported in Figs. 4c and 4d. We study the effectiveness of NSGA-II and *NSGDP* on three synthetic NRP instances. We execute NSGA-II on each instance over 30 runs. In order to intuitively observe the differences, in Figs. 4a and 4b, we report only the proportion of optimal solutions, and relative hypervolume of Pareto-front found by NSGA-II.

*RQ1.1 can be answered with Fig. 4*. In all cases, there are thousands of solutions on the true Pareto-fronts. In all three RALIC instances, the relative hypervolume of solutions found by NAGA-II ranges from 98.68 to 99.96 percent-fairly close to the optimal solutions. It denotes that, in our study, NSGA-II is able to find the solutions with a good convergence near the true Pareto-optimal front. This is because we allowed NSGA-II to use sufficient computation resources with 1,000 population and 1,000 generations. However, with respect to the number of optimal solutions found, NSGA-II may fail to find at least 73.03 percent of the optimal solutions. The percentage of missed optimal solutions can be yet up to 99.95 percent when considering uncertainty as an extra optimisation objective. Therefore, despite the fact that the convergence of NSGA-II is close to true Pareto-front for NRP, the randomness of NSGA-II makes it difficult to find complete optimal solutions. Moreover, compared to the inexact algorithm currently proposed to NRP, *NSGDP* saves 15.21 percent lost revenue on average for the RALIC dataset. It reveals that additional uncertainty is introduced to solutions by the algorithm itself. Additionally, according to Fig. 4c, when decision makers do not consider requirements uncertainty, they can get response from *NSGDP* immediately ($0.37s$ on average), and wait for up to $616.93s$ to get results from NSGA-II. If decision makers take requirements uncertainty into consideration, *NSGDP* is ($35.33s$ on average) still faster than NSGA-II ($675.26s$ on average) in general (Fig. 4d).

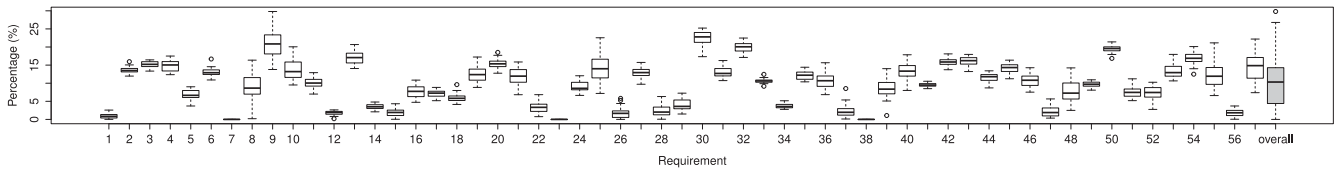
**RQ1.2** Comparing the decisions provided by *NSGDP* and NSGA-II, how much difference can be observed in design space?

According to the answer of *RQ1.1*, we can see that, even through NSGA-II converges to the true Pareto-front in objective space, it can find only a small proportion of optimal solutions. However, it is possible that such a small difference in objective space is caused by a prominent difference in design space. In order to investigate the hypothesis, we intend to inspect the *requirements selection probability*, which we define as the chance of requirement being included in the entire generated solution set. Therefore, we compare the overall requirements selection probability provided by NSGA-II and *NSGDP*, and analyse how much chance that the requirements decision is wrong when applying NSGA-II instead of exact approach. The probability of getting wrong requirements decision is measured by the difference of the requirements selection between NSGA-II solutions and benchmarks regarding to each requirement. Fig. 5 pictures the chance that NSGA-II gives wrong requirements decision with respect to each requirement and the overall probability. It depicts the essential impact raised from using an approximate algorithm.

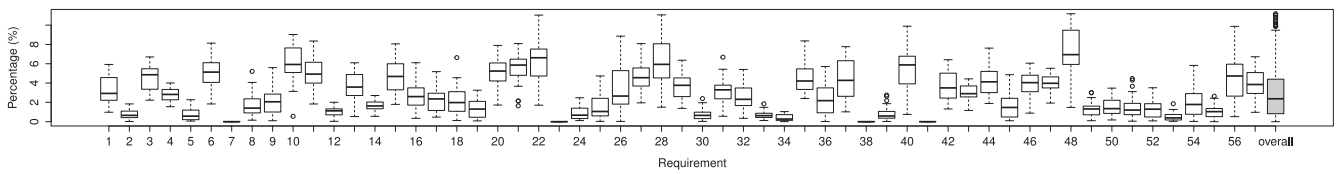
In RALIC experimental study, due to the effects of randomness from an approximate algorithm, in different runs, the requirements selections are volatile. According to Fig. 5, requirements uncertainty would aggravate the impact of the inexactness of NSGA-II in general. In all instances, the probability of receiving a wrong decision in *simulation-NRP* is almost double than that of *point-NRP*. Therefore, in the present algorithmic uncertainty, requirements uncertainty places decision makers at more serious risk of getting wrong requirements decision. We can see that, in an 'in-between' scenario, the upper bound of the chance of getting wrong requirements decision could rise from 16.25 to 36.48 percent, and the median overall chance rises from 2.01 to 10.94 percent). Even



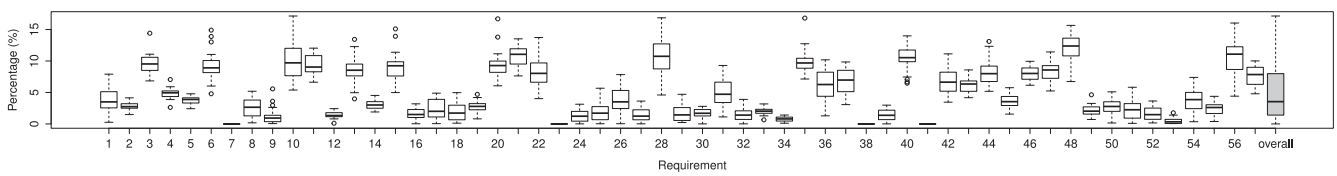

(a) Without considering uncertainty (*point-NRP* - highly optimistic scenario)



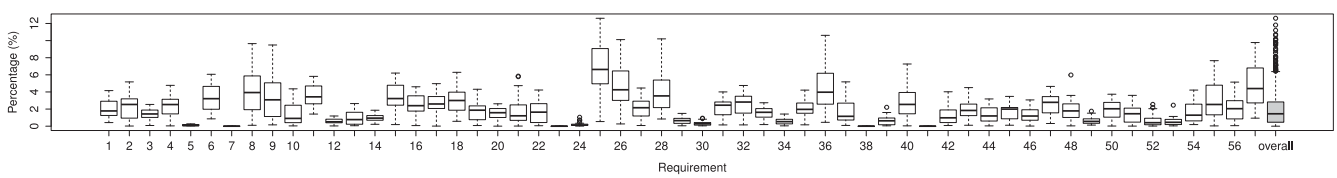

(b) Taking account of uncertainty (*simulation-NRP - highly optimistic scenario*)



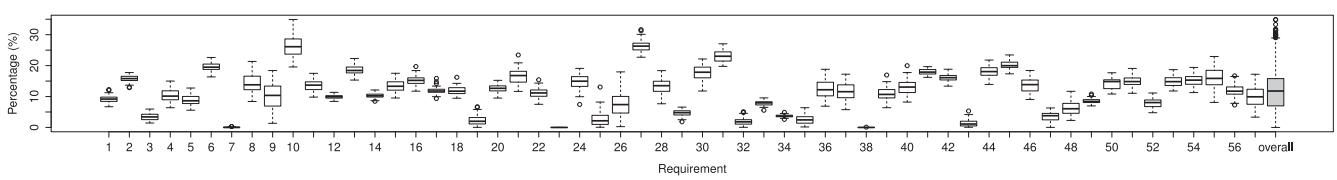

(c) Without considering uncertainty (*point-NRP* - highly pessimistic scenario)




(d) Taking account of uncertainty (*simulation-NRP - highly pessimistic scenario*)




(e) Without considering uncertainty (*point-NRP* - 'in-between' scenario)

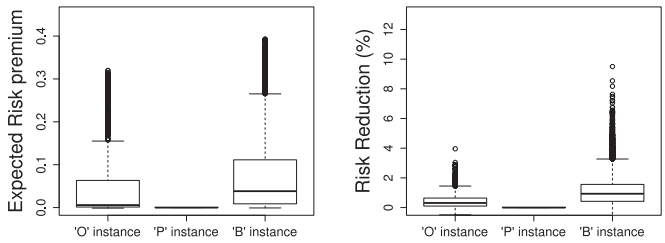



(f) Taking account of uncertainty (*simulation-NRP - 'in-between' scenario*)

Fig. 5. **Answers** *RQ1.2*. These box plots show the chance that NSGA-II provides wrong requirements selection decision for each requirement in RALIC instance. The grey box plot depicts the overall chance of getting wrong requirements decision.

if it were possible that in a particular run or particular scenario NSGA-II can offer a minor wrong decision, the non-determinism makes it produce a different answer in a different run. The erratic result may result in providing completely disorganised decisions. It emphasises that decision makers definitely should raise concerns about the impact of the stochastic algorithm on requirements selection. It is noteworthy that such impact implies some patterns. We found that the chance of getting a wrong decision is negatively correlated with the implementation cost of requirement (Spearman $\rho$ up to $0.72$ and $p \ll 0.0001$). That is, the larger the requirement implementation cost, the less the chance that making a wrong decision in requirements selection. There are only three exceptions (Requirement 7, 23, and 38). NSGA-II has nearly perfect match agreement on these three requirements over three NRP instances. The reason for this exception may be due to the inherit exclusive-or dependencies between these requirements (discussed in Section 4.3.3). (*This answers RQ1.2*).

In summary, to account for uncertainty in NRP, it is important for decision makers to understand the source of uncertainty in solution. Although NSGA-II can generate approximate solutions with good convergence with respect to objective space, the results of NSGA-II are still incomplete and suboptimal. However, the solution quality information is not meaningful for decision makers when they have to make decisions for each requirement. Even for the solutions which are very close to true Pareto-front, the decisions of selecting




(a) Expected risk premium (b) Reduction of risk

Fig. 6. The box-plots show how much expected risk premium and reduction of risk can achieve by taking account of requirements uncertainty. The names of instance 'O', 'P', and 'B' represents the highly optimistic RALIC instance, and highly pessimistic RALIC instance, and 'in-between' RALIC instance, respectively. **These figures answer** *RQ2*.

requirements are surprisingly distinguishable. In other words, relying on the requirements selections generated by NSGA-II would result in misleading the requirement decision. Consequently, the wrong decision would further cause the failure of a software project. Last but not least, *NSGDP* not only can guarantee the exactness of result but also outperform NSGA-II by offering a faster response to decision makers. This enables decision makers to receive feedback from our framework instantaneously. All of the above results emphasise the value of *NSGDP*, thus, answering why the requirement optimisation community should consider the exact approach, and promote the motivation of our research.

#### 4.3.2 RQ2: After Eliminating the Algorithmic Uncertainty by Using NSGDP, What Is the Impact of the Requirements Uncertainty?

After ruling out the algorithmic uncertainty, we would like to evaluate the impact of the requirements uncertainty. The results of the analysis are depicted in Fig. 6.

Fig. 6 statistically explains the impact of requirements uncertainties on RALIC NRP instance (with 150 percent budget overrun). There are 9,868, 1,149, and 31,417 optimal solutions found when considering requirements uncertainty (*sNPR*) in a highly optimistic scenario, a highly pessimistic scenario, and an 'in-between' scenario, respectively. And there are 221, 0, and 1,045 outliers in Fig. 6b, and 572, 0, and 967 outliers in Fig. 6a. The percentage of outliers is lower than 5.79 percent. It could be observed that, overlooking requirements uncertainty can contribute to suffer up to 10.09 percent risk that overrun more than 150 percent budget, and get at most 0.39 utility in return.

The impact of requirements uncertainty is negligible in a highly pessimistic scenario. This is because the worst case requirements uncertainty has been taken into account in requirements estimation. Taking account of uncertainty during requirements selection does not matter much for decision makers.

On the other hand, the impact of requirements uncertainty in a highly optimistic scenario is slightly less than in an 'in-between' scenario. This circumstance probably is the consequence of involving extremely large uncertainty in requirements estimation. In general, the principle of *simulation-NRP*, which provides robust-yet-suboptimal solutions, is replacing uncertain requirement with appropriate 'less uncertain' requirement(s). Meanwhile, all requirements in a highly optimistic scenario are extremely uncertain. There is a few relatively 'less uncertain' requirements can be chosen. Accordingly, our framework cannot reduce too much impact of requirements uncertainty in a highly optimistic scenario. In spite of sacrificing a little extra utility to reduce the risk by a small degree, this still offers decision makers more options than point-based estimation approach. For a decision maker, who is risk-averse, this risk reduction is more valuable than the gained utility for him or her. So the decision maker will not choose optimal-yet-risky solutions, and would accept the guaranteed robust-yet-suboptimal solutions. Otherwise, optimal-yet-risky solutions would be more attractive for risk-loving decision makers. (*This answers RQ2*).

In summary, requirements uncertainty would result in uncertainty for the overall software release plan. In order to minimise this risk, some loss of perceived utility must be accepted. The 'loss' involved is only a 'perceived' loss, in any case, because it is a loss that only occurs when the point estimate turns out to be absolutely 'spot on'. Since this is unlikely in general (one of the inherent problems of point-based or 'spot' estimates), the loss is therefore only a 'perceived' loss. That is, it is precisely the loss that will be perceived by an engineer/manager who (unrealistically) believes that point estimates are always spot on.

#### 4.3.3 RQ3: After Eliminating the Algorithmic Uncertainty by Using NSGDP, Is There Any Pattern Between the Requirements Characteristics and Requirements Inclusions?

The previous answer to research question *RQ2* offers a 'macroscopic' suggestion to a decision maker, helping them to understand the trade-off among different objectives. However, the result cannot provide more details about the nature of requirements, which may inspire decision makers to prioritise the requirements for further evaluation and inclusion. To aid this problem, *RQ3* promotes a detailed 'microscopic' investigation of requirements analysis for RALIC dataset.

*RQ3.1* Which requirements are the most sensitive, so require the closest attention from decision makers?

As distinct from conventional sensitivity analysis, here we take an algorithmic view of the problem. Fig. 7 describes the difference in the paired candidate solutions in terms of the requirements selection probability.

The difference in the paired candidate solutions is defined as follows. For a particular requirement $req$, set $A$ denotes the paired solutions that contain $req$ in *simulation-NRP* solutions, and set $B$ denotes the paired solutions that contain $req$ in *point-NRP* solutions. The intersection of these sets indicates the number of pairs that contain $req$ in both parts (set $A \cap B$). The height of bar in Fig. 7 is the symmetric difference of $A$ and $B$ ($A \triangle B$). More precisely, $A \setminus B$ is denoted by the height of the red (light grey in black and white) bar, while the size of $B \setminus A$ is denoted by the height of the blue (dark grey in black and white) bar. If the height of bar is 0, it means the selection of this requirement in all paired candidate solutions is identical. In this situation, the result reveals that, although it is unrealistic in general, in this specific instance the point-based estimate can be relied upon (even in the presence of the

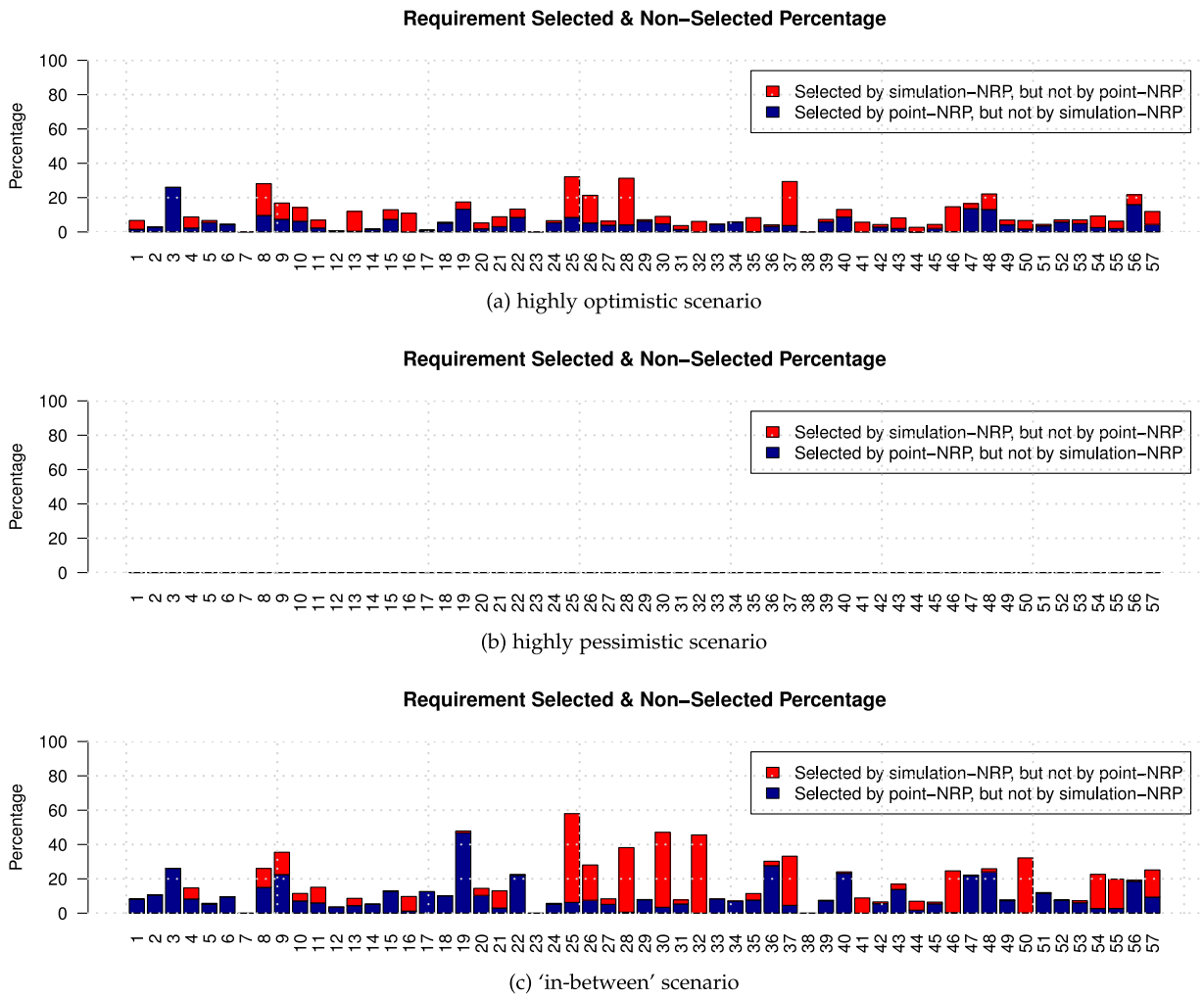


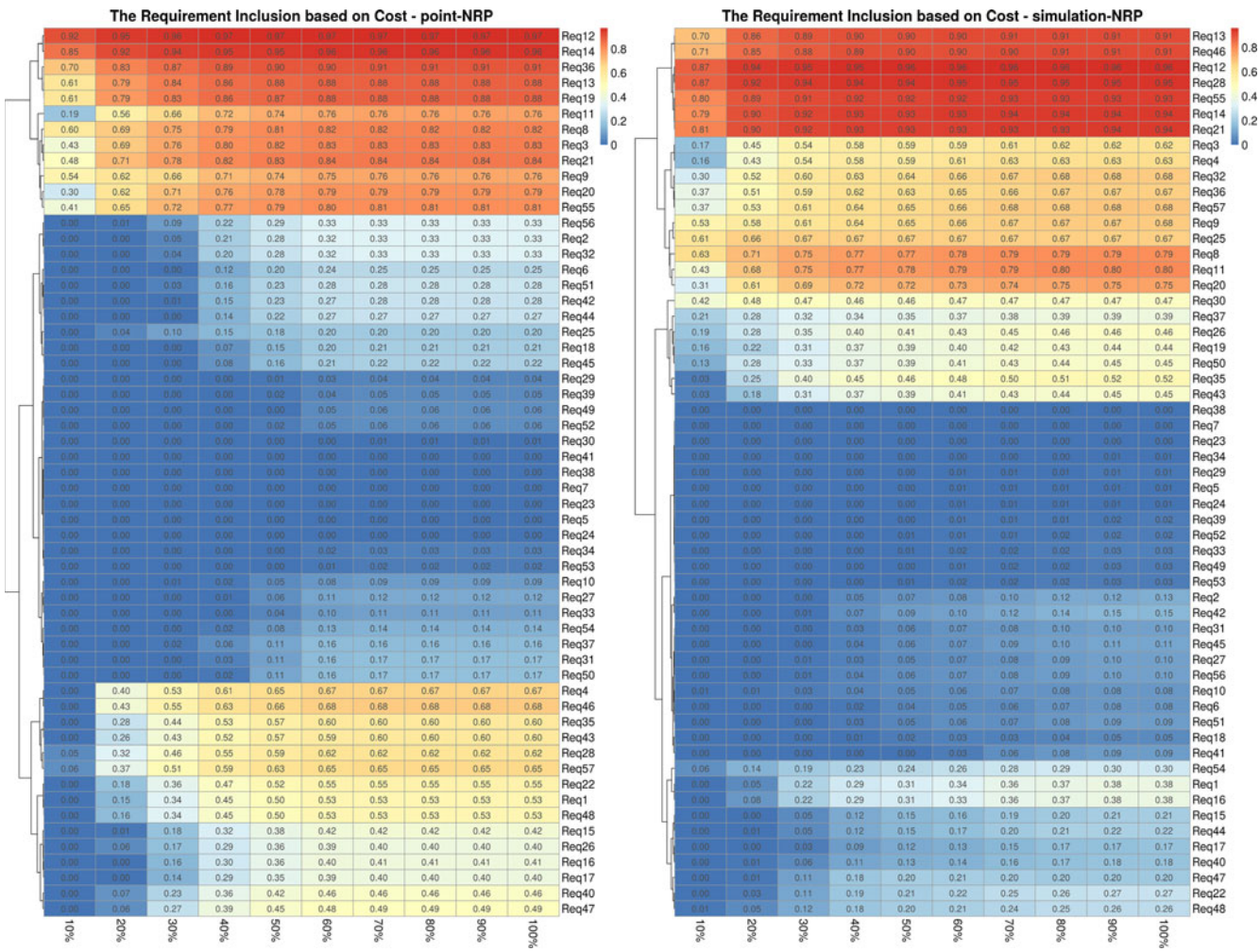

Fig. 7. **Answers** *RQ3.1*. The difference of requirement inclusion between robust-yet-suboptimal (*simulation-NRP*) solutions and the corresponding optimal-yet-risky solutions (*point-NRP*) in terms of requirements selection probability.

extreme range of risks we model ($10\% - 300$ percent), and two boundary NRP instances). From Fig. 7 we observe that 3 of the 57 requirements have this common property in all instances. For these 3 requirements, our analysis has thus revealed that we could simply revert to considering the point-based estimate as sufficiently robust, even in the presence of extreme risk. However, for the remaining 54 requirements, our analysis demonstrates the importance of modelling risk. Another interesting finding is, in a highly pessimistic scenario, there is no difference between *simulation-NRP* and *point-NRP* methods, and the difference is minor in the highly optimistic scenario. The reason has been discussed in Section 4.3.2.

We take Kendall's $\tau$ correlation coefficient to statistically analyse the correlation between the difference of requirements selection probability ($percentage\,(selected\_sNRP) - percentage\,(selected\_pNRP)$) and its own risk in the highly optimistic and 'in-between' scenarios experiments (where *sNRP* and *pNRP* are the abbreviations of *simulation-NRP* and *point-NRP*, respectively). The analysis result shows that there is a negative correlation between these two attributes ($p \ll 0.001$ and $\tau$ up to $-0.675$). Namely, the requirement with lower uncertainty has more chance to be selected by a risk-aware approach. Therefore, decision makers can observe the sensitivity of each requirement from the perspective of the algorithm. In RALIC experimental study, Requirement 3 is the most sensitive requirement with respect to 150 percent budget overrun in both highly optimistic and 'in-between' NRP scenarios, while Requirement 19 is the most sensitive in 'in-between' NRP scenarios. By contrast, Requirements 25, 26, 28, 37, and 46 are more insensitive. This recommends a risk-averse decision maker to be deeply concerned with Requirement 3, and assign high priority to Requirements 25, 26, 28, 37, and 46 (*answer RQ3.1*).

*RQ3.2* Which requirements have the same inclusion behaviours, and can thus be clustered together?

With increasing numbers of requirements, it will be tedious and time-consuming to analyse each requirement manually. Identifying inclusion behaviours, the tendency of including a requirement in the solutions on the Pareto-front as the budget increases, and analysing the differences between them may allow us to cluster requirements to reduce cognitive overload. *METRO* uses a heat-map (Fig. 8) to visualise the inclusion of requirements in the solutions on the Pareto-front with respect to the results generated by *point-NRP* and *simulation-NRP*. Moreover, in order to measure and highlight the similarities and differences, we cluster related requirements by computing the *complete euclidean* distance among requirements' inclusion percentage and present the results of the corresponding *Hierarchical Clustering*. This




(a) *point-NRP* (b) *simulation-NRP*

Fig. 8. **Answers** *RQ3.2*. The clustered inclusion trends of requirements where $\theta = 150$ percent.

approach is exemplified by the results of 'in-between' NRP instance. From Fig. 8, we can observe that there are 4 major clusters identified in the result of both *point-NRP* and *simulation-NRP*, which can help the decision maker to inspect at a much smaller number of groups of related requirements. Additionally, instead of prioritising all requirements, decision makers can first prioritise the requirements groups before prioritising the requirements within each cluster.

*The answer of RQ3.2 is that*, in 'in-between' RALIC instance, Requirement 12 is treated similarly with Requirement 13, 14, 19, 36 and prioritised as the most critical requirements group by *point-NRP* approach, while Requirement 12 is grouped with 13, 14, 21, 28, 46, and 55 which are all formalised as the most critical requirements by *simulation-NRP* approach. Therefore, in order to gain higher profit performance while minimising budget overrun risk, Requirements 12, 13, 14, 21, 28, 46, and 55 should be prioritised as the highest priority requirements group. Strikingly, some requirements could never be selected in any solution with any budget by the point-based approach. By looking at these requirements, we find that some of them (Requirements 3, 7, 23, and 41) participate in exclusive-*Or* dependencies. These four requirements are strongly dominated by corresponding mutually exclusive requirements in terms of optimisation goals. However, there is a nuance in the results offered by simulation-based approach. The risk-aware *simulation-NRP* approach *does* select Requirement 47 in some circumstances. The possible reason is that, by taking uncertainty into account, Requirement 47 is more robust than Requirement 41. Therefore, Requirement 47 can attract considerable attention when there is abundant budget (i.e., 50 percent of total budget) to neutralise its unrewarding traditional optimisation goals (i.e., revenue and cost).

To sum up, requirement characteristics play an important role in their inclusion in the solutions on Pareto-front. *METRO* can provide support to help decision makers identify relations between the different solutions by looking at the details of each individual solution. With respect to independent requirements, intrinsic uncertainty negatively correlates with inclusion when minimising solution risk. For mutually exclusive requirements, the inclusion of one requirement relies on the dominance of these requirements' fitness value. Therefore, the dominated requirements are seldom selected, compared with their conflicted twin.

## 5 Threats to Validity

In this section, we evaluate *METRO* with respect to its construct validity, internal validity, and external validity.

### 5.1 Construct Validity

In this paper, only triangle probability distribution is used to represent uncertainty. However, there are other kinds of uncertainties representation that might be used in risk analysis. For example, Gaussian distribution, uniform distribution, and discrete probability distribution. Catering for other distributions would not be a problem: our framework computes the estimation uncertainties of requirement attributes using MCS, and MCS can simulate most kinds of uncertainties straightforwardly (by sampling the scenarios based on input probability distribution directly). Therefore, *METRO* could use other kinds of uncertainty distribution to model the uncertainties of requirements.

### 5.2 Internal Validity

Internal validity is concerned with any possible factor that may perturb the experimental evaluations. The perturbations may include inappropriate parameter settings, and the implementation of algorithms. In our experimental study, during the experiment, we excluded other system applications, so the experimental machine only ran our application. Moreover, the solver of our framework is an exact approach, thus the stochastic nature of algorithm can be excluded.

In our experimental study, there is one internal threat to validity concerning the construction of probability distribution information for the requirements uncertainty. To gather such information we need access to historical project process data. However, this may be infeasible for researchers, due to confidentiality and/or other commercial concerns. The results shown in the experimental study may be affected when different probability distributions are used to represent requirement uncertainty.

The other threat to internal validity is concerned with the accuracy of the elicited probability distributions of requirements attributes. There are some scientific methods for eliciting the probability distributions of uncertainties, but such elicitation is sensitive due to the cognitive biases of the selected experts [33]. In our paper, due to the lack of uncertainty information within the RALIC data set, we generated the uncertainty distributions for the estimation error of the requirement cost, informed by a survey from literature. We cannot know the true estimate uncertainty for one project. Therefore, to minimise the impact of estimation accuracy, we study three synthetic NRP instances. There are two boundary scenarios, in which the uncertainty of a requirement is estimated either highly optimistically or pessimistically, and one 'in-between' scenario.

Therefore, we believe our framework *METRO* can solve real NRP with various uncertainties and provide valuable insight on decision analysis to decision makers.

### 5.3 External Validity

In the experimental study reported in the paper, we evaluated *METRO* on three synthetic data sets. These three data sets are derived from one real world data set from University College London, which contains two types of dependencies, and 143 raw requirements. However, the instances may bring external threats to the generalisability of our experimental results. The information gap between synthetic instances and real world instances may lead to a bias for the evaluation results. The experimental results for the RALIC NRP instances demonstrate that the proposed *METRO* framework can provide insights useful to a decision maker. However, there is no guarantee that, in all real world software projects, it would be possible to obtain such insights. Thus, we cannot generalise this experimental study to other NRP problems. For generalisation, more work is required to analyse different scenarios, models of uncertainty, as well as different NRP formulations.

With regard to scalability, *METRO* processes the RALIC dataset (with 57 refined requirements and 10,000 scenarios). The mean execution time for one *simulation-NRP* solver optimisation is 35.33s, and 0.37s for one *point-NRP* solver optimisation. The total execution time for one analysis is less than 40 seconds on average. The most critical threat for computational consumption is *NSGDP*. The running time of original Nemhauser-Ullmann algorithm is polynomially bounded in the size of the problem and the number of solutions on the Pareto-front. Since there are thousands of Pareto optimal solutions on the *simulation-NRP* Pareto-front, the execution time for one *simulation-NRP* optimisation grows exponentially compared with *point-NRP*. In addition, the second heaviest computational consumption is MCS. While the scale of the problem is increasing, our methods become more complex and time-consuming. A simple way to address this issue would be to reduce the number of scenarios. Consequently, the computation time will reduce, while the simulation error will increase. Since this paper focusses on proposing a novel framework to support decision analysis with uncertainty in NRP, optimising the performance of this approach will be an interesting further work.

## 6 Related Work

### 6.1 Previous Work on Requirement Optimisation and Analysis

The term *software requirement* is defined as "the property which must be exhibited in order to solve some problem in the real world" [43]. Essentially, a software requirement is typically a complex combination of sub requirements to satisfy different stakeholders and deployment environments [44]. In our research, requirements optimisation and analysis focuses on requirements selection and prioritisation.

In the literature, several authors have addressed decision analysis support for requirements optimisation and analysis. Saaty [45] introduced *Analytical Hierarchy Process* (AHP) to solve the decision support about planning, priority setting, and resource allocation in 1980. Karlsson adopted this approach for software requirements selection and prioritisation [1] in 1996, and extended it as a cost-value approach in 1997 [46]. However, the efficiency of AHP is limited by problem scale since it requires manual pairwise comparison of requirements. Linear programming techniques were introduced to assess requirements by Jung [47] in 1998. The assessment function was formulated as a single-objective function with a cost constraint function.

The term *Next Release Problem* was coined by Bagnall et al. [2] in 2001. The NRP aims to search for feasible and (near) ideal solution set to balance the requests from different stakeholders by applying various meta-heuristic algorithms. The NRP was formulated as a single-objective

optimisation problem by Bagnall et al. [2]. Zhang et al. [12] generalised the single objective NRP to multiple-objective NRP (MONRP).

### 6.2 Uncertainty & Risk Handling in General

In previous work on the problems of uncertainty, researchers usually adapt quantitative analysis methods, such as uncertainty analysis [48], to evaluate the robustness of the "model". On the other hand, though the uncertainty analysis can evaluate how sensitive the solutions are to possible estimation uncertainties, it cannot offer robust solutions by itself, based on decision makers' degree of risk aversion. Some authors suggest to investigate uncertainties *during* the process of optimisation to immunise solutions against production tolerances, parameter drifts, and model sensitivities by robust optimisation rather than adopting the post-analysis approaches [15], [49], [50].

### 6.3 Uncertainty & Risk Handling in Requirements Engineering

To study the uncertainty in the area of requirements optimisation, Harman et al. [14] used a local sensitivity analysis approach "One-At-a-Time" [51] to analyse the data sensitivity of NRP and MONRP. This approach measured parameter sensitivity by perturbing variables upward or downward to try out various 'what-if' scenarios. Al-Emran et al. [52], [53] performed probabilistic sensitivity analysis to evaluate the impact of uncertainties in operational release planning and product release planning in 2010. To avoid stochastic nature of the meta-heuristic algorithms, in 2014, Harman et al. [4] applied a naive exact algorithm, a variant of the Nemhauser-Ullmann's algorithm [54], to study precise sensitivity analysis of NRP without considering requirements interaction.

In the meantime, independent researchers have developed different concepts of robustness, measurements for robustness, and robust optimisation approaches in different research disciplines. The applications and studies of robust optimisation can be widely found in other non software engineering research literature [55], [56], [57], [58], [59] but are seldom found in the requirements engineering literature. In requirements engineering, to the best of our knowledge, there are only four studies applying robust optimisation on requirements engineering area.

In 2011, Heaven and Letier first proposed an optimisation framework, which integrated with stochastic simulation, for guiding the choice of system design solutions on high-level goals in quantitative Goal Models [60]. In 2014, Letier et al. applied statistical decision theory to illustrate the *expected information value* of model parameters based on [60] to offer further decision suggestions [6]. Paixão and Souza were the first authors to introduce a robust optimisation framework to the NRP problem in 2013 [17]. They used interval to model the uncertainties of requirements implementation cost, and defined a small population of scenarios to represent the uncertainty of requirement value. Thus, the uncertainty of requirement value is represented as the discrete variable. Each scenario was assigned a probability of occurrence. The desired level of robustness of decision makers was determined by a *control parameter*. Their robust NRP model tries to maximise the overall release solution value with respect to all possible scenarios, while minimising the implementation cost of release solution with respect to worst case. Thereby, the outcome of their approach is a conservative robust solution, which can avoid the impact of uncertainty in a worst scenario. Li et al. [18] proposed a simulation-based robust NRP (*simulation-NRP*) model. They formalised the uncertainty of requirement cost and value by Probability Density Function. Then, Monte-Carol Simulation was employed to simulate thousands of scenarios to represent the uncertainties of NRP model. The consequence of uncertainty was modelled in the probabilistic way (e.g., the probability of project budget overrun) for monitoring. Finally, a multi-objective optimisation approach was applied to maximise the value of solution, minimise the cost of solution in terms of all corresponding scenarios, as well as minimising the likelihood of occurrence and the consequences of a given future undesirable event. The *simulation-NRP* model enables decision makers to balance the trade-off among expected cost and value of solutions based on the solutions' probabilistic robustness.

### 6.4 Novelty of Our Approach

Since it is concerned with uncertainty handling, *METRO* exploits robust optimisation to deal with the uncertainty which is inherent in all requirements optimisation problems. Compared to the previous work of Harman et al. [4], [14] and Al-Emran et al. [52], [53], *METRO* can provide solutions that guard against uncertainty (for different degrees of the decision-maker's risk aversion) whereas uncertainty analysis merely analyses the sensitivity of uncertain model parameters and their impacts. Our approach, therefore, advances on previous work by taking account of risk in the decisions suggested, rather than simply reporting upon its possible pernicious effects.

*METRO* is built on an exact algorithm, which is used as the core NRP solver. Previous work by Heaven et al. [60], Paixão et al. [17], and Li et al. [18] relied solely upon (non-deterministic) randomised meta-heuristic algorithms. While our exact algorithm will always find the optimal solution, these previously proposed meta-heuristic algorithms may only find reasonable approximate solutions. While this is acceptable in general, for the specific problem of handling risk we face here, it is important for the decision maker to know that all uncertainty derives from the problem itself and not from the algorithm used to tackle it. *METRO* thus avoids information loss due to the non-deterministic nature of the approximation algorithm used as the core NRP solver.

Letier et al. [6] seek to overcome the limitations of approximate meta-heuristic algorithms, by using an exhaustive search as the core solver. Although the exhaustive search will always find the optimal solution, it is inherently expensive and may not scale sufficiently to be more generally applicable. For a NRP model which consists of $n$ requirements, there are $2^n$ solutions in the objective space. *METRO* introduces requirements interaction pre-processing and the conflict graph to augment the previous exact algorithms [26] thereby further enhancing scalability.

## 7 Conclusion

Uncertainty is inevitable in early requirements engineering. The requirements engineering community has explored

quantitative multi-objective decision techniques and search-based approaches to produce optimal solutions to requirements decision support problems [10], [12], [61], [62]. Decision makers are informed of possible trade-offs among conflicting objectives by visualising the Pareto optimal solutions generated by these quantitative methods. However, little work has been done to model design time uncertainties, to interpret the consequences of those uncertainties, and to support decision makers in analysing the inherent characteristics of model parameters [4], [6], [14], [18], [60]. Decisions that have to be made under incomplete knowledge about software project.

In this paper, we introduced a decision analysis framework, *METRO*, incorporating multi-objective simulation optimisation techniques, exact optimisation, and uncertainty analysis, to systematically aid decision support and analysis in the presence of uncertainty. We argued that eliminating algorithmic uncertainty may reduce revenue loss. We also argued that exposing the uncertainties and assessing the impacts of uncertainties quantitatively (by systematically analysing the variations between the point-based estimate approach and a simulation-based approach) can provide better decision insights than looking at point-based estimates alone. Our framework interprets the information derived from the results of the point-based estimate approach and a simulation-based approach to address decision analysis. The derived information allows requirements engineers to judge and weigh the trade-off between relatively robust-yet-suboptimal solutions and apparently optimal-yet-risky solutions. It also helps them in identifying highly 'sensitive' requirements, according to their robustness. This enables them to make decisions and evaluate requirements incrementally. Additionally, our framework also explains the inclusion patterns of requirements from two points of view. These patterns could inspire decision makers to capture elaborate requirement priorities, which can reduce the cognitive load on the decision maker. Future work will conduct more empirical studies as well as applying the *METRO* approach to the real world data sets to yield a better understanding of its value and wider applicability.

## Acknowledgments

The authors greatly thank Francisco Palomo-Lozano for his helpful suggestion. The work reported herein was supported by the China Scholarship Council (CSC) and the UK EPSRC project EP/J017515 (DAASE). The authors are also grateful to Kathy Harman for additional proof reading.

## References

[1] J. Karlsson, "Software requirements prioritizing," in *Proc. 2nd Int. Conf. Requirements Eng.*, Apr. 1996, pp. 110–116.
[2] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 883–890, 2001.
[3] M. Harman and B. F. Jones, "Search-based software engineering," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 833–839, 2001.
[4] M. Harman, J. Krinke, I. Medina-Bulo, F. Palomo-Lozano, J. Ren, and S. Yoo, "Exact scalable sensitivity analysis for the next release problem," *ACM Trans. Softw. Eng. Methodology*, vol. 23, no. 2, pp. 19:1–19:31, Apr. 2014.
[5] N. Veerapen, G. Ochoa, M. Harman, and E. K. Burke, "An integer linear programming approach to the single and bi-objective next release problem," *Inf. Softw. Technol.*, vol. 65, pp. 1–13, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584915000658
[6] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 883–894.
[7] H. Ziv, D. Richardson, and R. Klösch, "The uncertainty principle in software engineering," 1996.
[8] J. Helton, J. Johnson, C. Sallaberry, and C. Storlie, "Survey of sampling-based methods for uncertainty and sensitivity analysis," *Rel. Eng. Syst. Safety*, vol. 91, no. 10–1, pp. 1175–1209, 2006.
[9] W. S. Humphrey, *A Discipline for Software Engineering*, 1st ed. Boston, MA, USA: Addison-Wesley, 1995.
[10] A. Finkelstein, M. Harman, S. Mansouri, J. Ren, and Y. Zhang, ""Fairness analysis" in requirements assignments," in *Proc. 16th IEEE Int. Requirements Eng.*, Sep. 2008, pp. 115–124.
[11] D. Greer and G. Ruhe, "Software release planning: An evolutionary and iterative approach," *Inf. Softw. Technol.*, vol. 46, no. 4, pp. 243–253, 2004.
[12] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, 2007, pp. 1129–1137.
[13] A. Budzier, "Why your IT project maybe riskier than you think," *Harvard Business Rev.*, vol. 89, no. 9, pp. 23–25, 2011.
[14] M. Harman, J. Krinke, J. Ren, and S. Yoo, "Search based data sensitivity analysis applied to requirement engineering," in *Proc. 11th Annu. Conf. Genetic Evol. Comput.*, 2009, pp. 1681–1688.
[15] H.-G. Beyer and B. Sendhoff, "Robust optimization-a comprehensive survey," *Comput. Methods Appl. Mechanics Eng.*, vol. 196, no. 33, pp. 3190–3218, 2007.
[16] A. L. Soyster, "Technical note: Convex programming with set-inclusive constraints and applications to inexact linear programming," *Operations Res.*, vol. 21, no. 5, pp. 1154–1157, 1973.
[17] M. Paixão and J. Souza, "A scenario-based robust model for the next release problem," in *Proc. 15th Annu. Conf. Genetic Evol. Comput. Conf.*, 2013, pp. 1469–1476.
[18] L. Li, M. Harman, E. Letier, and Y. Zhang, "Robust next release problem: Handling uncertainty during optimization," in *Proc. Conf. Genetic Evol. Comput.*, 2014, pp. 1247–1254.
[19] Y. Zhang, M. Harman, A. Finkelstein, and S. A. Mansouri, "Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation," *Inf. Softw. Technol.*, vol. 53, no. 7, pp. 761–773, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584911000334
[20] J. M. Hammersley, D. C. Handscomb, and G. Weiss, "Monte Carlo methods," *Physics Today*, vol. 18, 1965, Art. no. 55.
[21] Y. Zhang, M. Harman, and S. L. Lim, "Empirical evaluation of search based requirements interaction management," *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 126–152, Jan. 2013.
[22] A. Atamtrk, G. L. Nemhauser, and M. W. Savelsbergh, "Conflict graphs in solving integer programming problems," *Eur. J. Oper. Res.*, vol. 121, no. 1, pp. 40–55, 2000.
[23] K. Jansen and S. Öhring, "Approximation algorithms for time constrained scheduling," *Inf. Comput.*, vol. 132, no. 2, pp. 85–108, Feb. 1997.
[24] K. L. Hoffman and M. Padberg, "Solving airline crew scheduling problems by branch-and-cut," *Manage. Sci.*, vol. 39, no. 6, pp. 657–682, Jun. 1993.
[25] U. Pferschy and J. Schauer, "The knapsack problem with conflict graphs," *J. Graph Algorithms Appl.*, vol. 13, no. 2, pp. 233–249, 2009.
[26] G. L. Nemhauser and Z. Ullmann, "Discrete dynamic programming and capital allocation," *Manage. Sci.*, vol. 15, no. 9, pp. 494–505, 1969.
[27] T. Brunsch and H. Röglin, "Improved smoothed analysis of multi-objective optimization," in *Proc. 44th Annu. ACM Symp. Theory Comput.*, 2012, pp. 407–426.
[28] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. Dag, "An industrial survey of requirements interdependencies in software product release planning," in *Proc. 5th IEEE Int. Symp. Requirements Eng.*, 2001, Art. no. 84.
[29] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 132–190, Jun. 2003.
[30] W. Royce, *Software Project Management*. Noida, UP, India: Pearson Education, 1998.

[31] T. DeMarco and T. Lister, *Waltzing with Bears: Managing Risk on Software Projects*. Reading, MA, USA: Addison-Wesley, 2013.
[32] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule, "Identifying software project risks: An international delphi study," *J. Manage. Inf. Syst.*, vol. 17, no. 4, pp. 5–36, Mar. 2001.
[33] A. O'Hagan, et al., *Uncertain Judgements: Eliciting Experts' Probabilities*. Hoboken, NJ, USA: Wiley, 2006.
[34] J. W. Pratt, "Risk aversion in the small and in the large," *Econometrica*, vol. 32, no. 1/2, pp. 122–136, 1964.
[35] S. L. Lim and A. Finkelstein, "Stakerare: Using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 707–735, May 2012.
[36] B. W. Boehm, *Software Engineering Economics*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall, 1981.
[37] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale IT projects on time, on budget, and on value," *Harvard Business Rev.*, McKinsey and University of Oxford in 2012. 2013.
[38] M. Jørgensen and K. Moløkken-Østvold, "How large are software cost overruns? A review of the 1994 {CHAOS} report," *Inf. Softw. Technol.*, vol. 48, no. 4, pp. 297–301, 2006.
[39] J. Lynch, "The Standish group report," [Online]. Available: http://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf, 1994, Accessed on: Jan. 12, 2015.
[40] N. Fogelstrom, M. Svahnberg, and T. Gorschek, "Investigating impact of business risk on requirements selection decisions," in *Proc. 35th Euromicro Conf. Softw. Eng. Advanced Appl.*, Aug. 2009, pp. 217–223.
[41] J. C. Felli and G. B. Hazen, "Sensitivity analysis and the expected value of perfect information," *Medical Decision Making*, vol. 18, no. 1, pp. 95–109, 1998.
[42] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, and S. Brinkkemper, "An empirical study of meta-and hyper-heuristic search for multi-objective release planning," *Res. Note*, vol. 14, 2014, Art. no. 7.
[43] A. Abran, P. Bourque, R. Dupuis, and J. W. Moore, Eds., *Guide to the Software Engineering Body of Knowledge—SWEBOK*. Piscataway, NJ, USA: IEEE Press, 2001.
[44] I. F. Alexander and L. Beus-Dukic, *Discovering Requirements: How to Specify Products and Services*. Hoboken, NJ, USA: Wiley, 2009.
[45] T. Saaty, *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. New York, NY, USA: McGraw-Hill, 1980.
[46] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Softw.*, vol. 14, no. 5, pp. 67–74, Sep. 1997.
[47] H.-W. Jung, "Optimizing value and cost in requirements analysis," *IEEE Softw.*, vol. 15, no. 4, pp. 74–78, Jul. 1998.
[48] Z. Zi, "Sensitivity analysis approaches applied to systems biology models," *IET Syst. Biol.*, vol. 5, no. 6, pp. 336–346, Nov. 2011.
[49] H. Zhang, B. Kitchenham, and D. Pfahl, "Software process simulation modeling: Facts, trends and directions," in *Proc. 15th Asia-Pacific Softw. Eng. Conf.*, Dec. 2008, pp. 59–66.
[50] A. Ben-Tal andA. Nemirovski, " Robust optimization methodology and applications," *Math. Program.*, vol. 92, no. 3, pp. 453–480, 2002.
[51] A. Saltelli, S. Tarantola, and F. Campolongo, " Sensitivity analysis as an ingredient of modeling," *Statistical Sci.*, vol. 15, no. 4, pp. 377–395, 2000.
[52] A. Al-Emran, P. Kapur, D. Pfahl, and G. Ruhe, "Studying the impact of uncertainty in operational release planning—an integrated method and its initial evaluation," *Inf. Softw. Technol.*, vol. 52, no. 4, pp. 446–461, Apr. 2010.
[53] A. Al-Emran, D. Pfahl, and G. Ruhe, "Decision support for product release planning based on robustness analysis," in *Proc. 18th IEEE Int. Requirements Eng. Conf.*, 2010, pp. 157–166.
[54] G. L. Nemhauser and Z. Ullmann, "Discrete dynamic programming and capital allocation," *Manage. Sci.*, vol. 15, no. 9, pp. 494–505, 1969.
[55] M. Li, S. Azarm, and V. Aute, "A multi-objective genetic algorithm for robust design optimization," in *Proc. Conf. Genetic Evol. Comput.*, 2005, pp. 771–778.
[56] E. Kazancioglu, et al., "Robust optimization of an automotive valvetrain using a multiobjective genetic algorithm," in *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2003, pp. 97–108.
[57] M. Papadrakakis, N. Lagaros, and V. Plevris, "Design optimization of steel structures considering uncertainties," *Eng. Struct.*, vol. 27, no. 9, pp. 1408–1418, 2005.
[58] A. Thompson and P. Layzell, " Evolution of robustness in an electronics design," in *Evolvable Systems: From Biology to Hardware*, J. Miller, A. Thompson, P. Thomson, and T. Fogarty, Eds. Berlin, Germany: Springer, 2000, pp. 218–228.
[59] A. Kumar, A. J. Keane, P. B. Nair, and S. Shahpar, "Robust design of compressor fan blades against erosion," *J. Mech. Des.*, vol. 128, no. 4, pp. 864–873, 2006.
[60] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in *Proc. 19th IEEE Int. Requirements Eng. Conf.*, Aug. 2011, pp. 79–88.
[61] Y. Zhang, A. Finkelstein, and M. Harman, " Search based requirements optimisation: Existing work and challenges," in *Requirements Engineering: Foundation for Software Quality*, B. Paech and C. Rolland, Eds. Berlin, Germany: Springer, 2008, pp. 88–94.
[62] Y. Zhang, E. Alba, J. J. Durillo, S. Eldh, and M. Harman, "Today/future importance analysis," in *Proc. 12th Annu. Conf. Genetic Evol. Comput.*, 2010, pp. 1357–1364.

**Lingbo Li** received the BSc degree from Southeast University, China and the MSc degree from University of Birmingham, United Kingdom. He is currently working toward the PhD degree in the CREST Centre, Department of Computer Science, University College London, under the supervision of Professor Mark Harman and Doctor Emmanuel Letier. His current research focuses on applying search-based techniques on software requirements engineering, developing decision support analysis methods to assist software project decision makers in semi-automatically analysing, and evaluating large scale software systems.

**Mark Harman** is professor of Software Engineering in the Department of Computer Science at University College London, where he directs the CREST centre and is Head of Software Systems Engineering. He is known for work on source code analysis, software testing, app store analysis and Search Based Software Engineering (SBSE), a field he co-founded and which has grown rapidly to include over 1,600 authors spread over more than 40 countries. His work has been used by many organisations including Daimler, Ericsson, Google, Huawei, Microsoft and Visa. Prof. Harman is co-director of Appredict, an app store analytics company, spun out from UCL's UCLappA group, and chief scientific advisor to MaJiCKe, and automated test data generation start up.

**Fan Wu** received the BSc degree from Tsinghua University, China. He is currently working toward the PhD degree in CREST Centre, Department of Computer Science, University College London, supervised by Professor Mark Harman and Doctor Jens Krinke. He is interested in automatic software improvement using search-based approaches, automating the search for best trade-off between multiple non-functional properties, and Mutation Testing for memory vulnerabilities.

**Yuanyuan Zhang** received the PhD in software engineering from Kings College London, in 2010. She is currently a principal research associate in the CREST centre, University College London. Her research interests include search-based requirements optimisation, app store mining and analysis and evolutionary computation. She has published more than 20 papers including RE and RE journal. She is the co-author of several invited keynote papers at leading international conferences, including SPLC 2014 and ICST 2015. She has served on program committees including GECCO, SSBSE, AIRE, MOBS, RELENG, RET and as the program co-chair for SSBSE 2013 and been elected onto the steering committee for SSBSE.