Search-based Uncertainty-wise Requirements Prioritization

Yan Li[1] , Man Zhang[2], Tao Yue[2,3], Shaukat Ali[2] and Li Zhang[1]
[1]Beihang University, [2]Simula Research Laboratory, [3]University of Oslo
yanll@buaa.edu.cn
{manzhang, tao, shaukat}@simula.no
lily@buaa.edu.cn

***Abstract*—To ensure the quality of requirements, a common practice, especially in critical domains, is to review requirements within a limited time and monetary budgets. A requirement with higher importance, larger number of dependencies with other requirements, and higher implementation cost should be reviewed with the highest priority. However, requirements are inherently uncertain in terms of their impact on the requirements implementation cost. Such cost is typically estimated by stakeholders as an interval, though an exact value is often used in the literature for requirements optimization (e.g., prioritization). Such a practice, therefore, ignores uncertainty inherent in the estimation of requirements implementation cost. This paper explicitly taken into account such uncertainty for requirement prioritization and formulates four objectives for uncertainty-wise requirements prioritization with the aim of maximizing 1) the importance of requirements, 2) requirements dependencies, 3) the implementation cost of requirements, and 4) cost overrun probability. We evaluated the multi-objective search algorithm NSGA-II together with Random Search (RS) using the RALIC dataset and 19 artificial problems. Results show that NSGA-II can solve the requirements prioritization problem with a significantly better performance than RS. Moreover, NSGA-II can prioritize requirements with higher priority earlier in the prioritization sequence. For example, in the case of the RALIC dataset, the first 10% of prioritized requirements in the prioritization sequence are on average 50% better than RS in terms of prioritization effectiveness.**

Keywords—Requirements Prioritization, Uncertainty, Search Algorithms, Multi-Objective Optimization.

I. INTRODUCTION

The quality of requirements is critical in any non-trivial system/software development lifecycle and requirements are expected to possess certain attributes such as unambiguity, consistency, and completeness [1]. To ensure that requirements meet such attributes, the common practice is via conducting requirements reviews. In a reviewing process, relevant stakeholders often play the role as reviewers, who are supposed to identify requirement issues against concerned requirements attributes [1]. A requirement with higher importance, the dependence of on a large number of other requirements, with a higher cost to implement, should be given more weight than the other requirements during a review process. Such a review process often has a limited time and monetary budget. Therefore, it is practically needed to have an automated requirements prioritization solution for review by taking into account different requirements attributes.

Search Based Software Engineering (SBSE) has been applied and proven to be effective in requirements prioritization [2]. In the literature, requirements are usually prioritized according to their characteristics such as importance, dependency, estimated benefit to stakeholders and cost in terms of resource required to realize requirements [2-4]. Unfortunately, uncertainty in requirements implementation cost and cost overrun probability are rarely considered, while uncertainty is an inherent characteristic of software engineering [5].

From our perspective, uncertainty in requirements engineering is the "lack of knowledge" of requirements engineers about the system to be developed, the development process, customer needs, and any other factors that take requirements as input to produce important system artifacts (e.g., architecture and implementation) and support critical decision making, along the system development lifecycle. Such uncertainty may have an impact on various phases of the system development [6], the feasibility, cost, and duration of requirements implementation [7]. As recommended in [8], requirement implementation cost should be taken into account when prioritizing requirements, which is often estimated by stakeholders as an interval. An exact value is often taken for requirements prioritization, which does not take into account uncertainty in requirements implementation cost, thereby, putting projects in risk and even leading the projects to fail.

Uncertainty should be taken into account by mathematical measurement and risk management methods in requirements optimization problems. Researchers have used uncertainty management into requirements engineering decision for the Next Release Problems (NRP) [6, 9]. In the requirements reviewing process, the uncertainty of requirements implementation cost should also be measured quantitatively when exploring the solutions for requirements prioritization.

Ideally, requirements with higher uncertainty in implementation cost, measured with cost overrun probability, should be given a higher priority for review and one of requirements review objectives could be disclosing and analyzing uncertainty in requirements such that the impact of such uncertainty could be mitigated, controlled and/or eliminated in an elegant manner during the following development phases of a project.

In this paper, we formulate the objectives of requirements prioritization as maximizing: 1) the importance of require-

ments (*IMP*), 2) requirements dependencies (*DEP*), 3) the implementation cost of requirements (*COST*), and 4) probability of requirements cost overrun (*PCO*). To obtain *PCO* information, the Monte-Carlo Simulation (MCS) based on the triangular distribution [10] is chosen to simulate scenarios to generate requirements implementation cost, since MCS simulations can compute good approximations in a statistical cost analysis [7], in which a triangle probability distribution is advocated to simulate the uncertainty in requirements cost. Based on these simulation scenarios, the probability of cost overrun is calculated. We evaluated the fitness function using the RALIC dataset [11] as the real-world case study with a multi-objective search algorithm: Fast Nondominated Sorting Genetic Algorithms (NSGA-II), together with Random Search (RS) as the comparison baseline. In addition, to assess the scalability of the fitness function together with the search algorithms, we used 19 artificial problems of varying complexity in terms of the number of requirements. Results of the evaluation show that: 1) NSGA-II achieved a better performance than RS for *IMP*, *DEP*, *COST*, and *PCO*, respectively, and NSGA-II can prioritize requirements faster, e.g., the first 10% of the requirements in the prioritization sequence have the highest prioritization effectiveness as compared to RS for the RALIC dataset ; 2) different mode values of the triangular distribution to calculate uncertainty in requirements implementation cost has little impact on the performance of the search algorithms; and 3) the performance of NSGA-II decreases with the increase in the number of requirements.

The rest of the paper is organized as follows. In Section II, we present a formal representation of our optimization problem and the fitness function. In Section III, empirical evaluation on the real-world case study and artificial problems is presented, followed by results and analysis (Section IV). Sections V, VI and VII present the threats to validity, related work, conclusion, and future work, respectively.

II. Problems Representation and Fitness Function

This section presents the problem representation (Section A), followed by uncertainty measurement in Section B, and the proposed fitness function in Section C.

A. Problem Representation

A system *S* has a set of requirements $Req = \{R_1, R_2, R_3 \ldots R_n\}$, where *n* is the total number of requirements in the set. Each requirement R_i has several associated attributes defined below:

1) *Importance*: Importance of requirement R_i ($importance_i$) represents its perceived importance. Requirements can be classified as inessential, desirable or mandatory [12]. Numerical values are usually used to represent requirements importance, which can be in a fine or coarse granularity, such as an integer ranging from 1 to 5, or a float ranging from 0 to 1 [13].

2) *Dependency*: A requirement might have a low priority from a stakeholder' perspective, but it is essential for the success of the system as it is the basis for other requirements' implementation. Such dependencies should be identified [14] since the implementation of a particular requirement might impact the implementations of requirements depending on it. As discussed in [15], requirements dependency is the relationship between requirements and more than twenty dependency types have been proposed. In our context, we use the "Requires" dependency type to describe that R_i has to be implemented before R_j as "R_j Requires R_i". In this paper, we use an integer $depNum_i$ to represent the dependency value of R_i, i.e., the number of requirements requiring R_i.

3) *Implementation Cost:* Each requirement needs resources such as human resource, instrumentations, and facilities, to implement it. We use an integer value $cost_i$ to measure the cost for each requirement (person hours) needed for implementing R_i.

4) *Cost Overrun Probability:* This attribute is related to uncertainty in requirement implementation cost overrun and each requirement R_i has an associated probability value p_i.

A solution of requirement prioritization $\vec{x} = \{x_1, x_2, \ldots x_n\}$ is a sequence of requirements, in which x_i means the position of requirement R_i in the sequence (ranging from *0* to *n-1*) and the value of x_i is unique. $x_i = j$ means that the requirement R_i is the j_{th} requirement in the sequence to be reviewed. A smaller value of x_i means a preceding position in sequence $\vec{x}$, that is, *0* means the first position, while *n-1* means the last position of a requirement. Thus, there are *n*! possible prioritization solutions for *n* requirements, where *n* is the number of requirements representing the number of decision variables and *n!* represents the size of search space.

B. Uncertainty Measurement

In our context, uncertainty occurs when stakeholders estimate the cost of implementing a requirement and is measured as *cost overrun probability*. As discussed in [7], Monte-Carlo Simulation (MCS) was used to simulate a large number of scenarios and compute good approximations of cost overrun probability. This method has been used to address the Next Release Problem (NRP) as reported in [6, 9]. In our requirements prioritization problem, requirements engineers usually can provide an estimated value of requirements implementation cost and ignore the uncertainty of implementation cost. Therefore, we apply MCS to simulate uncertainty in requirements implementation cost and therefore calculate the probability of requirements cost overrun based on simulated scenarios.

1) *Cost Probability Distribution*: The triangular distribution is advocated to simulate uncertainty in requirements implementation cost [7] and it is a continuous probability distribution with lower limit *a*, upper limit *b* and mode *c*, where $a \leq c \leq b$. The probability density function (PDF) is defined in [10] as:

$$f(x) = \begin{cases} 0, & x < a \text{ , or } x > b \\ \frac{2(x-a)}{(b-a)(c-a)}, & a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)}, & c < x \leq b \end{cases} \quad (1)$$

In Section III.A, we will discuss how to determine the triangular distribution parameters (*a*, *b*, and *c*) of the implementation cost of a requirement in our experiments.

2) *The probability of Cost Overrun under MCS:* Based on the probability distribution of the cost of implementing a requirement, many simulation scenarios can be simulated through MCS. In a scenario, MCS samples a random value using the triangular distribution for a requirement, representing the implementation cost under the scenario. Thus, each requirement R_i under the simulated scenario *j*, has a cost value $cost_{i,j}$. The implementation cost of requirements under multi-simulation-scenarios are represented as:

$$cost(n, sn) = \begin{pmatrix} cost_{1,1} & \cdots & cost_{1,sn} \\ \vdots & \ddots & \vdots \\ cost_{n,1} & \cdots & cost_{n,sn} \end{pmatrix} \quad (2)$$

in which, *n* is the number of requirements and *sn* is the number of simulated scenarios.

Based on MCS, among *sn* scenarios, we count the times ts_i, of $cost_{i,j}$ larger than the implementation cost expected by stakeholders ($cost_i$, presented in Section II.A) for each requirement. The probability of cost overrun of requirement R_i (p_i) is represented as the frequency of cost overrun, which is calculated as:

$$p_i = P\{cost_{i,j} > cost_i\} = ts_i / sn \quad (3)$$

C. Fitness Function

Based on the definitions provided in Section A and Section B, we provide formal definitions of the four objectives of a solution $\vec{x}=\{x_1, x_2, \ldots x_n\}$ and the fitness function below.

1) Objective $IMP(\vec{x})$ calculates the total importance value of the prioritization sequence $\vec{x}$. To differentiate the impact of the position in $\vec{x}$ of a requirement to $IMP(\vec{x})$, each requirement R_i has a weight $\frac{n-x_i}{n}$ of $importance_i$ to $IMP(\vec{x})$, which means that a requirement in a preceding position has a bigger contribution to $IMP(\vec{x})$.

$$IMP(\vec{x}) = \frac{\sum_{i=1}^{n} \frac{n-x_i}{n} * importance_i}{\sum_{i=1}^{n} importance_i} \quad (4)$$

2) Objective $DEP(\vec{x})$ calculates the total dependency value of the prioritization sequence $\vec{x}$. Similar to $IMP(\vec{x})$, each requirement R_i has a weight $\frac{n-x_i}{n}$ related to its position in $\vec{x}$.

$$DEP(\vec{x}) = \frac{\sum_{i=1}^{n} \frac{n-x_i}{n} * depNum_i}{\sum_{i=1}^{n} depNum_i} \quad (5)$$

3) Objective $COST(\vec{x})$ calculates the total expected cost value of the prioritization sequence $\vec{x}$. Similarly, each requirement R_i has a weight $\frac{n-x_i}{n}$ related to its position in $\vec{x}$.

$$COST(\vec{x}) = \frac{\sum_{i=1}^{n} \frac{n-x_i}{n} * cost_i}{\sum_{i=1}^{n} cost_i} \quad (6)$$

4) Objective $PCO(\vec{x})$ calculates the total probability value of the prioritization sequence $\vec{x}$.

$$PCO(\vec{x}) = \frac{\sum_{i=1}^{n} \frac{n-x_i}{n} * p_i}{\sum_{i=1}^{n} p_i} \quad (7)$$

Based on Formula (4) -(7), the problem of requirements prioritization for review can be formulated as: searching for a solution $\vec{x}$ from *n!* solutions that should:

Maximize ($IMP(\vec{x})$), Maximize ($DEP(\vec{x})$), Maximize ($COST(\vec{x})$), and Maximize ($PCO(\vec{x})$).

III. Empirical Evaluation

A. Experiment Design

Our experiments aim to evaluate the fitness function addressing our requirements prioritization problem with two algorithms: NSGA-II and RS (as the baseline for comparison) for the RALIC dataset and 19 artificial problems.

1) *Research questions:* We address the following research questions: ***RQ1:*** Is NSGA-II effective to solve our prioritization problem, compared with RS? ***RQ2:*** How does mode *c* of the triangular distribution impact the performance of the search algorithms? ***RQ3:*** How does the increment in the number of requirements impact the performance of NSGA-II?

RQ1 helps us in assessing whether the requirements prioritization problem is complex and deserves the use of a complex multi-objective search algorithm. RQ2 studies the impact of mode *c* of the triangular distribution used to simulate uncertainty in requirements implementation cost on the performance of NSGA-II. RQ3 analyzes the impact of the growth of the number of requirements on the performance of the algorithms. We used several metrics to compare the performance of NSGA-II and RS, including the objectives (*IMP*, *DEP*, *COST*, and *PCO*) and *HV* (Hypervolume), measuring the convergence and uniform diversity of a Pareto front [16]).

To illustrate how fast NSGA-II prioritizes requirements for RQ1, we introduced an effectiveness metric for measuring how effective of an algorithm for prioritizing requirements by taking all the objectives into account. Firstly, we defined the overall fitness value (*OFV*) for a solution $\vec{x} = \{x_1, x_2, \ldots x_n\}$, based on each requirement's attributes (*importance*, *depNum*, *cost* and *p* defined in Section II.A) as:

$$OFV = \sum_{i=1}^{n} \frac{(importance_i + depNum_i + cost_i + p_i)}{4} \quad (8)$$

where *i* is the position of a requirement in a solution and *n* is the total number of requirements being prioritized.

Then, OFV_{fp} is the value of *OFV* in terms of prioritizing *fp*% of the total *n* number of requirements, as defined below:

$$OFV_{fp} = \sum_{i=1}^{n*fp\%} \frac{(importance_i + depNum_i + cost_i + p_i)}{4} \quad (9)$$

where *fp* could be 10, 20, …100.

Therefore, we define the improvement percentage of *OFV* in terms of prioritizing *fp*% of the total requirements, when comparing with RS, as:

$$AIPOFV_{fp} = \frac{OFV_{fp\_NSGA-II} - OFV_{fp\_RS}}{OFV_{fp\_RS}} \quad (10)$$

2) *RALIC Dataset:* RALIC is a large-scale software project to enhance the existing access control system at the University College London, having more than 60 stakeholders

groups and approximately 30,000 users [13]. The RALIC dataset contains data of stakeholders and their requirements, including raw requirements textual description, recommendations from stakeholders, and importance values from stakeholders on requirements. The RALIC dataset and additional data about the cost (person hours) for each requirement are both provided in [11]. There are 143 requirements and 31 dependency relations among these requirements that are used in our requirements prioritization problem.

There is no uncertainty data in the RALIC dataset. According to the conclusion in [6], the actual cost of implementing the requirements ranges from 25% to 400% of expected value and each requirement in the RALIC dataset has an implementation cost value *cv*, which is an expected cost value, estimated based on historical data. Thus, in the triangular distribution of requirements implementation cost (Formula (1) in Section II.B), we set the lower limit *a* as 25%**cv* and the upper limit *b* as 400%**cv* in Formula (5) to describe the triangular distribution of implementation cost of a requirement.

To discover the effect of different values of mode *c* on the prioritization problem, we selected five values for *c* ranging from *a* to *b* with an increment $\frac{1}{4}$*(b-a) and the values are *a*, $a+\frac{1}{4}$*(b-a) , $a+\frac{1}{2}$*(b-a) , $a+\frac{3}{4}$*(b-a)* and *b*, respectively. When we simulate 10, 000 scenarios for each value of mode *c* to compute good approximations of the probability of requirements cost overrun using MCS. Among the selected five values of mode *c*, there are two boundary scenarios: highly optimistic scenarios, i.e., the one with the mode value equals to the lowest value *a* and highly pessimist scenario, i.e., the one with the mode value equals to the highest value *b* [7]. The other three values ($a+\frac{1}{4}$*(b-a), $a+\frac{1}{2}$*(b-a), $a+\frac{3}{4}$*(b-a)*) represent the 'in-between' scenarios [6]. As discussed in [6], in highly optimistic scenarios, uncertainty in requirements implementation cost is underestimated, while in highly pessimist scenarios, such uncertainty is overestimated.

3) *Artificial Problems.* In addition, to evaluate whether the fitness function defined in Section II.C really addresses our requirements prioritization problem even with a different number of requirements, we carefully defined 19 artificial problems. We created them with the increasing number of requirements ranging from 100 to 1000 with an increment of 50. Note that, these artificial problems were inspired based on the RALIC dataset to represent problems of varying complexity. For each requirement, we randomly generated a value for the number of its dependent requirements and its estimated implementation cost value. Settings to the parameters of the triangular distribution (i.e., a, b, and c) are the same as for the RALIC dataset.

4) *Search Algorithms and Parameter Settings:* We used jMetal [16] for the implementation of NSGA-II and RS. The population size was set to 100 and the maximum number of evaluations was set to 25000. We used the suggested default parameters settings from [16] including a binary tournament for the selection of parents, the simulated binary crossover for recombination, and the polynomial mutation as the mutation operator.

B. Statistical Tests

To address RQ1, the Wilcoxon signed-rank test and the Vargha and Delaney statistics were used, based on the guidelines of using statistical tests for randomized algorithms [19] to compare the performance of NSGA-II and RS. The Wilcoxon signed-rank test with a significance level of 0.05 was used to calculate p-values for deciding whether there is a significant difference between the pair of the search algorithms.

The Vargha and Delaney statistics ($\widehat{A}_{12}$) was used to calculate the effect size. Given the maximizing objectives and *HV*, $\widehat{A}_{12}$ is used to compare the probability of yielding a higher fitness value for two algorithms A and B. If $\widehat{A}_{12}$ is equal to 0.5, the two algorithms are equivalent. If $\widehat{A}_{12}$ is greater than 0.5, it means the first algorithm A has higher chances of obtaining a higher value than B.

Moreover, to address RQ2 and RQ3, we choose the Spearman's rank correlation coefficient (ρ) [20] to measure the correlation between metrics (objectives and *HV*) with mode *c* of the triangular distribution (RQ2) and with the number of requirements (RQ3), respectively. The value of ρ ranges from -1 to 1, i.e., there is a positive correlation if ρ is equal to 1 and a negative correlation when ρ is -1. A ρ close to 0 shows that there is no correlation between the two sets of data. We also report the significance of correlation using Prob>|ρ|, a value lower than 0.05 means significant correlation.

C. Experiment execution

Each algorithm was run for 100 times (25000 generations each time) to account for random variation and we collected the Pareto front of the last generation. A Pareto front contains optimal solutions that are non-dominated to each other with respect to the objectives (Section II.C). Since we ran each algorithm 100 times, we have 100 Pareto fronts that were used as the input for statistical analyses (Section III.B). We used a PC with Intel Core i7-3630 QM CPU 2.4 GHz with 8GB of RAM, running Windows 7 operating system.

IV. Results and Analysis

A. Results for the RALIC Dataset

In this section, we only report results for RQ1 and RQ2 since we only have one set of requirements in RALIC. RQ3 will be answered only for the artificial problems.

1) *RQ1:* We compared the four objectives (*IMP*, *DEP*, *COST*, and *PCO*) and *HV* of NSGA-II and RS with the Wilcoxon signed-rank test and the Vargha and Delaney test with different mode *c* values of the triangular distribution (TABLE I).

From TABLE I, we can observe that NSGA-II performed significantly better than RS in terms of the four maximizing objectives (*IMP*, *DEP*, *COST*, and *PCO*) and *HV*, since $\widehat{A}_{12}$ values are greater than 0.5 and *p* values are less than 0.05. Thus, we can conclude that NSGA-II achieved significantly better performance than RS for the RALIC Dataset in terms

TABLE I RESULTS FOR THE WILCOXON SIGNED-RANK TEST AND THE VARGHA AND DELANEY TEST AT THE SIGNIFICANCE LEVEL OF 0.05 FOR THE RALIC DATASET

Pair of algorithms	Mode *c*	*IMP*		*DEP*		*COST*		*PCO*		*HV*	
		A	p	A	p	A	p	A	p	A	p
NSGA-II vs. RS	*a*	0.61	**<0.05**	0.61	**<0.05**	0.52	**<0.05**	0.91	**<0.05**	0.58	**<0.05**
	a+1/4(b-a)*	0.58	**<0.05**	0.67	**<0.05**	0.59	**<0.05**	0.94	**<0.05**	0.75	**<0.05**
	a+1/2(b-a)*	0.59	**<0.05**	0.71	**<0.05**	0.62	**<0.05**	0.94	**<0.05**	0.84	**<0.05**
	a+3/4(b-a)*	0.59	**<0.05**	0.74	**<0.05**	0.63	**<0.05**	0.96	**<0.05**	0.90	**<0.05**
	b	0.64	**<0.05**	0.79	**<0.05**	0.68	**<0.05**	0.95	**<0.05**	0.89	**<0.05**

*A: $\hat{A}_{12}$, p: *p*-value. All *p*-values less than 0.05 are identified as bold.

of the metrics (four objectives and HV).

To illustrate how fast NSGA-II prioritizes these requirements, we randomly selected a solution from all generated solution for each algorithm (i.e., NSGA-II and RS) to calculate $AIPOFV_{fp}$. We also plotted $AIPOFV_{fp}$ (defined in Formula (10) in Section III.A) in terms of prioritizing *fp*% of the total requirements, where *fp* = 10, 20, …100 (Fig. 1). From Fig. 1, we can observe that NSGA-II can prioritize requirements as quickly as possible since NSGA-II achieved the highest value of *AIPOFV*, e.g., the first 10% of the requirements in the prioritization sequence have the highest prioritization effectiveness, i.e., *AIPOFV* as compared to RS. Thus, we can conclude that NSGA-II can very quickly prioritize requirements by placing the highest priority requirements earlier in the prioritization sequence.

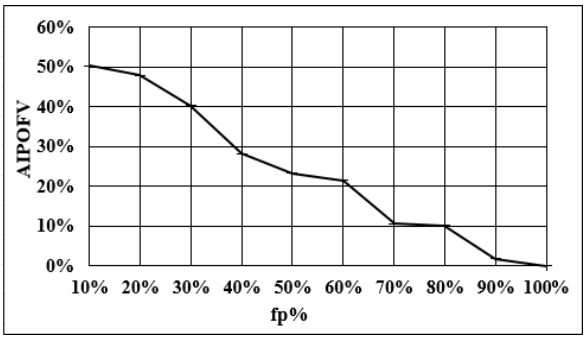


Fig. 1. Average Improvements of OFV for different percentages of the requirements in the prioritization sequence for the RALIC Dataset

2) *RQ2:* We compared the four objectives (*IMP*, *DEP*, *COST*, and *PCO*) and *HV* of NSGA-II under the five different values of mode *c* with the Wilcoxon signed-rank test and the Vargha and Delaney test (Table II).

In terms of *IMP*, when mode *c* value is *a*, NSGA-II performed significantly worse than the cases when it takes the other four values. No significant differences can be observed in NSGA-II's performance when mode *c* takes values other than *a*.

In terms of *DEP*, when mode *c* value is *a*, NSGA-II achieved the best performance, followed by the case when mode *c* takes value *a+1/4*(b-a)*. No significant differences among the other three values (i.e., *a+1/2*(b-a)*, *a+3/4*(b-a)* and *b*) can be observed as shown in Table II.

In terms of *COST*, when mode *c* value is *a*, NSGA-II achieved significantly worse performance than the other three values. No significant differences can be observed among these three mode *c* values, while NSGA-II achieved the best performance when mode *c* value is *b*.

In terms of *PCO*, when mode *c* value is *b*, NSGA-II achieved better but not significantly better performance than the value *a+3/4*(b-a)*, and achieved significantly better than mode c value being the other three values (i.e., *a*, *a+1/4*(b-a)*, and *a+1/2*(b-a)*). NSGA-II achieved no significantly different performance among these four values ((i.e., *a*, *a+1/4*(b-a)*, *a+1/2*(b-a)* and *a+3/4*(b-a)*). Thus, we can conclude that there is no significant difference when mode *c* takes different values. In terms of *HV*, NSGA-II achieved the best performance when mode *c* takes the value *a* and there is no significant difference among the other four values.

The mode *c* controls simulated cost values of a requirement, which consequently decides the probability of cost overrun, while requirements importance, dependency, and implementation cost remain unchanged. When mode *c* takes a smaller value, simulated scenarios are more optimistic and therefore the probability of requirements cost overrun is underestimated to a greater extent. Table III shows the distribution of probability of cost overrun for 143 requirements in the RALIC set under different simulation scenarios (the mode *c* value being *a*, *a+1/4*(b-a)*, *a+1/2*(b-a)*, *a+3/4*(b-a)* or *b*). From Table III, we can see that with a higher value for mode *c*, requirements have a higher mean value (*Mean* in Table III) and a smaller standard deviation for cost overrun probability (*Std Dev* in Table III). A smaller standard deviation indicates that the requirements cost overrun probabilities are clustered closely around the mean value. Based on Formula (7) of Section II.C, higher values of the cost overrun probability lead to both the increase in the numerator and the denominator in

TABLE II. RESULTS FOR THE WILCOXON SIGNED-RANK TEST AND THE VARGHA AND DELANEY TEST BETWEEN DIFFERENT MODE C VALUES AT THE SIGNIFICANCE LEVEL OF 0.05 FOR THE RALIC DATASET

Pair of Mode *c*		*IMP*		*DEP*		*COST*		*PCO*		*HV*	
		A	p	A	p	A	p	A	p	A	p
a	*a+1/4*(b-a)*	0.39	**<0.05**	0.71	**<0.05**	0.44	**<0.05**	0.48	0.72	0.73	**<0.05**
	a+1/2(b-a)*	0.39	**<0.05**	0.79	**<0.05**	0.39	**<0.05**	0.50	0.98	0.68	**<0.05**
	a+3/4(b-a)*	0.34	**<0.05**	0.79	**<0.05**	0.42	**<0.05**	0.46	0.07	0.78	**<0.05**
	b	0.38	**<0.05**	0.72	**<0.05**	0.33	**<0.05**	0.42	**<0.05**	0.69	**<0.05**
a+1/4(b-a)*	*a+1/2*(b-a)*	0.50	0.98	0.61	**<0.05**	0.44	0.17	0.52	0.64	0.48	0.61
	a+3/4(b-a)*	0.45	0.08	0.64	**<0.05**	0.48	0.93	0.48	0.44	0.55	0.32
	b	0.49	0.28	0.56	0.66	0.36	**<0.05**	0.44	**<0.05**	0.50	0.62
a+1/2(b-a)*	*a+3/4*(b-a)*	0.45	0.16	0.55	0.35	0.53	0.17	0.46	0.26	0.57	0.50
	b	0.49	0.58	0.47	0.15	0.41	0.15	0.42	**<0.05**	0.52	0.77
a+3/4(b-a)*	*b*	0.54	0.72	0.43	**<0.05**	0.40	**<0.05**	0.45	0.07	0.46	0.11

*A: $\hat{A}_{12}$, p: p-value. All p-values less than 0.05 are identified as bold.

Formula (7), but the numerator increases less than the denominator. Meanwhile, a smaller standard deviation of cost overrun probability means different solutions of requirement prioritization have little differences in terms of *PCO*. This is the reason that NSGA-II didn't achieve significantly different performance when mode *c* takes these five values, i.e., *a*, *a+1/4*(b-a)*, *a+1/2*(b-a)*, *a+3/4*(b-a)*, and *b*.

TABLE III DISTRIBUTION OF THE PROBABILITY OF REQUIREMENTS COST OVERRUN

Mode c	Probability of Cost Overrun			
	Max	Min	Mean	Std Dev
a	0.6519	0.628	0.6401	0.0043
a+1/4(b-a)*	0.8093	0.7912	0.7998	0.0038
a+1/2(b-a)*	0.9275	0.912	0.9201	0.0029
a+3/4(b-a)*	0.9469	0.9368	0.9426	0.0022
b	0.9641	0.9535	0.9601	0.0019

While importance, dependency, and implementation cost of requirements are not changed, the requirements have a higher probability of cost overrun and a small variation in cost overrun probability, NSGA-II tends to find solutions with better performance in *IMP*, *COST*, and *PCO*, but worse performance in *DEP*. Since *HV* reflects the convergence and diversity of a Pareto set and a larger standard deviation of cost overrun probability under mode *c* taking the value *a* has a strong role in guiding search when requirements importance, dependency, and implementation cost are not changed, NSGA-II achieved a better performance in the same generations (25,000 generations) under mode *c* taking the value *a* than the cases when mode *c* takes the other four values when taking *HV* into account.

In addition, we calculated the Spearman's rank correlation of *IMP*, *DEP*, *COST*, *PCO* and *HV* with mode *c* (Table V). We can observe the same result with Table II, that is, the performance of *IMP*, *COST*, and *PCO* increases with the increase of the value of mode *c*, since the Spearman's ρ value is negative, while the performance of DEP and HV decreases with the increase of the value of mode *c*. However, *IMP*, *COST*, and *PCO* have very weak correlations with mode *c*, since absolute ρ values are less than 0.2. In terms of *DEP* and *HV*, ρ values are greater than 0.2 but less than 0.3. Therefore, we can observe relatively stronger correlations between *DEP* and mode *c*, and *HV* and mode *c*. Note that as discussed in [21, 22], the strength of the correlation of the value between 0 and 0.2 means a very weak correlation and between 0.2 and 0.4 means a weak correlation. That is, although when mode *c* takes different values, NSGA-II achieved different performance in terms of *IMP*, *DEP*, *COST*, *PCO* and *HV* (Table II), there is no observable impact of mode *c* values on *IMP*, *DEP*, *COST*, *PCO* and *HV* based on the spearman' rank correlation.

TABLE V RESULTS OF THE SPEARMAN'S CORRELATION ANALYSIS OF METRICS WITH MODE C OF THE TRIANGULAR DISTRIBUTION –THE RALIC DATASET

Algorithm	Metrics	mode *c*	
		Spearman's ρ	Prob>|ρ|
NSGA-II	*IMP*	0.126	**<.0001**
	DEP	-0.249	**<.0001**
	COST	0.164	**<.0001**
	PCO	0.163	**<.0001**
	HV	-0.229	**<.0001**

* Prob>| ρ| values lower than 0.05 are identified as bold.

B. Results for the Artificial Problems

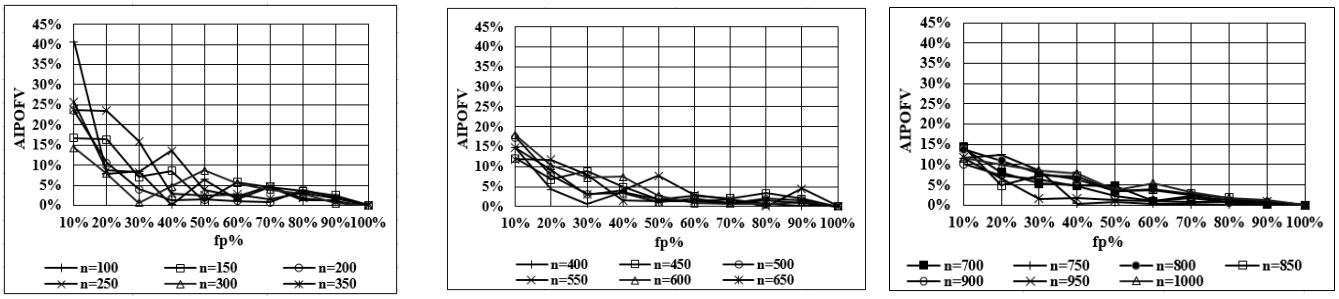
1) *RQ1:* We developed 19 artificial problems with the number of requirements varying from 100 to 1000 with an increment of 50. To answer RQ1, we conducted the Wilcoxon signed-rank test and the Vargha and Delaney test to compare NSGA-II and RS in terms of *IMP*, *DEP*, *COST*, *PCO*, and *HV* for each problem. All the detailed results of each problem are provided in Apendix A. We summarize the results of the Vargha and Delaney statistic test (with or without the Wilcoxon signed-rank test) for the 19 artificial problems in Table IV. From Table IV, we can observe that NSGA-II significantly outperformed RS for all the 19 problems, in terms of the four objectives and *HV*. This implies that our prioritization problem is complex and further justifies the use of search algorithms.

To illustrate how fast NSGA-II prioritizes these requirements, we also plotted $AIPOFV_{fp}$ in terms of prioritizing *fp*% of the total requirements, where *fp* = 10, 20, …100, for 19 artificial problems (Fig. 2). From Fig. 2, we can observe that NSGA-II can prioritize requirements very quickly, as compared to RS in terms of prioritizing highest priority requirements earlier in the prioritization sequence.

2) *RQ2:* To answer RQ2, we compared the four objectives (*IMP*, *DEP*, *COST*, and *PCO*) and *HV* of NSGA-II with different values of mode *c* using the Wilcoxon signed-rank test and the Vargha and Delaney test for 19 problems (Apendix B) and the results are summarized in Table VI. From Table VI, we can observe that, in term of *IMP*, *COST*, and *PCO*, NSGA-II achieved the worst performance when mode *c* takes the value *a*. As discussed in Section IV.A, mode

TABLE IV. RESULTS OF THE VARGHA AND DELANEY STATISTICAL TEST BETWEEN DIFFERENT ALGORITHMS– 19 ARTIFICIAL PROBLEMS (WITHOUT/WITH THE WILCOXON SIGNED-RANK TEST)

Pair of Algorithms	**Mode *c***	NSGA-II VS. RS														
		IMP			*DEP*			*COST*			*PCO*			*HV*		
		A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B
NSGA-II vs.RS	*a*	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0
	a+1/4(b-a)*	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0
	a+1/2(b-a)*	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0
	a+3/4(b-a)*	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0
	b	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0	19/19	0/0	0/0



a) Artificial Problems with Requirements from 100 to 350

b) Artificial Problems with Requirements from 400 to 650

c) Artificial Problems with Requirements from 700 to 1000

Fig. 2. Average Improvements of OFV with different percentages of the requirements in the prioritization sequences for the 19 artificial problems

TABLE VI. RESULTS OF THE VARGHA AND DELANEY STATISTICAL TEST BETWEEN DIFFERENT MODE C VALUES FOR NSGA-II – THE 19 ARTIFICIAL PROBLEMS (WITHOUT/WITH THE WILCOXON SIGNED-RANK TEST)

Pair of Mode *c*		*IMP*			*DEP*			*COST*			*PCO*			*HV*		
		A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B	A>B	A<B	A=B
a	*a+1/4*(b-a)*	0/0	19/17	0/2	15/14	4/4	0/1	1/0	18/17	0/2	0/0	19/18	0/1	18/12	1/1	0/6
	a+1/2(b-a)*	0/0	19/19	0/0	15/13	4/4	0/2	0/0	19/19	0/0	0/0	19/19	0/0	17/11	2/1	0/7
	a+3/4(b-a)*	0/0	19/19	0/0	14/13	5/5	0/1	0/0	19/19	0/0	0/0	19/19	0/0	17/11	2/0	0/8
	b	0/0	19/19	0/0	15/14	4/4	0/1	0/0	19/19	0/0	0/0	19/19	0/0	16/10	3/2	0/7
a+1/4(b-a)*	*a+1/2*(b-a)*	2/1	17/17	0/1	13/11	6/3	0/5	2/0	17/15	0/4	1/1	18/16	0/2	14/2	5/0	0/17
	a+3/4(b-a)*	3/2	16/16	0/1	16/14	3/3	0/2	0/0	19/16	0/3	1/1	18/17	0/1	14/4	5/0	0/15
	b	2/0	17/16	0/3	14/12	5/5	0/2	0/0	19/18	0/1	0/0	19/18	0/1	14/5	5/1	0/13
a+1/2(b-a)*	*a+3/4*(b-a)*	4/2	15/12	0/5	15/12	4/4	0/3	5/3	14/12	0/4	6/4	13/11	0/4	16/2	3/1	0/16
	b	4/1	15/11	0/7	13/12	6/6	0/1	1/1	18/17	0/1	1/1	18/15	0/3	11/5	8/0	0/14
a+3/4(b-a)*	*b*	8/4	11/8	0/7	11/9	8/8	0/2	8/4	11/9	0/6	5/4	14/11	0/4	10/2	9/1	0/16

c controls simulated cost values of a requirement, which therefore decide the cost overrun probability. When mode *c* is *a*, simulated scenarios are with a smaller cost overrun probability and higher variation in the cost overrun probability. When looking into the other four values of mode c (*a+1/4*(b-a)*, *a+1/2*(b-a)*, *a+3/4*(b-a)*) and b), in terms of *IMP*, *COST* and *PCO*, NSGA-II achieved a better performance when mode *c* takes the value *a+1/4*(b-a)*, orderly followed by *a+1/2*(b-a)*, *a+3/4*(b-a)* and *b*. In terms of *DEP*, NSGA-II achieved the best performance when mode *c* takes the value of *a*, followed by *a+1/4*(b-a)*, *a+1/2*(b-a)*, *a+1/4*(b-a)* and *b*. In terms of *HV*, the performance of NSGA-II achieved the significantly better performance when the mode *c* value is *a* than the cases when the mode *c* value is from the other four values. For these four values the results are: 1) 12 out of 19 problems for *a+1/4*(b-a)*, 2) 11 out of 19 problems for *a+1/2*(b-a)* or *a+3/4*(b-a)*, 3) 10 out of 19 problems for *b*. NSGA-II has no significant differences for 17 out of 19 problems when mode *c* takes the value *a+1/4*(b-a)* or *a+1/2*(b-a)*, for 16 out of 19 problems when mode *c* is *a+1/2*(b-a)* or *a+3/4*(b-a)*, and for 16 out of 19 problems when mode *c* is *a+3/4*(b-a)* or *b*. In conclusion, NSGA-II achieved a significantly better performance in terms of *HV* when mode *c* is *a* than the other four values and there is no significant difference in *HV* when mode *c* is one of these four values (*a+1/4*(b-a)*, *a+1/2*(b-a)*, *a+1/4*(b-a)* and *b*).

Though results in Table VI show that with different mode *c* values, NSGA-II had different performance, we can observe from the results of Vargha and Delaney statistics for each problem that $\hat{A}_{12}$ values are close to 0.5, implying that, although the results are statistically significant, the difference with mode *c* being different values is not large. To further investigate the relationship between mode *c* and the performance of NSGA-II, we also calculated the Spearman's rank correlation of metrics (*IMP*, *DEP*, *COST*, *PCO*, and *HV*) with mode *c* for all solutions of the 19 artificial problems (left part of Table VII). From Table VII, we can observe similar results as the ones we observed with the RALIC dataset: the absolute ρ values are very close to 0 (less than 0.1 for *IMP*, *COST*, *PCO* and *HV*, and less than 0.12 for *DEP*). This means that a very weak correlation between the metrics and mode *c* was identified. This conforms to what we discussed in Section III.A: uncertainty in requirements implementation cost is underestimated in highly optimistic scenarios (mode *c* with the value *a*) and the probability of cost overrun for each requirement is lower. With the increase in the mode *c*, the probability

TABLE VII SPEARMAN'S CORRELATION ANALYSIS OF METRICS WITH MODE C OF THE TRIANGULAR DISTRIBUTION – THE 19 ARTIFICIAL PROBLEMS

Algorithm	Metric	mode *c*		the number of requirements	
		Spearman's ρ	Prob>|ρ|	Spearman's ρ	Prob>|ρ|
NSGA-II	*IMP*	0.089	**<.0001**	-0.475	**<.0001**
	DEP	-0.117	**<.0001**	-0.533	**<.0001**
	COST	0.074	**<.0001**	-0.549	**<.0001**
	PCO	0.086	**<.0001**	-0.651	**<.0001**
	HV	-0.054	**<.0001**	-0.921	**<.0001**

* Prob>| ρ| values lower than 0.05 are identified as bold.

of cost overrun for each requirement increases accordingly, while other attributes (i.e., *importance*, *dependency*, and *implementation cost*) of requirements remain unchanged. Thus, we can conclude that *NSGA-II* is not impacted by the parameters of the triangular distribution.

3) *RQ3:* To answer RQ3, we calculated the Spearman's rank correlation of the metrics (*IMP*, *DEP*, *COST*, *PCO*, and *HV*) with the number of requirements for all the solutions of the 19 artificial problems (right part of Table VII). As discussed in [22], the strength of the correlation for absolute values in [0.4, 0.59] means a moderate correlation, while that in [0.6, 0.79] means a strong correlation, and absolute values more than 0.8 mean a very strong correlation. For objectives (*IMP*, *DEP*, and *COST*), absolute ρ values in [0.4, 0.59] means that the correlation between these three objectives and the number of requirements is moderate. In terms of *PCO*, absolute ρ values greater than 0.6 mean the correlation between *PCO* and the number of requirements is strong. In terms of *HV*, absolute ρ values greater than 0.9 mean the correlation between *HV* and the number of requirements is very strong. Furthermore, ρ values between the number of requirements and all metrics are negative, which means that the performance of NSGA-II for these metrics decreases with the increase in the number of requirements.

Since the number of requirements n represents the number of decision variables and $n!$ represents the size of search space, the complexity of our search problem increases with the increase in the number of requirements. Taking *HV* into account, it decreases with the increase in the number of requirements. That is, the performance of NSGA-II decreases with the increase in the number of requirements taking convergence and diversity into account (*HV*).

C. Discussion

Based on the results for the RALIC data set and 19 artificial problems presented in Section IV.A and Section IV.B, we can conclude that: 1) NSGA-II achieved a significantly better performance than RS and can prioritize highest priority requirements earlier in the prioritization sequence; 2) metrics are independent of the mode *c* values of the triangular distribution. There is a weak correlation between mode *c* of triangular distribution and the metrics and mode *c* has little impact on the performance of NSGAII, and 3) the performance of NSGA-II (four objectives and *HV*) decreases with the increase in the number of requirements.

In our prioritization problem, we used requirements attributes such as importance, dependency, implementation cost, and the probability of cost overrun to prioritize requirements. There are different ways to obtain values of these attributes. For example, requirements are often classified into safety, functional, non-functional, or optional requirements. Based on such classification, the *Importance* value for each requirement can be calculated. Another way of obtaining *Importance* values is by asking stakeholders to provide importance as scores to requirements. Such a way is used in the RALIC dataset [13]. In terms of dependency, requirements engineers can analyze requirements and calculate the number of requirements that particular requirement is dependent on.

Our approach can be applied in various situations. First, our approach can be used in the context that requirements' implementation costs are provided by requirements engineers with an estimated value: the triangular distribution is used in MCS and the limit range of actual cost is [0.25,4] times of the estimated implementation cost based on the conclusion in [6]. Second, if historical data of real scenarios of a requirement implementation cost is available, we can extract the cost implementation distribution from these historical data. We can still use MCS and the extracted cost probability distribution is used in MCS to generate the probability of cost overrun since MCS can successfully simulate a large number of scenarios with future requirements implementation costs.

Our approach can be extended to other uncertainty-wise requirement optimization problems including uncertainty in other requirements attributes, such as the time required to implement a particular requirement. Based on our approach, MCS can also be used with a different probability distribution related to the uncertain attribute to simulate a large number of attribute values. In the future, we plan to conduct additional experiments to test different probability distributions.

V. Threats to Validity

Our key threat to construct validity is related to simulating the uncertainty of requirements implementation cost. We used MCS to simulate uncertainty on requirements implementation cost based on its probability distribution, and then the probability of cost overrun is calculated. Note that in the literature, MCS has been successfully used to simulate a large number of scenarios with future requirements implementation costs [6, 7] when simulation scenarios are up to 10,000.

Internal validity threats are related to internal factors of experiments that may impact the outcomes. A possible threat to internal validity is that we have experimented with only one configuration setting for the parameters of the algorithm, however, default configurations have proven to provide good results [23]. Another threat to internal validity is to use the triangular distribution to simulate uncertainty of requirements implementation cost. There are three parameters of the triangular distribution, the lower limit a, the upper limit b and mode c. The values of a and b are derived from the literature of requirements implementation cost [6]. We only investigated five different values of mode *c*. However, results of our experiments show that different values of mode c have no significant impact on the performance of NSGA-II.

In terms of external validity threat, we ran experiments on just one real case study with 143 requirements with 31 dependency relations, in addition to the 19 artificial problems of varying complexity. Undoubtedly, more case studies are still needed. The most probable conclusion validity threat in experiments with randomized algorithms is due to random variations. To address it, we repeated experiments 100 times to reduce the possibility that the results were obtained by chance.

VI. Related Work

Search Based Software Engineering (SBSE) techniques have already been applied to many problems throughout the

software engineering lifecycle [24]. Harman also explains some of the ways how SBSE was applied to the construction of predictive models in [25]. SBSE has been applied for supporting various requirements engineering activities including requirements selection and optimization (e.g.,[26, 27]), assignment (e.g., [28]) and prioritization (e.g., [2, 3]). In the field of requirements prioritization, different criteria have been considered such as business value, implementation cost, risk, and penalty. Researchers performed a systematic literature to identify and categorize prioritization criteria [29]. In our previous work, we proposed a search-based methodology for assigning requirements to various stakeholders to review by maximizing their familiarity to assigned requirements, meanwhile balancing the overall workload of each stakeholder [4]. Based on the work in [4], a cost-effective requirements assignment methodology was proposed and presented in [30]. But the uncertainty of requirements implementation cost is however not taken into account.

Uncertainty can hinder organizations in making strategic decisions due to, e.g., uncertain stakeholders' goals and priorities [31]. Here, uncertainty is defined as the lack of knowledge of the consequences of decision alternatives [31]. Uncertainty in requirements has been studied in the context of dynamical adaptive systems in the presence of environmental uncertainty [32, 33]. Researchers viewed self-adaptive systems as runtime entities that can be reasoned to understand the extent to which they are being satisfied and to support adapting decisions that can take advantage of the systems' self-adaptive machinery [34].

Other works focus on modeling and specifying uncertainty in requirements. Salay et al. [35] proposed annotations for modeling uncertainty in requirements models. Famelis and Santosa [36] proposed to use colored Entity-Relation models for explicitly capturing the MAVO partiality, as well as Points of Uncertainty, a concept representing a specific decision about which there is uncertainty. RELAX is a formal requirements specification language that relies on Fuzzy Branching Temporal Logic to specify the uncertain requirements in the self-adaptive system [37]. In [31], the U-RUCM methodology for explicitly specifying uncertainty as part of use case models was proposed, which is based on the general Restricted Use Case Modeling (RUCM) methodology [38, 39].

Some researchers work on the uncertainty in the areas of requirements optimization. Probabilistic sensitivity analysis is used to evaluate the impact of uncertainties in operational release planning and product release [40]. Letier et al. [7] proposed a method allowing software architects to describe uncertainty about the impact of alternatives on stakeholders' goals and to calculate the consequences of uncertainty through MCS. Li et al. [9] adopted a search-based optimization technique with MCS to address uncertainty and risk in the early stages of the software engineering and made explicit the trade-off between uncertainty/risk and traditional attributes of cost and revenue. Moreover, Li et al. [6] developed a decision support framework for NRP to manage algorithmic uncertainty and requirements uncertainty. Voola and Babu introduced requirements uncertainty into the basic prioritization technique Numerical Assignment by means of probability distribution [41] and assigned a rank to a requirement. In this paper, we, however, take uncertainty in requirement implementation cost into consideration when prioritizing requirements to be reviewed.

VII. Conclusion and Future Work

This paper introduced uncertainty into the requirement prioritization and formulated four objectives for uncertainty-wise requirements prioritization. The Monte-Carlo Simulation was used to simulate a large number of scenarios and probability of requirements implementation cost overrun was calculated based on these simulated scenarios. We empirically evaluated NSGA-II together with RS using the RALIC dataset and 19 artificial problems in terms of solving our problem. Results showed that NSGA-II can solve the requirements prioritization problem with a significantly better performance than RS, and can prioritize highest priority requirements earlier in a prioritization sequence. The performance of the algorithms is independent of the parameter settings of the triangular distribution used in MCS. The performance of both NSGA-II and RS, when measured with hypervolume indicator (*HV*), decreases with the increase in the number of requirements. In the future, we will evaluate our approach with more case studies and different strategies for simulating uncertainties. We will also in the future consider investigating other uncertainty-wise requirement optimization problems.

References

[1] C. Wohlin: Engineering and Managing Software Requirements. Springer Science & Business Media, 2005.

[2] P. Baker, M. Harman, K. Steinhöfel, A. Skaliotis: Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. 22nd IEEE International Conference on Software Maintenance, Philadelphia, Pennsylvania, September 24-27, 2006, pp. 176-185

[3] A. Herrmann, M. Daneva: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. 16th IEEE International Requirements Engineering, 2008, pp. 125-134

[4] Y. Li, T. Yue, S. Ali, L. Zhang: Zen-Reqoptimizer: A Search-Based Approach for Requirements Assignment Optimization. Empirical Software Engineering, 2016, 22 (1), pp. 175-234

[5] H. Ziv, D. Richardson, R. Klösch: The Uncertainty Principle in Software Engineering. submitted to Proceeding of the 19th International Conference on Software Engineering

[6] L. Li, M. Harman, F. Wu, Y. Zhang: The Value of Exact Analysis in Requirements Selection. IEEE Transactions on Software Engineering, 2016, 1 (99), pp. 1-18

[7] E. Letier, D. Stefan, E.T. Barr: Uncertainty, Risk, and Information Value in Software Requirements and Architecture. Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 883-894

[8] P. Berander, A. Andrews: Requirements Prioritization. in: Engineering and Managing Software Requirements: Springer, 2005, pp. 69-94.

[9] L. Li, M. Harman, E. Letier, Y. Zhang: Robust Next Release Problem: Handling Uncertainty During Optimization. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 1247-1254

[10] D. Johnson: The Triangular Distribution as a Proxy for the Beta Distribution in Risk Analysis. Journal of the Royal Statistical Society: Series D (The Statistician), 1997, 46 (3), pp. 387-398

[11] Ralic Dataset:https://soolinglim.wordpress.com/datasets/#ralic.

[12] J. Karlsson: Software Requirements Prioritizing. Proceedings of the Second International Conference on Requirements Engineering, 1996, pp. 110-116

[13] S.L. Lim: Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. in: University of New South Wales, 2010.

[14] ISO, IEC: IEEE 29148: 2011-Systems and Software Engineering-Requirements Engineering

[15] J. Li, L. Zhu, R. Jeffery, Y. Liu, H. Zhang, Q. Wang, M. Li: An Initial Evaluation of Requirements Dependency Types in Change Propagation Analysis. Proc. Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on, Year (accepted)

[16] J.J. Durillo, A.J. Nebro: Jmetal: A Java Framework for Multi-Objective Optimization. Advances in Engineering Software, 2011, 42 (10), pp. 760-771

[17] R.B. Agrawal, K. Deb, R. Agrawal: Simulated Binary Crossover for Continuous Search Space. Complex systems, 1995, 9 (2), pp. 115-148

[18] H. Li, Q. Zhang: Multiobjective Optimization Problems with Complicated Pareto Sets, Moea/D and Nsga-Ii. IEEE Transactions on Evolutionary Computation, 2009, 13 (2), pp. 284-302

[19] A. Arcuri: It Really Does Matter How You Normalize the Branch Distance in Search-Based Software Testing. Software Testing, Verification and Reliability, 2013, 23 (2), pp. 119-147

[20] D.J. Sheskin: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC, 2007.

[21] M.W. Lipsey, D.B. Wilson: Practical Meta-Analysis. Sage publications Thousand Oaks, CA, 2001.

[22] Statistical Education through Problem Solving:http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf.

[23] A. Arcuri, L. Briand.: A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. International Conference on Software Engineering (ICSE), 2011, pp. 1-10

[24] M. Harman: Why the Virtual Nature of Software Makes It Ideal for Search Based Optimization. in: D. Rosenblum, G. Taentzer (Eds.) Fundamental Approaches to Software Engineering: Springer Berlin Heidelberg, 2010, pp. 1-12.

[25] M. Harman: The Relationship between Search Based Software Engineering and Predictive Modeling. Proceedings of the 6th International Conference on Predictive Models in Software Engineering, Timi\şoara, Romania, 2010, pp. 1-13

[26] Y. Zhang, M. Harman, S.L. Lim: Empirical Evaluation of Search Based Requirements Interaction Management. Information and Software Technology, 2013, 55 (1), pp. 126-152

[27] A.M. Pitangueira, R.S.P. Maciel, M. Barros: Software Requirements Selection and Prioritization Using Sbse Approaches: A Systematic Review and Mapping of the Literature. Journal of Systems and Software, 2014, 103 (C), pp. 267-280

[28] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, Y. Zhang: A Search Based Approach to Fairness Analysis in Requirement Assignments to Aid Negotiation, Mediation and Decision Making. Requirements Engineering, 2009, 14 (4), pp. 231-245

[29] N. Riegel, J. Doerr: A Systematic Literature Review of Requirements Prioritization Criteria. International Working Conference on Requirements Engineering: Foundation for Software Quality, 2015, pp. 300-317

[30] Y. Li, T. Yue, S. Ali, L. Zhang: A Multi-Objective and Cost-Aware Optimization of Requirements Assignment for Review. Proc. IEEE Congress on Evolutionary Computation 2017, Donostia - San Sebastián, Spain, Year (accepted)

[31] M. Zhang, T. Yue, S. Ali, B. Selic, O. Okariz, R. Norgren, K. Intxausti, S. Charramendieta: Specifying Uncertainty in Use Case Models in Industrial Settings.Simula Research Laboratory.Technical Report

[32] B.H. Cheng, P. Sawyer, N. Bencomo, J. Whittle: A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty. International Conference on Model Driven Engineering Languages and Systems, 2009, pp. 468-483

[33] J. Whittle, P. Sawyer, N. Bencomo, B.H. Cheng, J.-M. Bruel: Relax: A Language to Address Uncertainty in Self-Adaptive Systems Requirement. Requirements Engineering, 2010, 15 (2), pp. 177-196

[34] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, A. Finkelstein: Requirements-Aware Systems: A Research Agenda for Re for Self-Adaptive Systems. Proc. Requirements Engineering Conference (RE), 2010 18th IEEE International, Year (accepted)

[35] R. Salay, M. Chechik, J. Horkoff, A. Di Sandro: Managing Requirements Uncertainty with Partial Models. Requirements Engineering, 2013, 18 (2), pp. 107-128

[36] M. Famelis, S. Santosa: Mav-Vis: A Notation for Model Uncertainty. 5th International Workshop on Modeling in Software Engineering (MiSE), 2013, pp. 7-12

[37] N. Esfahani, S. Malek: Uncertainty in Self-Adaptive Software Systems. Springer Berlin Heidelberg, 2013.

[38] G. Zhang, T. Yue, J. Wu, S. Ali: Zen-Rucm: A Tool for Supporting a Comprehensive and Extensible Use Case Modeling Framework. Demos/Posters/StudentResearch@ MoDELS, 2013, pp. 41-45

[39] T. Yue, L.C. Briand, Y. Labiche: Facilitating the Transition from Use Case Models to Analysis Models: Approach and Experiments. ACM Transactions on Software Engineering and Methodology (TOSEM), 2013, 22 (1), pp. 5

[40] A. Al-Emran, P. Kapur, D. Pfahl, G. Ruhe: Studying the Impact of Uncertainty in Operational Release Planning – an Integrated Method and Its Initial Evaluation. Information & Software Technology, 2010, 52 (4), pp. 446-461

[41] P. Voola, A.V. Babu: Requirements Uncertainty Prioritization Approach: A Novel Approach for Requirements Prioritization. Software Engineering: An International Journal (SEIJ), 2012, 2, pp. 37-49

Apendix A. RESULTS OF 19 ARTIFICIAL PROBLEMS FOR RQ1

TABLE VIII RESULTS FOR THE WILCOXON SIGNED-RANK TEST AND THE VARGHA AND DELANEY TEST AT THE SIGNIFICANCE LEVEL OF 0.05 BETWEEN NSGA-II AND RS FOR 19 ARTIFICIAL PROBLEMS

Number of requirements	Mode *c*	*IMP*		*DEP*		*COST*		*PCO*		*HV*	
		A	p	A	p	A	p	A	p	A	p
100	*a*	0.86	**<0.05**	0.89	**<0.05**	0.86	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	a+1/4(b-a)*	0.88	**<0.05**	0.94	**<0.05**	0.91	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	a+1/2(b-a)*	0.88	**<0.05**	0.99	**<0.05**	0.93	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.90	**<0.05**	0.98	**<0.05**	0.92	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	b	0.94	**<0.05**	0.99	**<0.05**	0.92	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
150	*a*	0.76	**<0.05**	0.78	**<0.05**	0.81	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+1/4(b-a)*	0.85	**<0.05**	0.92	**<0.05**	0.89	**<0.05**	1.00	**<0.05**	0.99	**<0.05**
	a+1/2(b-a)*	0.90	**<0.05**	0.94	**<0.05**	0.92	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.85	**<0.05**	0.95	**<0.05**	0.90	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	b	0.89	**<0.05**	0.97	**<0.05**	0.92	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
200	*a*	0.70	**<0.05**	0.70	**<0.05**	0.83	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
	a+1/4(b-a)*	0.74	**<0.05**	0.84	**<0.05**	0.84	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+1/2(b-a)*	0.77	**<0.05**	0.86	**<0.05**	0.86	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.75	**<0.05**	0.89	**<0.05**	0.87	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	b	0.83	**<0.05**	0.93	**<0.05**	0.89	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
250	*a*	0.65	**<0.05**	0.72	**<0.05**	0.75	**<0.05**	0.97	**<0.05**	0.98	**<0.05**
	a+1/4(b-a)*	0.68	**<0.05**	0.80	**<0.05**	0.78	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+1/2(b-a)*	0.74	**<0.05**	0.84	**<0.05**	0.81	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.75	**<0.05**	0.87	**<0.05**	0.82	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.80	**<0.05**	0.91	**<0.05**	0.83	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
300	*a*	0.51	**<0.05**	0.60	**<0.05**	0.71	**<0.05**	0.97	**<0.05**	0.80	**<0.05**
	a+1/4(b-a)*	0.54	**<0.05**	0.69	**<0.05**	0.74	**<0.05**	0.98	**<0.05**	0.91	**<0.05**
	a+1/2(b-a)*	0.59	**<0.05**	0.76	**<0.05**	0.78	**<0.05**	0.98	**<0.05**	0.98	**<0.05**
	a+3/4(b-a)*	0.64	**<0.05**	0.84	**<0.05**	0.79	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.63	**<0.05**	0.87	**<0.05**	0.78	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
350	*a*	0.59	**<0.05**	0.72	**<0.05**	0.65	**<0.05**	0.98	**<0.05**	0.91	**<0.05**
	a+1/4(b-a)*	0.61	**<0.05**	0.83	**<0.05**	0.67	**<0.05**	0.99	**<0.05**	0.95	**<0.05**
	a+1/2(b-a)*	0.65	**<0.05**	0.89	**<0.05**	0.68	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+3/4(b-a)*	0.64	**<0.05**	0.86	**<0.05**	0.73	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	b	0.68	**<0.05**	0.88	**<0.05**	0.71	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
400	*a*	0.73	**<0.05**	0.61	**<0.05**	0.68	**<0.05**	0.98	**<0.05**	0.96	**<0.05**
	a+1/4(b-a)*	0.77	**<0.05**	0.73	**<0.05**	0.69	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
	a+1/2(b-a)*	0.78	**<0.05**	0.80	**<0.05**	0.73	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.82	**<0.05**	0.84	**<0.05**	0.70	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.81	**<0.05**	0.87	**<0.05**	0.75	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
450	*a*	0.74	**<0.05**	0.59	**<0.05**	0.65	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	a+1/4(b-a)*	0.80	**<0.05**	0.76	**<0.05**	0.71	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+1/2(b-a)*	0.81	**<0.05**	0.81	**<0.05**	0.70	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+3/4(b-a)*	0.84	**<0.05**	0.87	**<0.05**	0.73	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.80	**<0.05**	0.87	**<0.05**	0.74	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
500	*a*	0.62	**<0.05**	0.61	**<0.05**	0.55	**<0.05**	0.99	**<0.05**	0.72	**<0.05**
	a+1/4(b-a)*	0.66	**<0.05**	0.67	**<0.05**	0.60	**<0.05**	0.99	**<0.05**	0.84	**<0.05**
	a+1/2(b-a)*	0.70	**<0.05**	0.73	**<0.05**	0.60	**<0.05**	0.99	**<0.05**	0.95	**<0.05**
	a+3/4(b-a)*	0.71	**<0.05**	0.80	**<0.05**	0.63	**<0.05**	0.99	**<0.05**	0.96	**<0.05**
	b	0.70	**<0.05**	0.79	**<0.05**	0.64	**<0.05**	1.00	**<0.05**	0.96	**<0.05**
550	*a*	0.65	**<0.05**	0.69	**<0.05**	0.70	**<0.05**	0.99	**<0.05**	0.97	**<0.05**
	a+1/4(b-a)*	0.69	**<0.05**	0.68	**<0.05**	0.70	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	a+1/2(b-a)*	0.70	**<0.05**	0.79	**<0.05**	0.73	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+3/4(b-a)*	0.72	**<0.05**	0.83	**<0.05**	0.74	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.71	**<0.05**	0.88	**<0.05**	0.75	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
600	*a*	0.60	**<0.05**	0.61	**<0.05**	0.58	**<0.05**	0.99	**<0.05**	0.69	**<0.05**
	a+1/4(b-a)*	0.61	**<0.05**	0.67	**<0.05**	0.59	**<0.05**	0.99	**<0.05**	0.85	**<0.05**
	a+1/2(b-a)*	0.66	**<0.05**	0.70	**<0.05**	0.66	**<0.05**	1.00	**<0.05**	0.89	**<0.05**
	a+3/4(b-a)*	0.67	**<0.05**	0.71	**<0.05**	0.64	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	b	0.64	**<0.05**	0.76	**<0.05**	0.68	**<0.05**	0.99	**<0.05**	0.96	**<0.05**
650	*a*	0.57	**<0.05**	0.70	**<0.05**	0.74	**<0.05**	0.98	**<0.05**	0.90	**<0.05**
	a+1/4(b-a)*	0.59	**<0.05**	0.81	**<0.05**	0.78	**<0.05**	0.98	**<0.05**	0.99	**<0.05**
	a+1/2(b-a)*	0.61	**<0.05**	0.86	**<0.05**	0.79	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+3/4(b-a)*	0.63	**<0.05**	0.84	**<0.05**	0.81	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.63	**<0.05**	0.85	**<0.05**	0.80	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
700	*a*	0.60	**<0.05**	0.54	**<0.05**	0.52	**<0.05**	0.98	**<0.05**	0.54	**<0.05**
	a+1/4(b-a)*	0.63	**<0.05**	0.60	**<0.05**	0.53	**<0.05**	0.99	**<0.05**	0.75	**<0.05**
	a+1/2(b-a)*	0.66	**<0.05**	0.68	**<0.05**	0.55	**<0.05**	0.99	**<0.05**	0.92	**<0.05**

	a+3/4(b-a)*	0.69	**<0.05**	0.67	**<0.05**	0.58	**<0.05**	0.99	**<0.05**	0.88	**<0.05**
	b	0.66	**<0.05**	0.78	**<0.05**	0.62	**<0.05**	1.00	**<0.05**	0.91	**<0.05**
750	*a*	0.67	**<0.05**	0.52	**<0.05**	0.59	**<0.05**	0.97	**<0.05**	0.70	**<0.05**
	a+1/4(b-a)*	0.69	**<0.05**	0.60	**<0.05**	0.64	**<0.05**	0.98	**<0.05**	0.87	**<0.05**
	a+1/2(b-a)*	0.68	**<0.05**	0.77	**<0.05**	0.66	**<0.05**	0.98	**<0.05**	0.98	**<0.05**
	a+3/4(b-a)*	0.68	**<0.05**	0.81	**<0.05**	0.67	**<0.05**	0.99	**<0.05**	0.97	**<0.05**
	b	0.70	**<0.05**	0.83	**<0.05**	0.65	**<0.05**	0.98	**<0.05**	0.96	**<0.05**
800	*a*	0.62	**<0.05**	0.69	**<0.05**	0.61	**<0.05**	0.99	**<0.05**	0.83	**<0.05**
	a+1/4(b-a)*	0.62	**<0.05**	0.77	**<0.05**	0.62	**<0.05**	0.99	**<0.05**	0.95	**<0.05**
	a+1/2(b-a)*	0.63	**<0.05**	0.80	**<0.05**	0.66	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	a+3/4(b-a)*	0.65	**<0.05**	0.81	**<0.05**	0.69	**<0.05**	0.99	**<0.05**	0.97	**<0.05**
	b	0.66	**<0.05**	0.86	**<0.05**	0.68	**<0.05**	0.99	**<0.05**	0.97	**<0.05**
850	*a*	0.61	**<0.05**	0.65	**<0.05**	0.51	**<0.05**	0.97	**<0.05**	0.67	**<0.05**
	a+1/4(b-a)*	0.66	**<0.05**	0.78	**<0.05**	0.54	**<0.05**	0.98	**<0.05**	0.88	**<0.05**
	a+1/2(b-a)*	0.66	**<0.05**	0.78	**<0.05**	0.54	**<0.05**	0.98	**<0.05**	0.87	**<0.05**
	a+3/4(b-a)*	0.66	**<0.05**	0.74	**<0.05**	0.56	**<0.05**	0.98	**<0.05**	0.88	**<0.05**
	b	0.66	**<0.05**	0.77	**<0.05**	0.56	**<0.05**	0.98	**<0.05**	0.93	**<0.05**
900	*a*	0.68	**<0.05**	0.70	**<0.05**	0.55	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	a+1/4(b-a)*	0.71	**<0.05**	0.70	**<0.05**	0.57	**<0.05**	0.99	**<0.05**	0.91	**<0.05**
	a+1/2(b-a)*	0.72	**<0.05**	0.82	**<0.05**	0.59	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
	a+3/4(b-a)*	0.76	**<0.05**	0.82	**<0.05**	0.60	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	b	0.75	**<0.05**	0.87	**<0.05**	0.62	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
950	*a*	0.75	**<0.05**	0.62	**<0.05**	0.63	**<0.05**	0.99	**<0.05**	0.94	**<0.05**
	a+1/4(b-a)*	0.76	**<0.05**	0.63	**<0.05**	0.69	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
	a+1/2(b-a)*	0.78	**<0.05**	0.73	**<0.05**	0.69	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+3/4(b-a)*	0.80	**<0.05**	0.77	**<0.05**	0.68	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
	b	0.80	**<0.05**	0.81	**<0.05**	0.70	**<0.05**	0.99	**<0.05**	1.00	**<0.05**
1000	*a*	0.63	**<0.05**	0.70	**<0.05**	0.75	**<0.05**	0.99	**<0.05**	0.98	**<0.05**
	a+1/4(b-a)*	0.66	**<0.05**	0.73	**<0.05**	0.76	**<0.05**	0.99	**<0.05**	0.99	**<0.05**
	a+1/2(b-a)*	0.68	**<0.05**	0.83	**<0.05**	0.79	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	a+3/4(b-a)*	0.70	**<0.05**	0.86	**<0.05**	0.79	**<0.05**	1.00	**<0.05**	1.00	**<0.05**
	b	0.69	**<0.05**	0.89	**<0.05**	0.80	**<0.05**	1.00	**<0.05**	1.00	**<0.05**

*A: $\hat{A}_{12}$, p: *p*-value. All *p*-values less than 0.05 are identified as bold.

Apendix B. RESULTS OF 19 ARTIFICIAL PROBLEMS FOR RQ2

TABLE IX. RESULTS FOR THE WILCOXON SIGNED-RANK TEST AND THE VARGHA AND DELANEY TEST BETWEEN DIFFERENT MODE C VALUES AT THE SIGNIFICANCE LEVEL OF 0.05 FOR THE RALIC DATASET

Number of requirements	Pair of Mode *c*		***IMP***		***DEP***		***COST***		***PCO***		***HV***	
			A	p	A	p	A	p	A	p	A	p
100	*a*	*a+1/4*(b-a)*	0.42	**<0.05**	0.47	**<0.05**	0.42	**<0.05**	0.41	**<0.05**	0.52	0.97
		a+1/2(b-a)*	0.42	**<0.05**	0.48	**<0.05**	0.43	**<0.05**	0.45	**<0.05**	0.49	0.88
		a+3/4(b-a)*	0.40	**<0.05**	0.45	**<0.05**	0.40	**<0.05**	0.40	**<0.05**	0.51	0.86
		b	0.39	**<0.05**	0.45	**<0.05**	0.40	**<0.05**	0.40	**<0.05**	0.46	0.29
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.497	**<0.05**	0.49	1.0	0.51	0.25	0.52	**<0.05**	0.48	1.00
		a+3/4(b-a)*	0.48	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.47	**<0.05**	0.50	0.89
		b	0.46	**<0.05**	0.47	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.45	0.30
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.49	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.51	0.92
		b	0.46	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.44	**<0.05**	0.47	0.33
	a+3/4(b-a)*	*b*	0.46	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.47	**<0.05**	0.45	0.44
150	*a*	*a+1/4*(b-a)*	0.38	**<0.05**	0.40	**<0.05**	0.36	**<0.05**	0.38	**<0.05**	0.43	**<0.05**
		a+1/2(b-a)*	0.33	**<0.05**	0.38	**<0.05**	0.32	**<0.05**	0.34	**<0.05**	0.37	**<0.05**
		a+3/4(b-a)*	0.40	**<0.05**	0.45	**<0.05**	0.36	**<0.05**	0.40	**<0.05**	0.46	0.26
		b	0.30	**<0.05**	0.36	**<0.05**	0.28	**<0.05**	0.29	**<0.05**	0.40	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.44	**<0.05**	0.48	**<0.05**	0.44	**<0.05**	0.45	**<0.05**	0.44	0.13
		a+3/4(b-a)*	0.51	**<0.05**	0.55	**<0.05**	0.48	0.26	0.52	**<0.05**	0.52	0.31
		b	0.41	**<0.05**	0.47	**<0.05**	0.40	**<0.05**	0.42	**<0.05**	0.46	0.50
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.60	**<0.05**	0.59	**<0.05**	0.57	**<0.05**	0.59	**<0.05**	0.59	**<0.05**
		b	0.49	0.08	0.52	**<0.05**	0.48	**<0.05**	0.48	0.73	0.53	0.39
	a+3/4(b-a)*	*b*	0.37	**<0.05**	0.41	**<0.05**	0.40	**<0.05**	0.38	**<0.05**	0.44	0.09
200	*a*	*a+1/4*(b-a)*	0.46	**<0.05**	0.45	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.51	0.20
		a+1/2(b-a)*	0.42	**<0.05**	0.48	**<0.05**	0.42	**<0.05**	0.40	**<0.05**	0.50	0.39
		a+3/4(b-a)*	0.44	**<0.05**	0.505	0.52	0.43	**<0.05**	0.40	**<0.05**	0.51	0.55
		b	0.31	**<0.05**	0.41	**<0.05**	0.32	**<0.05**	0.31	**<0.05**	0.40	<0.05
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.44	**<0.05**	0.52	0.10	0.46	**<0.05**	0.45	**<0.05**	0.51	0.91
		a+3/4(b-a)*	0.47	**<0.05**	0.55	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.52	0.72

		b	0.33	**<0.05**	0.46	**<0.05**	0.36	**<0.05**	0.36	**<0.05**	0.41	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.52	**<0.05**	0.53	**<0.05**	0.51	**<0.05**	0.50	**<0.05**	0.51	0.92
		b	0.37	**<0.05**	0.44	**<0.05**	0.40	**<0.05**	0.41	**<0.05**	0.41	0.06
	a+3/4(b-a)*	*b*	0.35	**<0.05**	0.40	**<0.05**	0.37	**<0.05**	0.38	**<0.05**	0.40	**<0.05**
250	*a*	*a+1/4*(b-a)*	0.43	**<0.05**	0.55	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.58	0.25
		a+1/2(b-a)*	0.37	**<0.05**	0.60	**<0.05**	0.42	**<0.05**	0.43	**<0.05**	0.57	0.35
		a+3/4(b-a)*	0.36	**<0.05**	0.60	**<0.05**	0.38	**<0.05**	0.39	**<0.05**	0.53	0.33
		b	0.30	**<0.05**	0.56	**<0.05**	0.37	**<0.05**	0.34	**<0.05**	0.51	0.48
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.45	**<0.05**	0.56	**<0.05**	0.47	**<0.05**	0.48	**<0.05**	0.50	0.83
		a+3/4(b-a)*	0.43	**<0.05**	0.57	**<0.05**	0.44	**<0.05**	0.45	**<0.05**	0.47	0.07
		b	0.38	**<0.05**	0.52	0.46	0.43	**<0.05**	0.40	**<0.05**	0.48	0.20
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.49	**<0.05**	0.53	**<0.05**	0.48	**<0.05**	0.47	**<0.05**	0.47	0.27
		b	0.43	**<0.05**	0.48	**<0.05**	0.47	**<0.05**	0.43	**<0.05**	0.48	0.55
	a+3/4(b-a)*	*b*	0.45	**<0.05**	0.45	**<0.05**	0.49	0.15	0.45	**<0.05**	0.52	0.58
300	*a*	*a+1/4*(b-a)*	0.43	**<0.05**	0.52	**<0.05**	0.45	**<0.05**	0.44	**<0.05**	0.53	0.85
		a+1/2(b-a)*	0.39	**<0.05**	0.54	**<0.05**	0.40	**<0.05**	0.40	**<0.05**	0.59	0.42
		a+3/4(b-a)*	0.33	**<0.05**	0.48	**<0.05**	0.36	**<0.05**	0.33	**<0.05**	0.46	0.05
		b	0.35	**<0.05**	0.48	**<0.05**	0.39	**<0.05**	0.34	**<0.05**	0.54	0.98
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.45	**<0.05**	0.52	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.56	0.46
		a+3/4(b-a)*	0.38	**<0.05**	0.46	**<0.05**	0.43	**<0.05**	0.38	**<0.05**	0.44	0.10
		b	0.41	**<0.05**	0.46	**<0.05**	0.46	**<0.05**	0.40	**<0.05**	0.52	0.66
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.42	**<0.05**	0.42	**<0.05**	0.45	**<0.05**	0.41	**<0.05**	0.40	**<0.05**
		b	0.45	**<0.05**	0.42	**<0.05**	0.48	**<0.05**	0.43	**<0.05**	0.46	0.39
	a+3/4(b-a)*	*b*	0.53	0.06	0.50	0.08	0.53	**<0.05**	0.52	0.95	0.56	0.10
350	*a*	*a+1/4*(b-a)*	0.47	**<0.05**	0.56	**<0.05**	0.48	**<0.05**	0.45	**<0.05**	0.66	**<0.05**
		a+1/2(b-a)*	0.44	**<0.05**	0.60	**<0.05**	0.45	**<0.05**	0.44	**<0.05**	0.68	**<0.05**
		a+3/4(b-a)*	0.44	**<0.05**	0.66	**<0.05**	0.40	**<0.05**	0.41	**<0.05**	0.72	**<0.05**
		b	0.41	**<0.05**	0.67	**<0.05**	0.41	**<0.05**	0.40	**<0.05**	0.63	0.07
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.55	**<0.05**	0.47	**<0.05**	0.49	0.09	0.52	0.65
		a+3/4(b-a)*	0.48	**<0.05**	0.62	**<0.05**	0.42	**<0.05**	0.46	**<0.05**	0.56	0.16
		b	0.44	**<0.05**	0.66	**<0.05**	0.43	**<0.05**	0.46	**<0.05**	0.51	0.78
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.49	0.58	0.57	**<0.05**	0.46	**<0.05**	0.47	**<0.05**	0.53	0.26
		b	0.46	**<0.05**	0.61	**<0.05**	0.46	**<0.05**	0.46	**<0.05**	0.48	0.62
	a+3/4(b-a)*	*b*	0.46	**<0.05**	0.54	**<0.05**	0.503	0.79	0.49	**<0.05**	0.46	0.33
400	*a*	*a+1/4*(b-a)*	0.45	**<0.05**	0.51	0.35	0.48	**<0.05**	0.46	**<0.05**	0.59	0.38
		a+1/2(b-a)*	0.41	**<0.05**	0.51	0.46	0.43	**<0.05**	0.41	**<0.05**	0.53	0.94
		a+3/4(b-a)*	0.40	**<0.05**	0.53	**<0.05**	0.45	**<0.05**	0.41	**<0.05**	0.56	0.67
		b	0.37	**<0.05**	0.51	0.65	0.40	**<0.05**	0.35	**<0.05**	0.60	0.25
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.47	**<0.05**	0.50	0.30	0.46	**<0.05**	0.45	**<0.05**	0.46	0.56
		a+3/4(b-a)*	0.45	**<0.05**	0.53	**<0.05**	0.49	0.20	0.45	**<0.05**	0.50	0.87
		b	0.44	**<0.05**	0.51	**<0.05**	0.44	**<0.05**	0.41	**<0.05**	0.53	0.92
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.49	0.10	0.53	**<0.05**	0.53	**<0.05**	0.50	0.30	0.53	0.51
		b	0.47	**<0.05**	0.51	0.10	0.47	**<0.05**	0.45	**<0.05**	0.56	0.38
	a+3/4(b-a)*	*b*	0.47	**<0.05**	0.48	**<0.05**	0.45	**<0.05**	0.44	**<0.05**	0.52	0.66
450	*a*	*a+1/4*(b-a)*	0.44	**<0.05**	0.47	**<0.05**	0.43	**<0.05**	0.44	**<0.05**	0.53	0.69
		a+1/2(b-a)*	0.41	**<0.05**	0.51	0.15	0.43	**<0.05**	0.43	**<0.05**	0.56	0.32
		a+3/4(b-a)*	0.38	**<0.05**	0.49	**<0.05**	0.39	**<0.05**	0.38	**<0.05**	0.57	0.66
		b	0.43	**<0.05**	0.54	**<0.05**	0.40	**<0.05**	0.43	**<0.05**	0.58	0.30
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.54	**<0.05**	0.49	0.44	0.49	**<0.05**	0.53	0.64
		a+3/4(b-a)*	0.44	**<0.05**	0.52	0.07	0.46	**<0.05**	0.44	**<0.05**	0.55	0.40
		b	0.49	0.07	0.57	**<0.05**	0.46	**<0.05**	0.49	0.09	0.57	0.24
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.46	**<0.05**	0.48	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.51	0.94
		b	0.51	**<0.05**	0.53	**<0.05**	0.47	**<0.05**	0.50	0.88	0.54	0.74
	a+3/4(b-a)*	*b*	0.55	**<0.05**	0.55	**<0.05**	0.51	**<0.05**	0.55	**<0.05**	0.53	0.21
500	*a*	*a+1/4*(b-a)*	0.47	**<0.05**	0.60	**<0.05**	0.46	**<0.05**	0.46	**<0.05**	0.67	**<0.05**
		a+1/2(b-a)*	0.43	**<0.05**	0.62	**<0.05**	0.45	**<0.05**	0.43	**<0.05**	0.67	**<0.05**
		a+3/4(b-a)*	0.40	**<0.05**	0.60	**<0.05**	0.41	**<0.05**	0.38	**<0.05**	0.67	**<0.05**
		b	0.44	**<0.05**	0.66	**<0.05**	0.42	**<0.05**	0.42	**<0.05**	0.62	0.12
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.46	**<0.05**	0.55	**<0.05**	0.48	**<0.05**	0.47	**<0.05**	0.51	0.74
		a+3/4(b-a)*	0.43	**<0.05**	0.52	**<0.05**	0.45	**<0.05**	0.42	**<0.05**	0.52	0.89
		b	0.47	**<0.05**	0.60	**<0.05**	0.45	**<0.05**	0.47	**<0.05**	0.49	0.44
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.47	**<0.05**	0.48	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.50	0.92
		b	0.51	0.26	0.55	**<0.05**	0.47	**<0.05**	0.50	0.27	0.48	0.72
	a+3/4(b-a)*	*b*	0.53	**<0.05**	0.57	**<0.05**	0.51	0.44	0.55	**<0.05**	0.48	0.90
550	*a*	*a+1/4*(b-a)*	0.47	**<0.05**	0.66	**<0.05**	0.49	**<0.05**	0.45	**<0.05**	0.73	**<0.05**
		a+1/2(b-a)*	0.44	**<0.05**	0.68	**<0.05**	0.45	**<0.05**	0.41	**<0.05**	0.77	**<0.05**

		a+3/4(b-a)*	0.42	**<0.05**	0.70	**<0.05**	0.44	**<0.05**	0.43	**<0.05**	0.79	**<0.05**
		b	0.43	**<0.05**	0.67	**<0.05**	0.41	**<0.05**	0.39	**<0.05**	0.74	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.47	**<0.05**	0.54	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.57	0.24
		a+3/4(b-a)*	0.45	**<0.05**	0.54	**<0.05**	0.47	**<0.05**	0.48	**<0.05**	0.58	0.06
		b	0.46	**<0.05**	0.52	**<0.05**	0.43	**<0.05**	0.44	**<0.05**	0.56	0.49
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.48	**<0.05**	0.502	0.08	0.49	0.79	0.52	**<0.05**	0.51	0.38
		b	0.49	**<0.05**	0.48	**<0.05**	0.46	**<0.05**	0.48	**<0.05**	0.49	0.93
	a+3/4(b-a)*	*b*	0.502	0.05	0.48	**<0.05**	0.46	**<0.05**	0.47	**<0.05**	0.48	0.44
600	*a*	*a+1/4*(b-a)*	0.50	0.82	0.59	**<0.05**	0.49	**<0.05**	0.48	**<0.05**	0.67	**<0.05**
		a+1/2(b-a)*	0.44	**<0.05**	0.66	**<0.05**	0.42	**<0.05**	0.41	**<0.05**	0.71	**<0.05**
		a+3/4(b-a)*	0.41	**<0.05**	0.66	**<0.05**	0.40	**<0.05**	0.41	**<0.05**	0.70	**<0.05**
		b	0.45	**<0.05**	0.69	**<0.05**	0.40	**<0.05**	0.40	**<0.05**	0.82	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.45	**<0.05**	0.60	**<0.05**	0.43	**<0.05**	0.43	**<0.05**	0.56	0.18
		a+3/4(b-a)*	0.42	**<0.05**	0.60	**<0.05**	0.42	**<0.05**	0.43	**<0.05**	0.57	0.74
		b	0.46	**<0.05**	0.63	**<0.05**	0.41	**<0.05**	0.42	**<0.05**	0.70	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.47	**<0.05**	0.51	0.05	0.48	**<0.05**	0.50	0.12	0.51	0.81
		b	0.51	0.05	0.54	**<0.05**	0.48	**<0.05**	0.49	**<0.05**	0.62	**<0.05**
	a+3/4(b-a)*	*b*	0.54	**<0.05**	0.53	**<0.05**	0.50	0.65	0.50	0.89	0.60	**<0.05**
650	*a*	*a+1/4*(b-a)*	0.49	**<0.05**	0.56	**<0.05**	0.46	**<0.05**	0.45	**<0.05**	0.61	**<0.05**
		a+1/2(b-a)*	0.45	**<0.05**	0.60	**<0.05**	0.43	**<0.05**	0.39	**<0.05**	0.68	**<0.05**
		a+3/4(b-a)*	0.43	**<0.05**	0.67	**<0.05**	0.42	**<0.05**	0.41	**<0.05**	0.72	**<0.05**
		b	0.44	**<0.05**	0.72	**<0.05**	0.43	**<0.05**	0.39	**<0.05**	0.80	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.55	**<0.05**	0.47	**<0.05**	0.42	**<0.05**	0.60	0.14
		a+3/4(b-a)*	0.45	**<0.05**	0.64	**<0.05**	0.45	**<0.05**	0.44	**<0.05**	0.64	**<0.05**
		b	0.46	**<0.05**	0.69	**<0.05**	0.47	**<0.05**	0.43	**<0.05**	0.74	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.48	**<0.05**	0.60	**<0.05**	0.48	**<0.05**	0.52	**<0.05**	0.54	0.35
		b	0.49	**<0.05**	0.66	**<0.05**	0.50	0.90	0.50	**<0.05**	0.61	**<0.05**
	a+3/4(b-a)*	*b*	0.51	0.34	0.56	**<0.05**	0.52	**<0.05**	0.49	0.52	0.58	0.05
700	*a*	*a+1/4*(b-a)*	0.48	**<0.05**	0.56	**<0.05**	0.48	**<0.05**	0.46	**<0.05**	0.74	**<0.05**
		a+1/2(b-a)*	0.43	**<0.05**	0.59	**<0.05**	0.47	**<0.05**	0.43	**<0.05**	0.74	**<0.05**
		a+3/4(b-a)*	0.41	**<0.05**	0.59	**<0.05**	0.45	**<0.05**	0.41	**<0.05**	0.71	**<0.05**
		b	0.43	**<0.05**	0.56	**<0.05**	0.39	**<0.05**	0.39	**<0.05**	0.71	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.45	**<0.05**	0.54	**<0.05**	0.49	**<0.05**	0.46	**<0.05**	0.51	0.61
		a+3/4(b-a)*	0.42	**<0.05**	0.55	**<0.05**	0.46	**<0.05**	0.44	**<0.05**	0.54	0.75
		b	0.45	**<0.05**	0.51	0.20	0.40	**<0.05**	0.42	**<0.05**	0.53	0.85
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.48	**<0.05**	0.53	**<0.05**	0.47	**<0.05**	0.48	**<0.05**	0.53	0.98
		b	0.497	0.30	0.47	**<0.05**	0.41	**<0.05**	0.46	**<0.05**	0.52	0.97
	a+3/4(b-a)*	*b*	0.52	**<0.05**	0.44	**<0.05**	0.44	**<0.05**	0.48	**<0.05**	0.49	1.00
750	*a*	*a+1/4*(b-a)*	0.47	**<0.05**	0.55	**<0.05**	0.45	**<0.05**	0.45	**<0.05**	0.65	**<0.05**
		a+1/2(b-a)*	0.46	**<0.05**	0.49	**<0.05**	0.43	**<0.05**	0.42	**<0.05**	0.61	0.07
		a+3/4(b-a)*	0.47	**<0.05**	0.50	**<0.05**	0.41	**<0.05**	0.38	**<0.05**	0.62	0.08
		b	0.46	**<0.05**	0.54	**<0.05**	0.44	**<0.05**	0.38	**<0.05**	0.71	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.504	0.59	0.43	**<0.05**	0.48	**<0.05**	0.47	**<0.05**	0.47	0.58
		a+3/4(b-a)*	0.51	0.89	0.44	**<0.05**	0.46	**<0.05**	0.44	**<0.05**	0.48	0.39
		b	0.51	0.36	0.49	**<0.05**	0.49	0.50	0.44	**<0.05**	0.56	0.31
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.498	0.53	0.51	0.26	0.48	**<0.05**	0.47	**<0.05**	0.52	0.96
		b	0.499	0.47	0.57	**<0.05**	0.52	**<0.05**	0.47	**<0.05**	0.60	**<0.05**
	a+3/4(b-a)*	*b*	0.499	0.92	0.56	**<0.05**	0.54	**<0.05**	0.50	**<0.05**	0.57	0.08
800	*a*	*a+1/4*(b-a)*	0.50	0.32	0.60	**<0.05**	0.50	0.67	0.49	0.39	0.67	**<0.05**
		a+1/2(b-a)*	0.48	**<0.05**	0.68	**<0.05**	0.47	**<0.05**	0.44	**<0.05**	0.79	**<0.05**
		a+3/4(b-a)*	0.46	**<0.05**	0.70	**<0.05**	0.43	**<0.05**	0.40	**<0.05**	0.77	**<0.05**
		b	0.45	**<0.05**	0.72	**<0.05**	0.45	**<0.05**	0.41	**<0.05**	0.77	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.61	**<0.05**	0.47	**<0.05**	0.44	**<0.05**	0.67	**<0.05**
		a+3/4(b-a)*	0.46	**<0.05**	0.62	**<0.05**	0.43	**<0.05**	0.41	**<0.05**	0.65	**<0.05**
		b	0.45	**<0.05**	0.66	**<0.05**	0.44	**<0.05**	0.42	**<0.05**	0.65	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.48	**<0.05**	0.52	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.49	0.98
		b	0.46	**<0.05**	0.55	**<0.05**	0.47	**<0.05**	0.48	**<0.05**	0.48	0.84
	a+3/4(b-a)*	*b*	0.49	0.07	0.54	**<0.05**	0.51	0.11	0.51	**<0.05**	0.50	0.97
850	*a*	*a+1/4*(b-a)*	0.45	**<0.05**	0.53	**<0.05**	0.46	**<0.05**	0.46	**<0.05**	0.65	**<0.05**
		a+1/2(b-a)*	0.45	**<0.05**	0.65	**<0.05**	0.46	**<0.05**	0.44	**<0.05**	0.79	**<0.05**
		a+3/4(b-a)*	0.46	**<0.05**	0.72	**<0.05**	0.43	**<0.05**	0.44	**<0.05**	0.87	**<0.05**
		b	0.45	**<0.05**	0.73	**<0.05**	0.44	**<0.05**	0.43	**<0.05**	0.84	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.51	**<0.05**	0.64	**<0.05**	0.50	0.77	0.49	0.68	0.65	**<0.05**
		a+3/4(b-a)*	0.52	**<0.05**	0.72	**<0.05**	0.47	**<0.05**	0.49	0.11	0.76	**<0.05**
		b	0.501	0.26	0.72	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.74	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.51	0.16	0.61	**<0.05**	0.47	**<0.05**	0.50	0.88	0.61	**<0.05**
		b	0.49	0.07	0.61	**<0.05**	0.47	**<0.05**	0.47	**<0.05**	0.60	**<0.05**

	a+3/4(b-a)*	*b*	0.48	**<0.05**	0.51	0.31	0.50	0.89	0.48	**<0.05**	0.50	1.00
900	*a*	*a+1/4*(b-a)*	0.47	**<0.05**	0.65	**<0.05**	0.50	0.50	0.47	**<0.05**	0.80	**<0.05**
		a+1/2(b-a)*	0.45	**<0.05**	0.67	**<0.05**	0.46	**<0.05**	0.43	**<0.05**	0.83	**<0.05**
		a+3/4(b-a)*	0.41	**<0.05**	0.71	**<0.05**	0.45	**<0.05**	0.43	**<0.05**	0.85	**<0.05**
		b	0.41	**<0.05**	0.70	**<0.05**	0.44	**<0.05**	0.39	**<0.05**	0.83	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.497	0.33	0.46	**<0.05**	0.46	**<0.05**	0.57	0.17
		a+3/4(b-a)*	0.44	**<0.05**	0.55	**<0.05**	0.45	**<0.05**	0.46	**<0.05**	0.60	**<0.05**
		b	0.43	**<0.05**	0.54	**<0.05**	0.44	**<0.05**	0.41	**<0.05**	0.60	0.09
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.46	**<0.05**	0.56	**<0.05**	0.50	0.43	0.50	0.21	0.53	0.27
		b	0.45	**<0.05**	0.56	**<0.05**	0.48	**<0.05**	0.46	**<0.05**	0.53	0.35
	a+3/4(b-a)*	*b*	0.49	**<0.05**	0.49	**<0.05**	0.48	**<0.05**	0.45	**<0.05**	0.51	0.92
950	*a*	*a+1/4*(b-a)*	0.48	**<0.05**	0.61	**<0.05**	0.44	**<0.05**	0.47	**<0.05**	0.69	**<0.05**
		a+1/2(b-a)*	0.46	**<0.05**	0.60	**<0.05**	0.44	**<0.05**	0.45	**<0.05**	0.74	**<0.05**
		a+3/4(b-a)*	0.44	**<0.05**	0.63	**<0.05**	0.44	**<0.05**	0.43	**<0.05**	0.78	**<0.05**
		b	0.44	**<0.05**	0.63	**<0.05**	0.41	**<0.05**	0.43	**<0.05**	0.74	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.48	**<0.05**	0.48	**<0.05**	0.50	0.94	0.48	**<0.05**	0.54	0.32
		a+3/4(b-a)*	0.45	**<0.05**	0.51	0.93	0.50	0.16	0.46	**<0.05**	0.60	0.05
		b	0.46	**<0.05**	0.52	**<0.05**	0.47	**<0.05**	0.46	**<0.05**	0.58	0.15
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.48	**<0.05**	0.53	**<0.05**	0.50	0.54	0.48	**<0.05**	0.57	0.26
		b	0.47	**<0.05**	0.54	**<0.05**	0.47	**<0.05**	0.48	**<0.05**	0.55	0.27
	a+3/4(b-a)*	*b*	0.50	0.22	0.51	**<0.05**	0.46	**<0.05**	0.50	0.28	0.49	0.51
1000	*a*	*a+1/4*(b-a)*	0.48	**<0.05**	0.62	**<0.05**	0.49	**<0.05**	0.46	**<0.05**	0.79	**<0.05**
		a+1/2(b-a)*	0.43	**<0.05**	0.63	**<0.05**	0.44	**<0.05**	0.44	**<0.05**	0.77	**<0.05**
		a+3/4(b-a)*	0.42	**<0.05**	0.65	**<0.05**	0.46	**<0.05**	0.42	**<0.05**	0.81	**<0.05**
		b	0.42	**<0.05**	0.67	**<0.05**	0.42	**<0.05**	0.38	**<0.05**	0.85	**<0.05**
	a+1/4(b-a)*	*a+1/2*(b-a)*	0.45	**<0.05**	0.51	0.15	0.46	**<0.05**	0.48	**<0.05**	0.53	0.44
		a+3/4(b-a)*	0.45	**<0.05**	0.54	**<0.05**	0.46	**<0.05**	0.47	**<0.05**	0.58	0.10
		b	0.45	**<0.05**	0.57	**<0.05**	0.42	**<0.05**	0.43	**<0.05**	0.70	**<0.05**
	a+1/2(b-a)*	*a+3/4*(b-a)*	0.498	0.40	0.53	**<0.05**	0.51	0.23	0.48	**<0.05**	0.55	0.33
		b	0.503	0.98	0.57	**<0.05**	0.47	**<0.05**	0.45	**<0.05**	0.66	**<0.05**
	a+3/4(b-a)*	*b*	0.505	0.59		**<0.05**	0.46	**<0.05**	0.46	**<0.05**	0.63	**<0.05**

*A: $\hat{A}_{12}$, p: p-value. All p-values less than 0.05 are identified as bold.