2nd International Symposium on Search Based Software Engineering

Search Based Optimization of Requirements Interaction Management

Yuanyuan Zhang
CREST Centre
King's College London
London, UK
yuanyuan.zhang@kcl.ac.uk

Mark Harman
CREST Centre
King's College London
London, UK
mark.harman@kcl.ac.uk

***Abstract*—There has been much recent interest in Search Based Optimization for Requirements Selection from the SBSE community, demonstrating how multi-objective techniques can effectively balance the competing cost and value objectives inherent in requirements selection. This problem is known as release planning (aka the 'next release problem). However, little previous work has considered the problem of Requirement Interaction Management (RIM) in the solution space. Because of RIM, there are many subtle relationships between requirements, which make the problem more complex than an unconstrained feature subset selection problem. This paper introduces and evaluates archive-based multi-objective evolutionary algorithm, based on NSGA-II, which is capable of maintaining solution quality and diversity, while respecting the constraints imposed by RIM.**

***Keywords*-Requirements, RIM, NSGA-II, Search-based Software Engineering**

I. Introduction

In the release planning for software development, the requirements interdependency relationship is an important element which reflects how requirements interact with each other in a software system. Furthermore, it also directly affects requirements selection activity as well as requirements traceability management, reuse and the evolution process.

According to Carlshamre et al.,

> "The task of finding an optimal selection of requirements for the next release of a software system is difficult as requirements may depend on each other in complex ways" [1].

Some requirements might have technical, structural or functional correlations that need to be fulfilled together or separately, or one requirement might be the prerequisite of another. The analysis and management of dependencies among requirements is called Requirements Interaction Management (RIM) which is defined as

> "the set of activities directed towards the discovery, management, and disposition of critical relationships among sets of requirements" [2].

Robinson et al. [2] gave the definition of requirement interaction:

> "Two requirements R_1 and R_2 is said to interact if (and only if) the satisfaction of one requirement affects the satisfaction of the other."

RIM consists of a series of activities related to requirement dependencies which are complex and challenging tasks.

Few previous authors [3], [4], [5] focus on the role of requirements dependencies in the solution space. However, dependencies can have a very strong impact on the development process in a typical real world project. Bagnall et al. [3] only considered the *Precedence* dependency type, representing the relationship as a directed, acyclic graph. Its vertices are denoted as individual requirements and its edges, directed from one vertex to another, are denoted as the *Precedence* dependency between the requirements. Greer and Ruhe extended the work by adding the *And* dependency type together with *Precedence* as the constraints in their EVOLVE model. Franch and Maiden applied the *i** approach to model dependencies for COTS component selection.

In this paper, the Search-based Requirements Selection Optimization framework allows for the five most common types of requirements dependencies. The objectives are to investigate the influences of requirements dependencies on the automated requirements selection process for release planning and to validate the ability of the proposed framework to find the optimal balance in the solution space for release planning under different circumstances.

The study is based on the assumption that the dependence identification activity has been completed. Here we present the most common interaction types found in the requirements literature. These will be studied in this paper. The paper will show how multi-objective SBSE can be adapted to take account of RIM.

And Given requirement R_1 is selected, then requirement R_2 has to be chosen.

Or Requirements R_1 and R_2 are conflicting to each other, only one of R_1, R_2 can be selected (Exclusive OR).

Precedence Given requirement R_1 has to be implemented before requirement R_2.

Value-related Given requirement R_1 is selected, then this selection affects the value of requirement R_2 to the stakeholder;

Cost-related Given requirement R_1 is selected, then this selection affects the cost of implementing

978-0-7695-4195-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SSBSE.2010.16

requirement R_2.

The rest of the paper is organized as follows: In Section II the problem is formalized as an SBSE problem, while Section III describes the data sets and algorithms used. Section IV presents the results for dependence aware requirements optimization and discusses the findings. Section V describes the context of related work in which the current paper is located. Section VI concludes the paper.

II. Fitness Function

In the context of Value/Cost-based requirements assignments analysis, the dependencies among requirements need to be accounted for within the fitness function. This section describes our fitness computation and how we incorporate RIM into this fitness.

Assume that the set of possible software requirements is denoted by:

$$\Re = \{r_1, \ldots, r_n\}$$

The requirements array R is defined by:

$$R = \left\{ \begin{array}{cccccc} r(1,1) & r(1,2) & \cdots & r(1,i) & \cdots & r(1,n) \\ r(2,1) & r(2,2) & \cdots & r(2,i) & \cdots & r(2,n) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ r(j,1) & r(j,2) & \cdots & r(j,i) & \cdots & r(j,n) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r(n,1) & r(n,2) & \cdots & r(n,i) & \cdots & r(n,n) \end{array} \right\}$$

First, we formalise the RIM constraints that were listed informally in the introduction to this paper.

And Define an equivalence relation ξ on the requirements array R such that $r(i,j) \in \xi$ means that r_i is selected if and only if requirement r_j has to be chosen.

Or Define an equivalence relation φ on the requirements array R such that $r(i,j) \in \varphi$ (equivalently $r(j,i) \in \varphi$) means that at most one of r_i, r_j can be selected.

Precedence Define a partial order χ on the requirements array R such that $r(i,j) \in \chi$ means that requirement r_i has to be implemented before requirement r_j.

Value-related Define a partial order ψ on the requirements array R such that $r(i,j) \in \psi$ means that if the requirement r_i is selected, then its inclusion affects the value of requirement r_j for the stakeholder.

Cost-related Define a partial order ω on the requirements array R such that $r(i,j) \in \omega$ means that if the requirement r_i is selected, then its inclusion affects the cost of implementing requirement r_j.

In addition, the relations ξ, φ and χ should satisfy

$$\xi \bigcap \varphi = \emptyset \quad \wedge \quad \xi \bigcap \chi = \emptyset$$

in order to guarantee consistency in the requirements dependency relationship.

The fitness function with dependency constraints is defined as follows:

$$\text{Maximize } f_1(\overrightarrow{x}) = \sum_{i=1}^{n} score_i \cdot x_i$$

$$\text{Maximize } f_2(\overrightarrow{x}) = -\sum_{i=1}^{n} cost_i \cdot x_i$$

subject to

$$x_i = x_j \quad \text{for all pairs} \quad r(i,j) \in \xi \quad \text{(And constraints)}$$

$$x_i \neq x_j \ \vee \ x_i = x_j = 0 \ \text{ for all pairs } \ r(i,j) \in \varphi \quad \text{(Or constraints)}$$

$$x_i = 1 \wedge x_j = 1 \quad \vee \quad x_i = 1 \wedge x_j = 0 \quad \vee \quad x_i = x_j = 0$$

$$\text{for all pairs} \quad r(i,j) \in \chi \quad \text{(Precedence constraints)}$$

In terms of Value-related and Cost-related requirements dependencies, they cannot be transformed from dependencies into constraints. So the fitness values of a solution are changed directly when there exists a Value-related or Cost-related dependency, as follows:

$$\text{If} \quad x_i = x_j = 1 \quad \text{for all pairs} \quad r(i,j) \in \psi \ \Rightarrow$$

$$\text{Update Fitness Value of } f_1(\overrightarrow{x})$$

$$\text{If} \quad x_i = x_j = 1 \quad \text{for all pairs} \quad r(i,j) \in \omega \ \Rightarrow$$

$$\text{Update Fitness Value of } f_2(\overrightarrow{x})$$

III. Experimental Set Up

To assess the likely impact of requirements dependencies on the automated requirements selection process, a set of empirical studies were carried out. This section describes the test data sets used and the search-based algorithm applied to the requirements interaction management.

A. Data Sets

The “27 combination random data sets” used in previous studies [6] were adopted in this studies. These are the basis of the data sets we will use in the empirical studies. The “27-random” data sets were generated randomly according to the problem representation. These synthetic test problems were created by assigning random choices for value and cost. The range of costs were from 1 through to 9 inclusive (zero cost is not permitted). The range of values were from 0 to 5 inclusive (zero value is permitted, indicating that the stakeholder places no value on this requirement).

Table I
27 COMBINATION RANDOM DATA SETS

	R_{small}	R_{medium}	R_{large}
C_{small}	$C_s\ R_s\ D_{low}$ $C_s\ R_s\ D_m$ $C_s\ R_s\ D_h$	$C_s\ R_m\ D_{low}$ $C_s\ R_m\ D_m$ $C_s\ R_m\ D_h$	$C_s\ R_l\ D_{low}$ $C_s\ R_l\ D_m$ $C_s\ R_l\ D_h$
C_{median}	$C_m\ R_s\ D_{low}$ $C_m\ R_s\ D_m$ $C_m\ R_s\ D_h$	$C_m\ R_m\ D_{low}$ $C_m\ R_m\ D_m$ $C_m\ R_m\ D_h$	$C_m\ R_l\ D_{low}$ $C_m\ R_l\ D_m$ $C_m\ R_l\ D_h$
C_{large}	$C_l\ R_s\ D_{low}$ $C_l\ R_s\ D_m$ $C_l\ R_s\ D_h$	$C_l\ R_m\ D_{low}$ $C_l\ R_m\ D_m$ $C_l\ R_m\ D_h$	$C_l\ R_l\ D_{low}$ $C_l\ R_l\ D_m$ $C_l\ R_l\ D_h$

Table II
SCALE RANGE OF '27-RANDOM' DATA SET

	Small	Medium	Large
No. of Stakeholders	2-5	6-20	21-50
No. of Requirements	1-100	101-250	251-600
	Low	Medium	High
Density of Matrix	0.01-0.33	0.34-0.66	0.67-1.00

Table III
SCALE OF A, B, C AND D DATA SETS: EXPLORATION OF THE CONFIGURATION SPACE FOR RIM

	R_{small}	R_{medium}	R_{large}
C_{small}			**C:** $C_s\ R_l\ D_m$
C_{median}		**A:** $C_m\ R_m\ D_m$	
C_{large}	**B:** $C_l\ R_s\ D_m$		**D:** $C_l\ R_l\ D_h$

Table IV
A, B, C AND D DATA SETS: CHOOSING A VARIETY OF RIM DISTRIBUTIONS

Data Set	No. of Stakeholders	No. of Requirements	Density of Matrix
A	11	230	0.53
B	34	50	0.39
C	4	258	0.51
D	21	412	0.98

This simulates the situation where a stakeholder ranks the choice of requirements (for value) and the cost is estimated to fall in a range: very low, low, medium, high, very high. The number of stakeholders and the number of requirements are divided into three situations, namely, small scale, medium scale and large scale; the density of the stakeholder-requirement matrix is defined as low level, medium and high level. Table I lists the combination of all cases schematically. As can be seen in Table II, the data set divides the range of a variable into a finite number of non-overlapping intervals of unequal width.

Any randomly generated, isolated data set clearly cannot reflect real-life scenarios. We do not seek to use our pseudo random generation of synthetic data as a substitute for real world data. Rather, we seek to generate synthetic data in order to explore the behavior of our algorithms in certain well defined scenarios. The use of synthetic data allows us to do this within a laboratory controlled environment. Specifically, we are interested in exploring the way the search responds when the data exhibits a presence or absence of correlation in the data. As well as helping us to better understand the performance and behavior of our approach in a controlled manner, this also allows us to shed light on the real world data, comparing results with the synthetic data.

To explore this, in the empirical studies, we generated four data sets exhibiting different scales and densities using the approach to data set generation depicted in Table I and Table II. We named the sets A, B, C and D. In the A data set, the parameter choices were chosen to be "medium" and the density of the stakeholder-requirement matrix was also chosen to be "medium". The parameters of the data set were randomly generated within the given scale intervals. More concretely, the number of requirements is 230, the number of stakeholders is 11 and the density of matrix is 0.53.

Following the same principle, the B, C and D data sets were generated. The scales and densities chosen and the specific parameters created are listed in Table III and Table IV.

In the four data sets that were generated, all the requirements were initially created to be independent. To introduce the dependency, relationships among requirements are added randomly but with respect to constraints. Five two-dimensional arrays $And(i,j)$, $Or(i,j)$, $Pre(i,j)$, $Val(i,j)$ and $Cos(i,j)$ ($1 \leq i \leq n$ and $1 \leq j \leq n$) are defined to represent the five requirements dependency types.

$$And(i,j), Or(i,j)\ and\ Pre(i,j) \in \{0,1\}$$

$And(i,j) = 1 \wedge And(j,i) = 1$ if requirement r_i and r_j have *And* dependency and 0 otherwise; $Or(i,j) = 1 \wedge Or(j,i) = 1$ if requirement r_i and r_j have *Or* dependency and 0 otherwise; $Pre(i,j) = 1 \wedge Pre(j,i) = 0$ if requirement r_i and r_j have *Precedence* dependency.

The above three dependency arrays are bit vectors which compactly store individual boolean values, as flags to indicate the relationship between requirements. As each random relationship is created, we check to ensure that the *And*, *Or* and *Precedence* dependence constraints are respected, thereby guaranteeing the generation of a valid instance.

In the $Val(i,j)$ and $Cos(i,j)$ arrays, the values are not 0 or 1, but rather the extent of impact of the *Value* or *Cost* which are expressed as a numerical percentage. $Val(i,j) \neq 0$ if requirements r_i and r_j have a *Value-related* dependency; $Cos(i,j) \neq 0$ if requirements r_i and r_j have a *Cost-related* dependency.

B. Algorithms

The search algorithms used in this work were NSGA-II [7] and a modified version we implemented that is specifically constructed to produce a good Pareto front for RIM-constrained problem. Our modification is inspired by Praditwong and Yao's Two-Archive multi-objective evolutionary optimization algorithm [8]. Results will be presented to compare the performance of two algorithms.

Keeping the final set of non-dominated solutions is good enough for general multi-objective optimization work. However, when we take account for requirements dependencies, the selected optimal non-dominated solutions might not respect the dependency constraints. As a result some solutions might have to be eliminated. This may mean rejecting otherwise 'optimal' solutions in favor of previously considered and otherwise less optimal solutions.

In order to preserve these potential candidate solutions, this paper introduces an archive-based variation of the NSGA-II algorithm to retain near optimal solutions (maintaining diversity and quantity of the solutions) based on constraints.

All search-based approaches were run for a maximum of 50,000 fitness function evaluations. The population was set to 500. We used a simple binary GA encoding, with one bit to code for each decision variable (the inclusion or exclusion of a requirement). The length of a chromosome is thus equivalent to the number of requirements. Each experimental execution of each algorithm was terminated when the generation number reached 101 (i.e after 50,000 evaluations). All genetic approaches used tournament selection (the tournament size is 5), single-point crossover and bitwise mutation for binary-coded GAs. The crossover probability was set to $P_c = 0.8$ and mutation probability to $P_m = 1/n$ (where n is the string length for binary-coded GAs). In the archive based NSGA-II algorithm, the total capacity of the archives was set to 500.

IV. Empirical Studies and Results

This section presents the experiments carried out to investigate the results in the presence of requirement dependencies and to compare the performance of two search algorithms.

There are two types of empirical study: a Dependency Impact Study (DIS) and a Scale Study (SS). Data set A is used for DIS and data sets B, C and D are used for SS.

In DIS, the experiment is designed for the purpose of evaluating the impacts of five different dependency types on the requirements selection process. Data set A is used throughout the DIS experiment in order to set up a uniform baseline for comparison. Three experiments were conducted in DIS, described as follows:

1) Applying the NSGA-II algorithm to data set A with and without dependencies separately, in order to carry out the comparison of the results of each dependency type (five types individually).
2) Applying the NSGA-II and archive based NSGA-II to data set A aiming to compare the performances of two algorithms under the dependency constraints (five types individually).
3) Considering dependence relationships as a whole in order to seek to investigate the difference among the solutions generated by the two algorithms.

In SS, we report results concerning the performance of the two algorithms as the data sets increase in size. There are three data sets B, C and D with the number of stakeholders ranging from 4 to 34 and the number of requirements ranging from 50 to 412.

In both studies, the five dependency types can be divided into two categories: fitness-invariant dependency (And, Or and Precedence) and fitness-affecting dependency (Value-related and Cost-related). Therefore we will discuss the two scenarios separately in each study. In addition, the same dependency density levels were used for all five dependencies. That is, we assume that they are equally common in the requirements correlations.

A. Dependency Impact Study

1) Aims: Three goals need to be achieved in DIS listed as follows:

1) The Pareto front should cover the maximum number of different situations and provide a set of well distributed solutions.
2) The solutions contained in the Pareto front should be as close as possible to the optimal Pareto front of the problem.
3) The solutions are required to pass the evaluation without failure to meet constraints.

2) And, Or and Precedence: In the first part of the section, we present the results of applying the NSGA-II and the archive based NSGA-II algorithms to handle *And*, *Or* and *Precedence* requirements dependencies. The results generated by the standard NSGA-II algorithm are shown in Figures 1, 2 and 3; and the results from the archive based NSGA-II algorithm are shown in Figures 4, 5 and 6 separately.

The '+', '◯' and '$*$' symbols plotted in the figures denote the final non-dominated solutions found. Each solution represents a subset of requirements selected. The '+' symbol represents the solutions found without regard to requirement dependencies. They are also marked in grey to

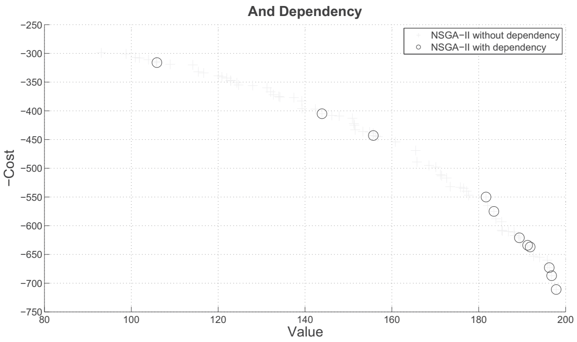

Figure 1. Results Comparison: with and without *And* dependency

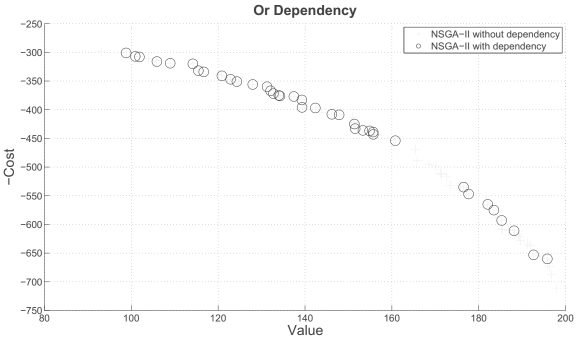

Figure 2. Results Comparison: with and without *Or* dependency

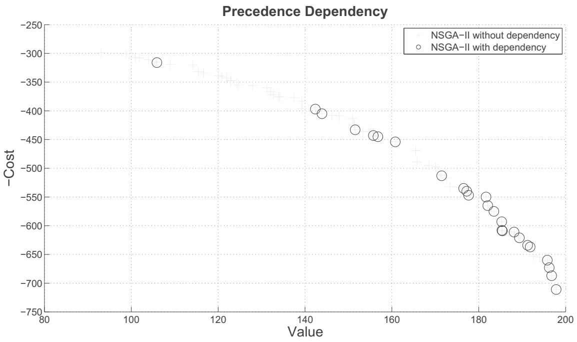

Figure 3. Results Comparison: with and without *Precedence* dependency

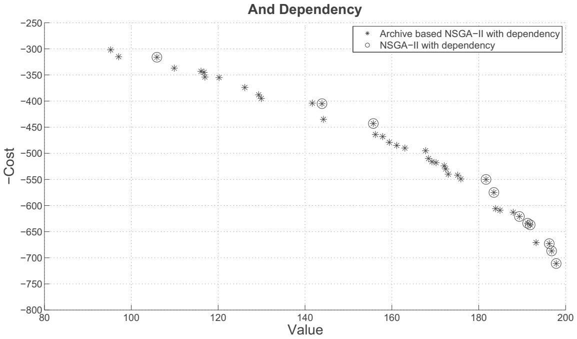

Figure 4. Results Comparison: Original and Degenerated Pareto fronts with *And* dependency

distinguish them from the others (which do take account of dependencies).

In Figures 1, 2 and 3, we observe that the shapes of Pareto fronts, consisting of a number of grey '+' symbols, are the same. These are solutions generated by the NSGA-II algorithm without consideration of requirements dependency relationship and so they are expected to be identical. They are used as the baseline to explore the impact of three types of dependencies on the requirements selection results.

We illustrate the results in Figure 1 when the *And* dependency relationships exist among the requirements. '◯' symbols indicate solutions which respect the *And* dependence. As can be seen from the graph, all the '◯' solutions still fall on the Pareto front composed of grey '+' symbols. However, there is a large decrease in the number of '◯' solutions compared to the number of '+' solutions. In other words, a few solutions survived and the rest were eliminated (from the selection) because of the failure to meet dependency constraints. Another obvious observation drawn from this graph is that the distribution of '◯' solutions is neither as smooth nor as uniform as the '+' solutions. That is, a certain number of big or small gaps exist among them. These two observations indicate that the algorithm can neither provide good solutions in quantity nor maintain a good diversity (quality) on the Pareto front under *And* dependency constraints.

Compared to the results of *Or* and *Precedence* dependencies in Figures 2 and 3, the downward trends in the number of '◯' solutions are roughly the same, but the extent is different. Similar observations can be made from the two figures: the results show a slight decrease in the number of the solutions. Moreover, the distribution is more continuous, exhibiting a few small gaps among the solutions.

In conclusion, the shapes of Pareto fronts (results) are affected by the different dependency constraints to a different extent. The *And* dependency problem appears to denote a tighter constraint than the *Or* and *Precedence* dependencies for search-based requirements optimization. The latter two denote problems for which it is relatively easy to find the solutions that satisfy the constraints.

To explore these findings in more detail, we designed a more robust adaptive algorithm for both tight and loose constraints; the archive-based NSGA-II algorithm. The results for three types of constraints are shown in Figures 4, 5 and 6 respectively. The '$*$' denotes the solution generated by the archive-based version of the NSGA-II algorithm and the '◯' by NSGA-II.

From Figure 4, we can see the archive based '$*$' solutions actually reach all the points on the previous '◯' Pareto front, sharing all the common points generated by NSGA-II. The

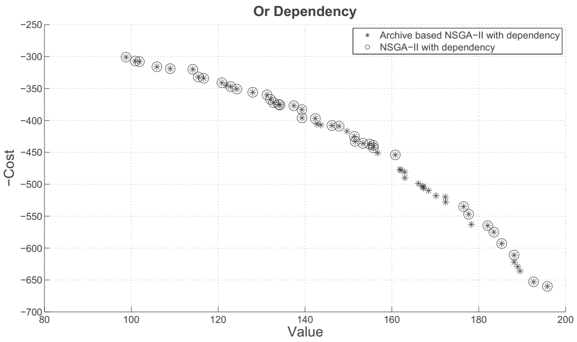

Figure 5. Results Comparison: Original and Degenerated Pareto fronts with *Or* dependency

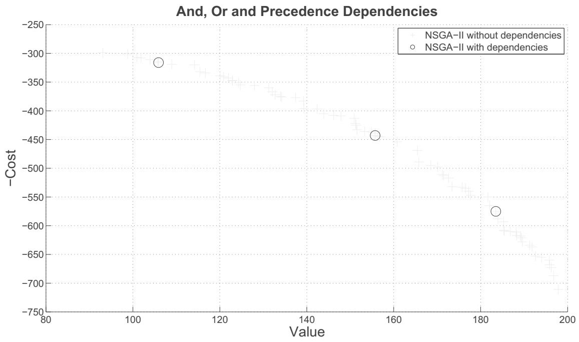

Figure 7. Results Comparison: with and without *And*, *Or* and *Precedence* dependencies

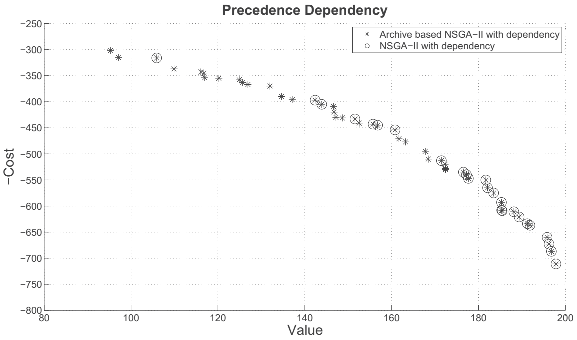

Figure 6. Results Comparison: Original and Degenerated Pareto fronts with *Precedence* dependency

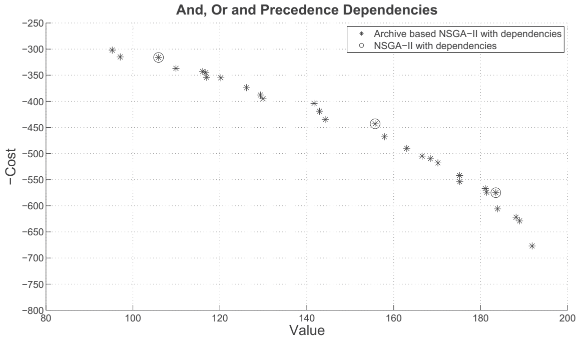

Figure 8. Results Comparison: Original and Degenerated Pareto fronts with *And*, *Or* and *Precedence* dependencies

Pareto front in this problem is orientated towards the upper right. The improved algorithm provided a *degenerated* ‘∗’ Pareto front.

The *degenerated* Pareto front means that the ‘∗’ front generated seems to become worse when compared to the grey ‘+’ front, but it discovers a larger number of good solutions to fill the gaps while meeting the constraints. That is, the diversity of solutions is significantly improved and the number of solutions on the Pareto front is also increased. The algorithm generated similar results when dealing with *Or* and *Precedence* dependency constraints, as illustrated in Figures 5 and 6.

These results indicate that the archive is able to ‘repair’ the gaps which open up in the Pareto front when RIM constraints are imposed. In this way, the archive-based technique can provide stable and fruitful solutions, which are not merely ‘good enough’ but also ‘robust enough’ under the strict constraints that characterize the problem.

Finally, all three dependencies were taken into consideration to access their overall combined impact. In the Figure 7, the ‘◯’ solutions denote the final results that satisfy all the dependencies constraints. Combining *And*, *Or* and *Precedence* dependencies together, the constraints become much tighter. It is easy to see that very few ‘◯’ solutions remain on the Pareto front based on the NSGA-II algorithm. By contrast, Figure 8 shows a smooth, relatively non-interrupted Pareto front, consisting of ‘∗’ solutions, generated by archive based NSGA-II.

3) Value-related and Cost-related: In this section we focus on the last two types of requirement dependencies: *Value-related* and *Cost-related*. These two impose no constraint on the fitness function but have direct influence on the fitness value.

The results are illustrated in Figure 9. There are four subgraphs in the figure: (a) is the original Pareto front without dependency generated by NSGA-II; (b) and (c) show the results under *Value-related* and *Cost-related* dependencies respectively; (d) presents the changed Pareto front when combining these two dependencies.

We observe that the shapes of the four Pareto fronts produced are different. They are not like the previous results of the first three dependencies: eliminating solutions on the unconstrained Pareto fronts or using a ‘degenerate’ front are not viable for *Value/Cost-related* constraints. *Value/Cost-related* relationships among the requirements can directly contribute to an increase or a decrease in the fitness values obtained for a selected solution. In this way, the shape of the Pareto front is changed more than once without dependency.

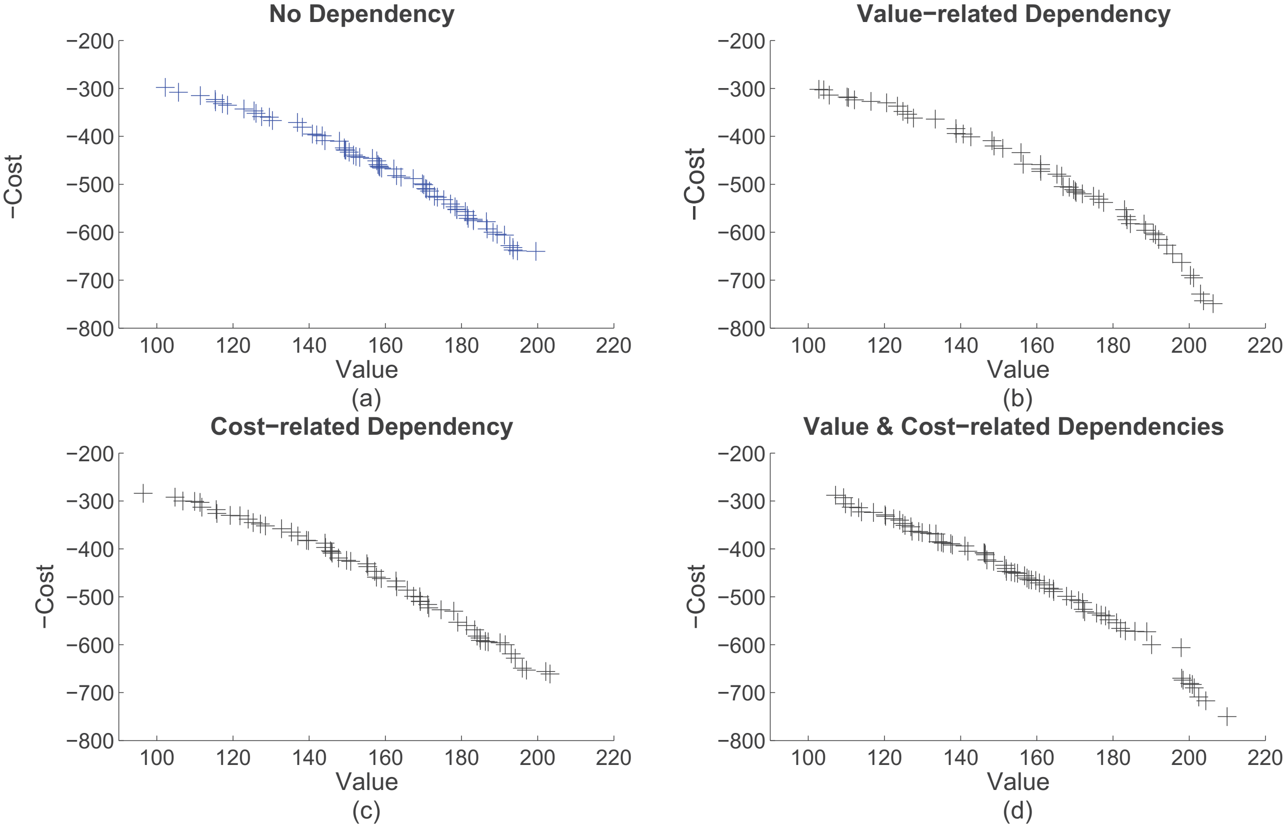

Figure 9. Results Comparison: Original and Changed Pareto front with Value-related and Cost-related dependencies

B. Scale Study

In this section, we report on the second empirical study – the Scale Study. The results are presented in the Figures 10, 11 and 12. As described at the beginning of Section IV, the techniques were applied to three data sets B, C and D generated from the smaller scale to a relatively larger one in terms of the number of stakeholders involved and the number of requirements fulfilled. The details are listed in Table III and Table IV.

In this study, all three dependency constraints are considered together. The results are plotted in one graph for each data set. In the figures, the grey Pareto front, consisting of a number of '+' solutions, denotes the results without handling dependencies generated by the NSGA-II algorithm; the '◯' solutions are the survivors after selection for meeting the constraint; the '$*$' solutions which are produced by the archive based NSGA-II algorithm constitute the *degenerated* Pareto front.

When the problem is gradually scaled up, from the graphs we can see that the number of the '◯' solutions consistently and rapidly decreases. As illustrated in Figure 12, the '◯' solutions have a poor spread over the Pareto front. By contrast, the '$*$' solutions still fill the gaps among the '◯' solutions and produce a relatively smooth and continuous Pareto front.

These three data sets B, C and D are considered using the same proportion of possible dependencies (6% of the number of requirements). Another observation from the three figures is that the distance between the original '+' Pareto front and the *degenerated* '$*$' Pareto front is wider in Figure 12 than in Figure 10. The Pareto fronts move towards the lower left part of the solution space, in order to find near-optimal solutions that have a good spread as well as having (more than) enough candidate solutions.

V. Related Work

Dependence analysis is a part of the overall traceability problem for requirements engineering. The task of requirement traceability is to identify and document traceability links among requirements and between requirements and following SE activities in both a forwards and backwards direction [9]. Requirements traceability is crucial for the success of the system. It enables detection of the conflicting requirements and reduction of missing requirements. Furthermore, it can track the progress of a project, assess the impact of various changes and provide complete information

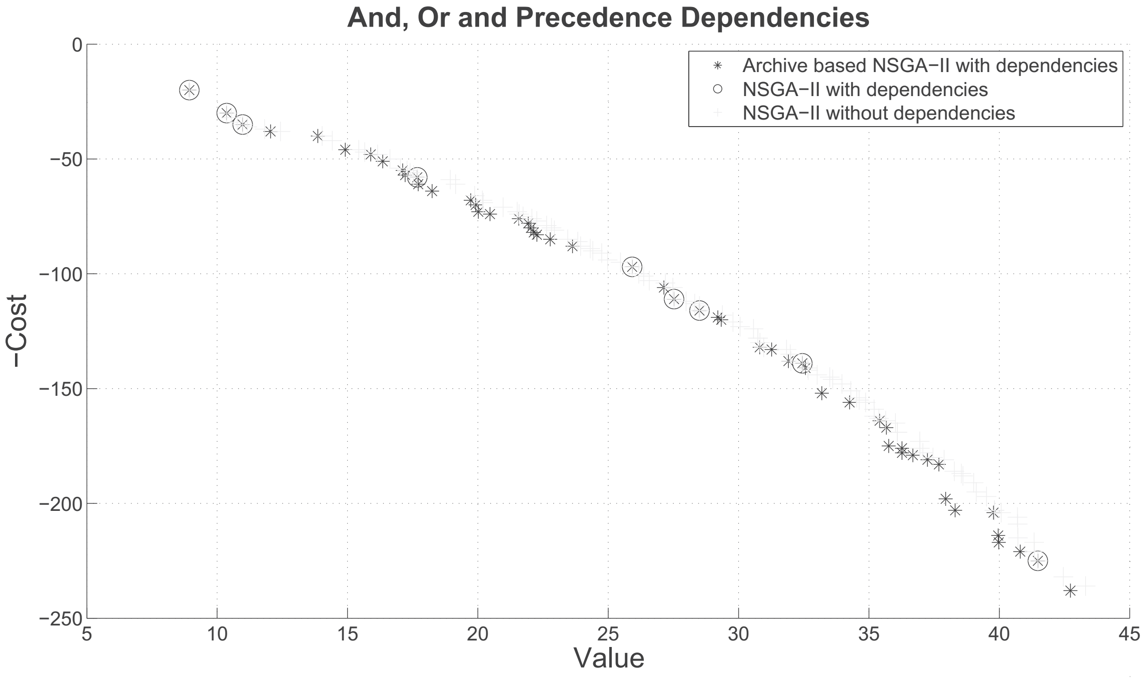

Figure 10. Results for Data Set B: 34 Stakeholders, 50 Requirements

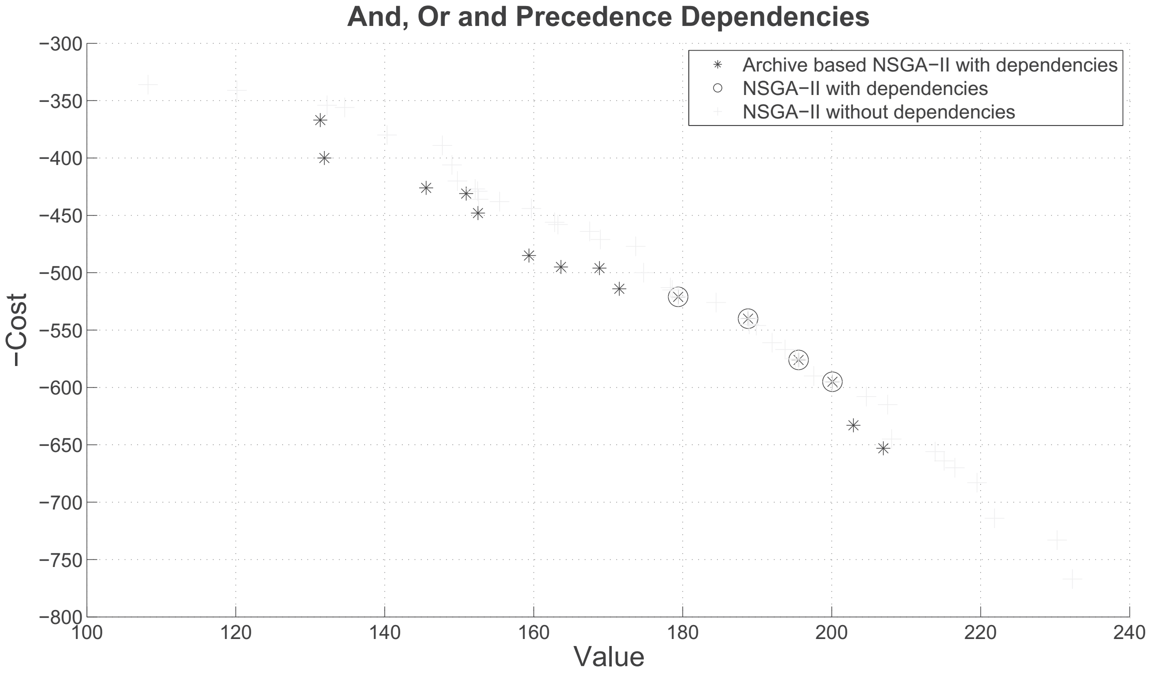

Figure 11. Results for Data Set C: 4 Stakeholders, 258 Requirements

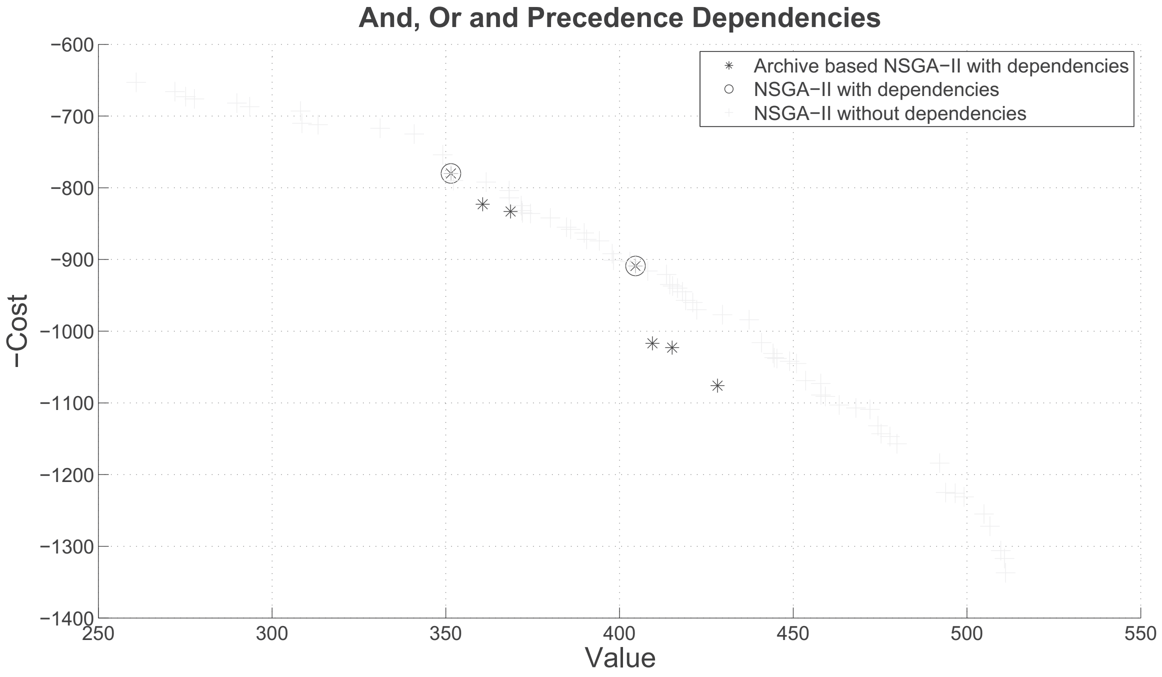

Figure 12. Results for Data Set D: 21 Stakeholders, 412 Requirements

in the SE lifecycle.

There are many ways to represent traceability links. The traceability matrix [10] and cross references [9] are both regarded as good practice which have been widely used in industry. In addition, a large number of requirements tools support traceability management [9]. One of the most famous tools is DOORS (Dynamic Object Oriented Requirements System) [11].

Pohl [12] proposed the *traceability meta model* to establish a traceability structure, which included dependence models aiming to describe the relations between trace objects. Karlsson et al. [13] opened up the discussion on supporting requirements dependencies in the requirements selection process. Robinson et al. [2] provided the basic concepts and scope of Requirements Interaction Management (RIM). They introduced RIM process in general and a historical perspective of RIM. Carlshamre and Regnell [14] described a two-dimensional (*scope* and *explicitness*) representation to investigate different types of dependencies. Subsequently Carlshamre et al. [1] extended their work and carried out an industrial survey of requirements interdependencies in software product release planning. A functional and value-related dependence classification scheme was proposed in detail. The survey also tried to find the possible relationship between the dependence types and development contexts. Dahlstedt and Persson [15], [16] provided an overview of research work concerning comparing and validating the different requirements dependencies classification frameworks.

There was some work which suggested that proper treatment of RIM should take account of different types of requirements interactions [1], [5], [13], [15], [17].

VI. Summary

This paper presented Requirements Interaction Management (RIM) and has taken RIM into consideration in the automated requirements selection process for the release planning problem. Five basic requirement dependencies were introduced. The first three types were considered to be constraints within the fitness functions; the latter two directly involved in the performance.

A“27 combination random data sets” model was generated to develop a procedure in order to better approximate real world situations. Two variable factors were considered in the data generation model. One is the different levels of data set scales which are related to the number of requirements and the number of stakeholders; the other is the density of the data sets.

To simulate the release planning selection process under requirements dependencies, two empirical studies were carried out; the Dependency Impact Study (DIS) which is designed to investigate the influences of five different dependency types and the Scale Study (SS) which concerns the performance of the two search techniques when the data sets scale up.

The same data set was used throughout the DIS experiment aiming to set up a uniform baseline for influence comparison. The results of the empirical studies illustrated that the *And* dependency appears to denote a tighter constraint than the *Or* and *Precedence* dependencies for search-based requirements optimisation. When all three dependencies were taken into consideration to access their overall combined impact, the constraints in this case became much tighter. For *Value-related* and *Cost-related* dependencies, they directly contributed to an increase or a decrease in the fitness values and further changed the shape of the Pareto front.

In SS, three data sets from smaller scale to a relatively larger one were applied. The results showed that Archive based NSGA-II could produce a smooth, relatively non-interrupted Pareto front compared to NSGA-II. When the data set was gradually scaled up, the number of solutions generated by the latter consistently and rapidly decreases. Instead, Archive based NSGA-II could still find better solutions both in diversity and quantity.

RIM is of vital importance from a software release planning point of view. For instance, the certain optional requirements can be put into one release or be separated into several releases according to their dependency relationships in order to save implementation cost and increase revenue.

Aided by search-based automated RIM, the requirements engineer can faster and more easily address this problem. For any non-trivial problem, many factors need to be considered in the requirements selection process. It is always important to look at the requirements from different perspectives. Unlike human-based search, automated search techniques carry with them no bias. They automatically scour the search space for solutions that best fit the (stated) human assumptions in the fitness function.

Acknowledgment

The authors would like to thank Kathy Harman assisted with proof reading.

References

[1] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag, "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning," in *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE '01)*, 2001.

[2] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements Interaction Management," Georgia State University, Tech. Rep. 99-7, August 1999.

[3] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The Next Release Problem," *Information and Software Technology*, vol. 43, no. 14, pp. 883–890, December 2001.

[4] X. Franch and N. A. Maiden, "Modelling Component Dependencies to Inform Their Selection," in *Proceedings of the 2nd International Conference on COTS-Based Software Systems (ICCBSS '03)*, ser. LNCS, vol. 2580. Ottawa, Canada: Springer, 10-12 February 2003, pp. 81–91.

[5] D. Greer and G. Ruhe, "Software Release Planning: An Evolutionary and Iterative Approach," *Information & Software Technology*, vol. 46, no. 4, pp. 243–253, March 2004.

[6] Y. Zhang, E. Alba, J. J. Durillo, S. Eldh, and M. Harman, "Today/Future Importance Analysis," in *Proceedings of International Conference on Genetic and Evolutionary Computation (GECCO '10)*. Portland, Oregon, USA: ACM, 7-11 July 2010, to appear.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[8] K. Praditwong and X. Yao, "A New Multi-Objective Evolutionary Optimisation Algorithm: The Two-Archive Algorithm," in *Proceedings of he 2006 International Conference on Computational Intelligence and Security (CIS '06)*, vol. 1. Guangzhou, China: IEEE Press, 3-6 November 2006, pp. 286–291.

[9] O. C. Z. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," in *Proceedings of the 1st International Conference on Requirements Engineering (RE '94)*. Colorado Springs, Colorado, USA: IEEE Computer Society, 18-21 April 1994, pp. 94–101.

[10] B. Ramesh, T. Powers, C. Stubbs, and M. Edwards, "Implementing Requirements Traceability: A Case Study," in *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering (RE '95)*. York, UK: IEEE Computer Society, 27-29 March 1995, pp. 89–95.

[11] "IBM Rational DOORS (Dynamic Object Oriented Requirements System), http://www.telelogic.com/Products/doors/."

[12] K. Pohl, *Process-Centered Requirements Engineering*. Research Studies Press, 1996.

[13] J. Karlsson, S. Olsson, and K. Ryan, "Improved Practical Support for Large-scale Requirements Prioritizing," *Requirements Engineering Journal*, vol. 2, no. 1, pp. 51–60, 1997.

[14] P. Carlshamre and B. Regnell, "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes," in *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA '00)*. London, UK: IEEE Computer Society, 4-8 September 2000, pp. 961–965. [Online]. Available: http://computer.org/proceedings/dexa/0680/06800961abs.htm

[15] Å. G. Dahlstedt and A. Persson, "Requirements Interdependencies - Moulding the State of Research into A Research Agenda," in *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (RefsQ '03)*, Klagenfurt/Velden, Austria, 16-17 June 2003.

[16] ——, *Engineering and Managing Software Requirements*. Springer Berlin Heidelberg, 2005, ch. 5 Requirements Interdependencies: State of the Art and Future Challenges, pp. 95–116.

[17] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements Interaction Management," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 132–190, June 2003.