"Fairness Analysis" in Requirements Assignments

Anthony Finkelstein[1], Mark Harman[2], S. Afshin Mansouri[3], Jian Ren[4] and Yuanyuan Zhang[2]

[1]University College London
Malet Place, London
WC1E 6BT, UK
a.finkelstein@cs.ucl.ac.uk

[2]King's College London
Strand, London
WC2R 2LS, UK
{mark.harman, yuanyuan.zhang}@kcl.ac.uk

[3]Brunel University
Uxbridge, Middlesex
UB8 3PH, UK
afshin.mansouri@brunel.ac.uk

[4]Chinasoft Int. Corp.
55 Xueyuan South, Beijing
100081, P.R.China
v-jiren@microsoft.com

Abstract

Requirements engineering for multiple customers, each of whom have competing and often conflicting priorities, raises issues of negotiation, mediation and conflict resolution. This paper uses a multi-objective optimisation approach to support investigation of the trade-offs in various notions of fairness between multiple customers. Results are presented to validate the approach using two real-world data sets and also using data sets created specifically to stress test the approach. Simple graphical techniques are used to visualize the solution space.

1 Introduction

This paper addresses a requirements analysis setting in which there are many customers, each with competing (and possibly conflicting interests). This is an increasingly prevalent because of the growing scale and complexity of the organisations that requirements analysis must address. Where there may be many customers, each with their own view on the sets of requirements to be prioritized, the goal of the requirements engineer may appear to resemble an invidious attempt to please "all of the people all of the time".

The authors have worked with Motorola on the problem of multi customer requirements. The techniques for fairness analysis proposed in this paper have been applied to a real world set of requirements from Motorola and the results are reported as part of the validation of this work. The Motorola data set concerns a set of 35 requirements for hand held communication devices. In this case, the customers are four mobile telephony service providers, each of which has a different set of priorities with respect to the features that they believe ought to be included in each handset. Motorola also maintains cost data, in the form of the estimated cost of implementation of each requirement. The paper shows how it is possible to explore trade offs and tensions between the customers in an attempt to satisfy multi definitions of fairness.

To address this problem, the paper adopts a search-based optimisation approach, which it uses to automate the exploration of the possible trade offs and conflicts between various notions of fairness. The search explores the space of possible allocations of requirements for the next release of the system.

Requirements analysis problems, with their large space of possible solution choices and complex and often competing constraints have proved to be natural candidates for optimisation based analysis. Previous work in this area has shown that meta heuristic optimisation techniques can be used to search for a balance between the costs and benefits associated with sets of requirements in what has come to be known as the Next Release Problem (NRP) [2, 16] and Release Planning [4, 20, 21, 22, 26, 27, 28]. That is, the problem is to find an answer to the question: 'Which requirements should appear in the next release of the system?'.

Existing work on this problem has tended to treat the NRP as a single objective problem formulation, in which the various constraints and objectives that characterize the requirements analysis problem are combined into a single

objective fitness function. A variety of optimisation algorithms have been applied to single objective formulations, including integer linear programming, greedy algorithms, branch and bound, simulated annealing and genetic algorithms [2, 16, 32]. Single objective formulations have the draw back that the maximisation of one concern might be achieved at the expense of the potential maximisation of another resulting in a bias guiding the search to a certain part of the solution space.

More recently however, there has been work on multi-objective formulations of the problem [29, 33]. In this work on the Multi-Objective Next Release Problem (MONRP), each of the objectives to be optimized is treated as a separate goal in its own right; the multiple objectives are not combined into a single (weighted) objective function. This allows the optimisation algorithm to explore the Pareto front of non-dominated solutions. Each of these non-dominated solutions denotes a possible assignment of requirements that maximizes all objectives without compromising on the maximisation of the others.

Hitherto, the only work on the MONRP has considered two possible bi-objective formulations, one in which the two objectives to be optimized are cost and value [33] and the other in which the two objectives are implementation-based and business-based [29]. However, no previous work has considered the problem of fairness analysis in requirement optimisation.

The problem of fairness in requirements allocation has two aspects:

1. What is a reasonable way to measure fairness?

2. To what extent can a solution be shown (to the stake holders) to be a fair allocation of requirements

These two aspects are interrelated and complicated by the fact that there is no single accepted notion of fairness. For example, an allocation might be deemed to be fair were it to satisfy the same number of requirements for each customer. However, this might be over simplistic; perhaps the solution should give each customer roughly equal value (as perceived by the customer) or, alternatively, roughly equal cost should be spent in implementing each customers' requirements.

This paper shows that using a multi-objective Pareto optimal search for optimal allocations of requirements, it is possible to treat each candidate notion of fairness as a separate optimisation objective in its own right. The paper shows that, using this multi objective approach, it is possible to explore the trade-offs between different notions of fairness and to attempt to locate solutions that balance these trade offs.

The result is feedback to the decision maker that serves two purposes: it allows the decision maker to see where there are potential problems in balancing concepts of fairness among customers and it allows the decision maker to demonstrate to the customer that the solution adopted is fair according to multiple fairness criteria.

In this way, the ability to automatically search for optimal regions of the 'fairness space' has applications in negotiation, mediation and conflict resolution during the requirements analysis process. It provides an unbiased and thorough exploration of trade offs and tensions within the multi-dimensional and complex space of customers and their requirements.

The primary contributions of the paper are as follows:

1. The paper gives several multi-objective formulations of fairness in requirements allocation.

2. The paper introduces a search based approach to explore the space of multiply fair allocations.

3. The paper reports results on the application of the search based optimisation approach to two real-world requirements data sets and to a series of synthetic data sets constructed to stress-test the approach.

The rest of the paper is organized as follows: In Section 2 the research problem is defined formally. Section 3 introduces the search algorithms studied and how they are tailored to the MONRP. Section 4 describes the experimental setup and environment. Section 5 presents the results of the experiments and discusses the findings. Section 6 describes the context of related work in which the current paper is located. Section 7 concludes.

2 Problem Formulation

This section gives definitions and characteristics of the MONRP problem as an extension of the traditional NRP model [2].

2.1 NRP Model

It is assumed that for an existing software system, there is a set of customers,

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

whose requirements are to be considered in the development of the next release of the software.

The set of possible software requirements is denoted by:

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

In order to satisfy each requirement, some resources need to be allocated. The resources needed to implement a particular requirement can be transformed into cost terms and considered to be the associated cost to fulfill the requirement.

Typically, these cost values are estimated, which is the case with the real world case studies presented below. The resultant cost vector for the set of requirements $r_i(1 \leq i \leq n)$ is denoted by:

$$Cost = \{cost_1, \ldots, cost_n\}$$

It is assumed that not all requirements are equally important for a given customer. The level of satisfaction for a given customer depends on the requirements that are satisfied in the next release of the software, which provide $value$ to the customers' organizations. Each customer $c_j(1 \leq j \leq m)$ assigns a *value* to requirement $r_i(1 \leq i \leq n)$ denoted by: $value(r_i, c_j)$ where $value(r_i, c_j) > 0$ if customer j desires implementation of the requirement i and 0 otherwise.

$$Value = \begin{Bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,i} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,i} & \cdots & v_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ v_{j,1} & v_{j,2} & \cdots & v_{j,i} & \cdots & v_{j,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m,1} & v_{m,2} & \cdots & v_{m,i} & \cdots & v_{m,n} \end{Bmatrix}$$

Each customer c_j has therefore, a subset of requirements that they expect to be satisfied denoted by R_j

$$\text{such that} \quad R_j \subseteq \Re, \quad \forall\, r \in R_j \quad value(r, c_j) > 0$$

The decision vector $\overrightarrow{x} = \{x_1, \ldots, x_n\} \in \{0, 1\}$ determines the requirements that are to be satisfied in the next release. In this vector, x_i is 1 if requirement i is selected and 0 otherwise. This vector denotes the solution to the problem.

2.2 Fairness in Requirements Assignments

Fairness is a deceptively simple concept; its implementation is complicated because the definition of fairness may have several equally valid, but possibly conflicting formulations. In order to capture and optimize fairness, a new aspect of the MONRP is explored: *Fairness in Requirement Assignments*. The principal motivation of fairness analysis is try to balance the requirement fulfillments between the customers. It could provide a convincing reference from the view of *marketing* and help the decision makers to maintain a record of fairness between the customers. It also may play a role in mediation, negotiation and dispute resolution.

Three factors are considered in this paper, namely, the number, the value and the cost of the requirements fulfilled for each customer. The aim is to calculate the absolute amount and the percentage of each factor that is present in a proposed MONRP solution. More formally, the three combinations studied in this paper are:

1. Fairness on absolute *number* of fulfilled requirements:

   $$\text{Maximize} \quad \overline{NA}$$

   $$\text{Minimize} \quad \sigma(NA)$$

   where $\overline{NA}$ is the mean value of the vector NA.

   The vector $NA = \{NA_1, \cdots, NA_m\}$ represents the absolute number of fulfilled requirements for each customer, where $NA_j = |R_j|$. Thus, the aim is to maximize the average absolute number of fulfilled requirements for all the customers whilst minimizing the standard deviation of the absolute number fulfilled requirements for each customer.

2. Fairness on absolute *value* of fulfilled requirements:

   $$\text{Maximize} \quad \overline{VA}$$

   $$\text{Minimize } \sigma(VA) \text{ where } VA_j = \sum_{i=1}^{n} value(r_i, c_j) \cdot x_i$$

   The vector $VA = \{VA_1, \cdots, VA_m\}$ represents the fulfilled value for each customer. In this vector, similarly, $VA_j(1 \leq j \leq m)$ is the j^{th} customer's fulfilled value:

   This objective function rewards solutions for which each customer obtains the same value. It penalizes solutions the more they depart from this equitable outcome.

3. Fairness on the percentage of *value* and *cost* of fulfilled requirements:

   The vector $Cost\_C = \{Cost\_C_1, \cdots, Cost\_C_m\}$ represents the costs of fulfilled requirement for each customer. In this vector, $Cost\_C_j(1 \leq j \leq m)$ is the j^{th} customer's fulfilled cost:

   $$Cost\_C_j = \sum_{i=1}^{n} cost_i \cdot x_i \quad \text{if} \quad r_i \in R_j$$

   The vector $VP = \{VP_1, \cdots, VP_m\}$ represents the percentage of fulfilled requirements' value for each customer.

   $$VP_j = \frac{VA_j}{\sum_{r \in R_j} value(r, c_j)} \times 100\%$$

   to minimize the standard deviation of spend on each of the customers,

   $$\text{Minimize} \quad \sigma(Cost\_C)$$

to minimize the standard deviation of the percentage of fulfilled value for customers,

$$\text{Minimize} \quad \sigma(VP)$$

to maximize the overall average fulfillment of each customers' objectives

$$\text{Maximize} \quad \overline{VP}$$

and finally to minimize the overall cost of the next release

$$\text{Minimize} \quad \sum_{i=1}^{n} cost_i \cdot x_i$$

3 Optimisation Algorithms

This section describes the search algorithms used in this paper. In the solution of Multi-Objective Optimisation Problems (MOOPs) there exist multiple and possibly conflicting objectives to be optimized simultaneously. There are various approaches to solve MOOPs. Among the most widely adopted techniques are: sequential optimisation, ϵ-constraint method, weighting method, goal programming, goal attainment, distance based method and direction based method. For a comprehensive study of these approaches, readers may refer to the survey by Szidarovsky et al. [31] and Collette and Siarry [7].

3.1 Pareto-Optimal Front

The Multi-Objective Optimisation Problem (MOOP) can be defined as the problem of finding a vector of decision variables $\overrightarrow{x}$, which optimizes a vector of M objective functions $f_i(\overrightarrow{x})$ where $i = 1, 2, \ldots, M$; subject to inequality constraints $g_j(\overrightarrow{x}) \geq 0$ and equality constraints $h_k(\overrightarrow{x}) = 0$ where $j = 1, 2, \ldots, J$ and $k = 1, 2, \ldots, K$. The objective functions are a mathematical description of performance criteria that are usually in conflict with each other [24].

Without loss of generality, a MOOP can be defined as follows:

$$\text{Maximize } \{f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_M(\overrightarrow{x})\}$$

subject to:

$$g_j(\overrightarrow{x}) \geq 0; \; j = 1, 2, \ldots, J$$

and

$$h_k(\overrightarrow{x}) = 0; \; k = 1, 2, \ldots, K.$$

where $\overrightarrow{x}$ is vector of decision variables; $f_i(\overrightarrow{x})$ is the i-th objective function; and $g(\overrightarrow{x})$ and $h(\overrightarrow{x})$ are constraint vectors.

These objective functions constitute a multi-dimensional space in addition to the usual decision space. This additional space is called the objective space, *Z*. For each solution $\overrightarrow{x}$ in the decision variable space, there exists a point in the objective space:

$$\overrightarrow{f}(\overrightarrow{x}) = Z = (z_1, z_2, \ldots, z_M)^T$$

In a Multi-Objective Optimisation Problem, we wish to find a set of values for the decision variables that optimizes a set of objective functions. A decision vector $\overrightarrow{x}$ is said to dominate a decision vector $\overrightarrow{y}$ (also written as $\overrightarrow{x} \succ \overrightarrow{y}$) iff:

$$f_i(\overrightarrow{x}) \geq f_i(\overrightarrow{y}) \;\; \forall \, i \in \{1, 2, \ldots, M\};$$

and

$$\exists \, i \in \{1, 2, \ldots, M\} \mid f_i(\overrightarrow{x}) > f_i(\overrightarrow{y}).$$

All decision vectors that are not dominated by any other decision vector are called *non-dominated* or Pareto-optimal and constitute the Pareto-optimal Front. These are solutions for which no objective can be improved without detracting from at least one other objective.

3.2 Characteristics

Among meta-heuristics, Evolutionary Algorithms (EAs) are particularly desirable to solve MOOPs, primarily because of their population-based nature. This enables them to capture the dominance relations in the population as a vehicle to guide the search towards Pareto-optimal front. They deal simultaneously with a set of possible solutions (the so-called population) which unlike traditional mathematical programming techniques, can find good approximations of Pareto-optimal set in a single run. Additionally, EAs are less susceptible to the shape or continuity of the Pareto-optimal front [5], whereas these two issues pose a barrier to classical mathematical programming techniques.

EAs usually contain several parameters that need to be 'tuned' for each particular application. For completeness, and to facilitate replicability, we give details of algorithmic tuning in Section 4.2. In addition, since the EAs are stochastic optimisation techniques, different runs tend to produce different results. Therefore, multiple runs of the same algorithm on a given problem are needed to statistically describe their performance on that problem. For a more detailed discussion of the application of EAs in multi-objective optimisation, the reader is referred to Coello et al. [6] and Deb [9].

To solve the MONRP, Multi-Objective EAs need to fulfill two primary roles:

1. Guiding the search towards the Pareto-optimal set to accomplish optimal or near-optimized solutions.

2. Maintaining a diverse population to achieve a well distributed non-dominated front, thereby fully exploring the solution space.

3.3 NSGA-II

The Non-dominated Sorting Genetic Algorithm-II (NSGA-II), introduced by Deb et al. [11] is an extension to an earlier Multi-Objective EA called NSGA developed by Srinivas and Deb [30]. The NSGA-II incorporates elitism to maintain the solutions of the best front found. The rank of each individual is based on the level of non-domination. The NSGA-II is a computationally efficient algorithm whose complexity is $O(mN^2)$, compared to NSGA with the complexity $O(mN^3)$, where *m* is the number of objectives and *N* is the population size.

The population is sorted using the non-domination relation into several fronts. Each solution is assigned a fitness value according to its non-domination level. In this way, the solutions in better fronts are given higher fitness values. The NSGA-II uses a measure of crowding distance to provide an estimation of the density of solutions belonging to the same front. This parameter is used to promote diversity within the population. Solutions with higher crowding distance are assigned a higher fitness compared to those with lower crowding distance, thereby avoiding the use of the fitness sharing factor with its associated computational cost [17].

Deb et al. [11] assumed that every individual i in the population has two attributes: non-domination rank (i_{rank}) and crowding distance $(i_{distance})$.

A partial order $\prec_n$ is defined as follows

$$i \prec j \text{ if } (i_{rank} < j_{rank})$$

$$\text{or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance}))$$

That is, between two solutions with differing non-domination ranks, the solution with the lower (better) rank is preferred. Otherwise, if both solutions belong to the same front, then the solution that is located in a less crowded region is preferred [11].

The algorithm can be described as follows. Initially, a random parent population P_0 with size *N* is created. Tournament selection, crossover, and mutation operators are used to create a child population Q_0 of size N [11]. The NSGA-II procedure executes the main loop described in Algorithm 1.

The NSGA-II algorithm was applied to the Fairness in Requirement Assignments Problem in order to identify Pareto front in different scenarios.

Algorithm 1: NSGA-II (main loop) Deb (2001)

```
1  while not stopping rule do
2      Let R_t = P_t ∪ Q_t
3      Let F = fast-non-dominated-sort (R_t)
4      Let P_{t+1} = φ and i = 1
5      while |P_{t+1}| + |F_i| ≤ N do
6          Apply
           crowding-distance-assignment(F_i)
7          Let P_{t+1} = P_{t+1} ∪ F_i
8          Let i = i + 1
9      end
10     Sort(F_i, ≺_n)
11     Let P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)]
12     Let Q_{t+1} = make-new-pop(P_{t+1})
13     Let t = t + 1
14 end
```

4 Experimental Set Up

4.1 Data Sets

This section describes the test data sets used to fulfill the research tasks of fairness analysis in requirements assignments. There are three data sets used in our experiments.

The first data set is generated randomly with 30 requirements and 5 customers according to the problem model. The values and costs are assigned as follows: random choices were made 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 customer places no value on, i.e. does not want, this requirement). This simulates the situation where a customer 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 authors' experience indicates that customers prefer such a coarse grained scale. While a finer level of granularity may be more theoretically interesting for the research purposes, in practice customers are uncomfortable with such fine-grained value assignments.

The second data set is taken from Motorola [3] as shown in Table 1. The Motorola data set has 4 customers and 35 requirements.

Table 2 shows the third data set that is taken from Greer 2004 [16]. The Greer data set has 5 customers and 20 requirements. Greer's data does not contain information about the cost of each requirement. For the purpose of feeding this useful industrial data into our algorithm, the cost of the requirements were generated randomly within the range from 10 to 1100, following a Gaussian distribution.

Table 1: Feature Data from Motorola

r_1	r_2	r_3	r_4	r_5	r_6	r_7
100	50	300	80	70	100	1000
r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}
40	200	20	1100	10	500	10
r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}
10	10	20	200	1000	120	300
r_{22}	r_{23}	r_{24}	r_{25}	r_{26}	r_{27}	r_{28}
50	10	30	110	230	40	180
r_{29}	r_{30}	r_{31}	r_{32}	r_{33}	r_{34}	r_{35}
20	150	60	100	400	80	40

4.2 Algorithmic Tuning

The algorithm was run for a maximum of 10,000 function evaluations. The algorithm was executed 20 times for each data set. The initial population was set to 200. A simple binary GA encoding was used, with each bit to code for a decision variable (the inclusion or exclusion of a requirement). The length of a chromosome is thus equal to the number of requirements. Each experimental execution of algorithms was terminated after 50 generation (i.e. after 10,000 evaluations). The genetic approach used the tournament selection (with tournament size of 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). Readers may refer to Goldberg [15] for detailed information about GAs and also to Deb [9] and Coello et al. [6] for a comprehensive review of multi-objective evolutionary algorithms.

Table 2: Feature Data Set taken from Greer 2004

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}
c_1	4	2	1	2	5	5	2	4	4	4
c_2	4	4	2	2	4	5	1	4	4	5
c_3	5	3	3	3	4	5	2	4	4	4
c_4	4	5	2	3	3	4	2	4	2	3
c_5	5	4	2	4	5	4	2	4	5	2
	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}
c_1	2	3	4	2	4	4	4	1	3	2
c_2	2	3	2	4	4	2	3	2	3	1
c_3	2	4	1	5	4	1	2	3	3	2
c_4	5	2	3	2	4	3	5	4	3	2
c_5	4	5	3	4	4	1	1	2	4	1

5 Results and Analysis

In this section, we present different fairness models in requirement assignments and the results of applying the NSGA-II algorithm to different problem instances. Three experiments were conducted and the results shown in Figure 1, 2 and 3 respectively. In order to demonstrate the evolutionary process of the NSGA-II algorithm, the initial populations, the populations generated by the median generation and the final non-dominated solutions were plotted in the figures. Each point represents a subset of requirements for the next release. The small '•', '$*$' and solid '▲' denote the increasingly better solutions found. Therefore, the algorithm's progress towards the final Pareto front produced is visualized by increasingly darker and larger points.

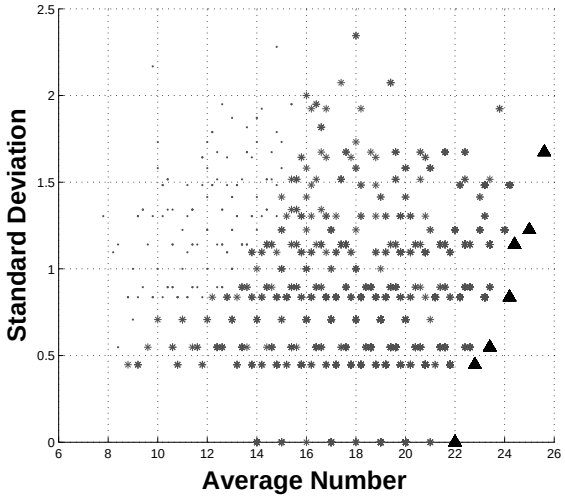
The results of the first experiment are shown in Figure 1 where all the populations are plotted for the three data sets. In this experiment, the two objectives are: a) minimize the standard deviation of the absolute number of fulfilled requirements for each customer and b) maximize the overall average number of fulfilled requirements for all customers.

We observe that the search techniques guide the population towards the Pareto front. The optimal fronts are shown in the results for both random and the Motorola data set. On these two fronts, the standard deviation of fulfilled requirements increases with overall average number. This implies that the more requirements are fulfilled, the less fairness is provided to the customers. This is partly because the customers in these two data sets demand different numbers of requirements. As the number of the selected requirements increases, it becomes easier for the algorithm to adjust the allocations of fulfilled requirements to different customers to obtain a lower standard deviation (more fairness). The most top-right solid '▲' on the fronts denotes the solutions in which all requirements for the customers are fulfilled.

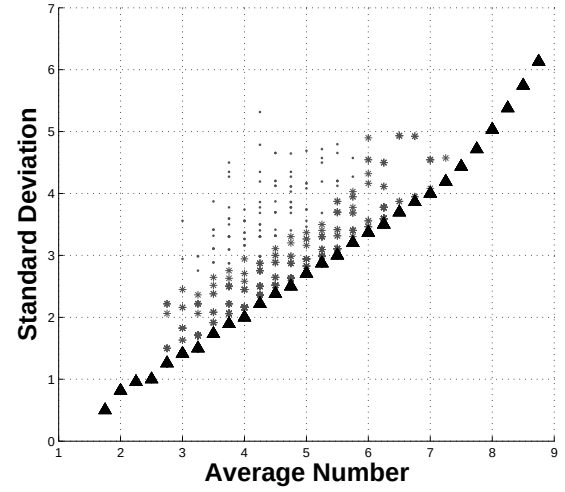
In Figure 1(a), the eight '$*$' along the X–axis with zero standard deviation show that NSGA–II is able to obtain subsets of requirements that fulfill each customer with the same number of requirements. However, in Figure 1(b), we cannot observe this sort of "perfectly-fair" solution. This is because of the difference between the sparsity pattern of the Customer-Requirement matrix of these two data sets.

In the Motorola data set, every requirement is demanded by only one customer exclusively, and the forth customer requests only a single requirement. This pattern dramatically increases the difficulty for NSGA–II to obtain the only "perfectly-fair" solution that fulfills each customer with only one requirement.

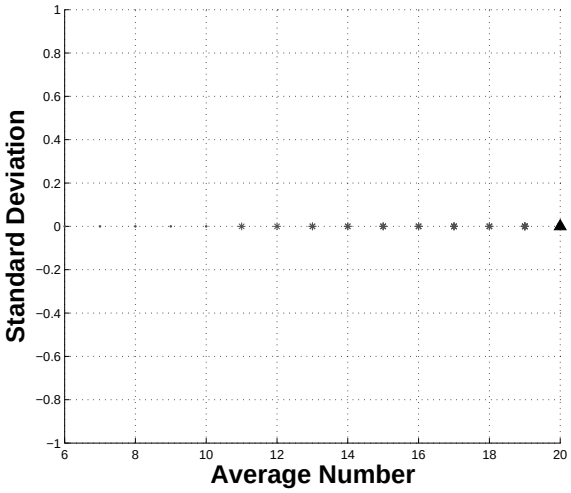
On the other hand, the result for the Greer data set shows the standard deviation remains at zero throughout the search. This is also because of the distribution of the data, which, in this case is perfectly uniform. That is, in the Greer data set, every customer requests every requirement,



(**a**) Result for Random Data Set

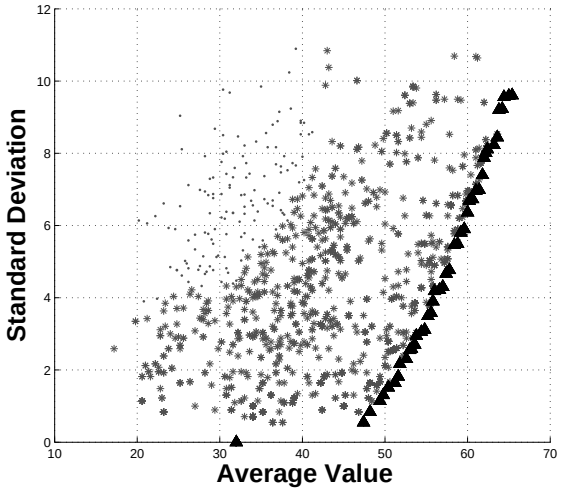


(**b**) Result for Motorola Data Set

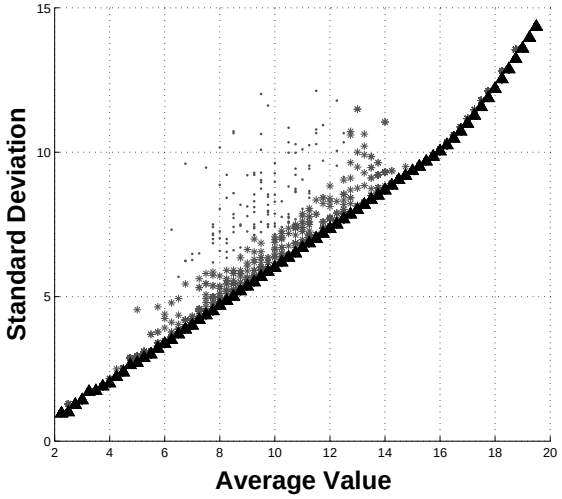


(**c**) Result for Greer Data Set

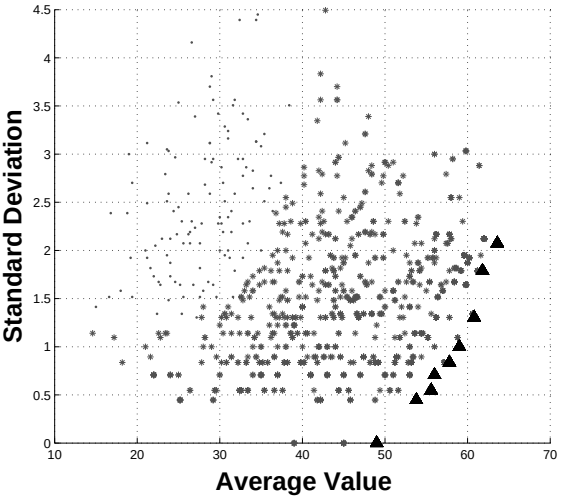
Figure 1: Results of Fairness on Absolute *Number* of Fulfilled Requirements



(**a**) Result for Random Data Set

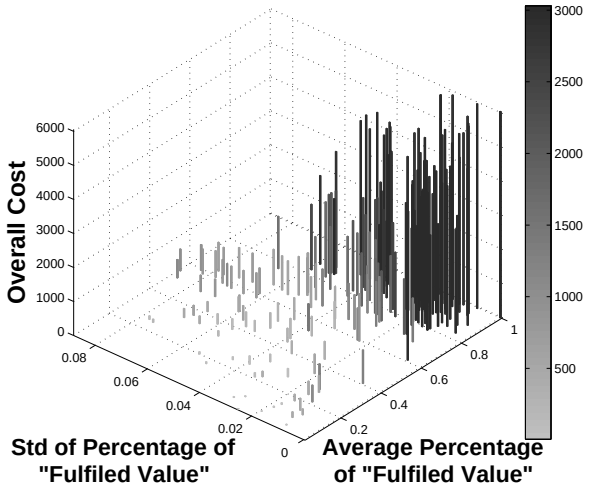


(**b**) Result for Motorola Data Set

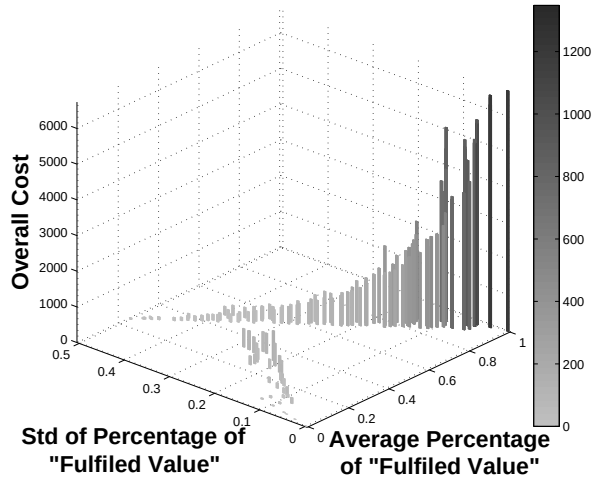


(**c**) Result for Greer Data Set

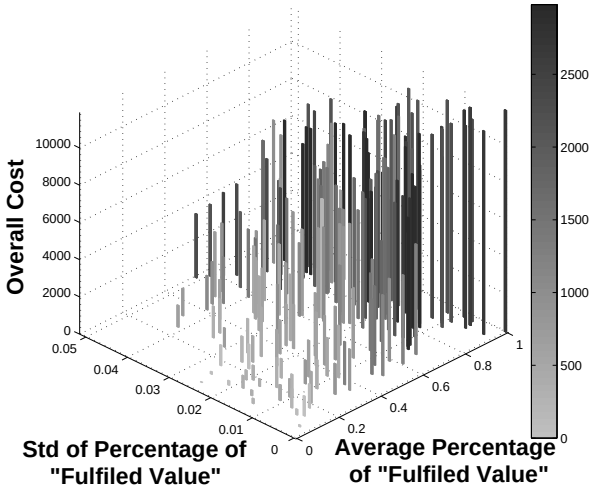
Figure 2: Results of Fairness on Absolute *Value* of Fulfilled Requirements



(**a**) Result for Random Data Set



(**b**) Result for Motorola Data Set



(**c**) Result for Greer Data Set

Figure 3: Results of Fairness on *Percentage* of Fulfilled Value and Cost

so all customers would have an equal number of fulfilled requirements, no matter which requirements are selected in the next release.

Figure 2 illustrates the results for the second experiment in which the two objective functions are: a) minimize the standard deviation of the absolute value of fulfilled requirements for each customer and b) maximize the overall average value of fulfilled requirements for all the customers. On the fronts of these results, a similar trend is observed: the degree of fairness decreases as the overall coverage increases.

In the third experiment, information on the cost of the requirements is taken into account. This allows us to obtain fairness information within different budget constraints. Four objectives are considered: a) minimizing the overall cost of the next release, b) minimizing the standard deviation of the cost spent on each customer, c) minimizing the percentage of fulfilled value for each customer and d) maximizing the overall average fulfilled value for all customers.

Here, we consider the fairness on both cost and value simultaneously. The results are plotted in Figure 3. It is something of a challenge to visualize a four-dimensional solution space in a two-dimensional figure. In this figure, each bar represents an optimal solution on the Pareto front. The location of each bar in the (x, y) plane shows the average fulfilled value for all customers and the standard deviation of fulfilled value for each customer respectively. The height of each bar shows the overall cost for each optimal solution. The standard deviation of the cost spent on each customer is shown by the gray scale of each bar.

From the results for all the data sets, it can be seen that as the overall fulfilled value increases along the X–axis, the standard deviation of cost spend on the customers also increases. This observation replicates the previous experiments reported in this paper.

There is also an interesting observation in Figure 3(b). There are no solutions in the 'empty triangle' area around 50% fulfillment on the average value. The reason for this lies in the fact that the fourth customer in the Motorola data set only requests a single requirement. Thus, the percentage of fulfilled value for this customer has to be either 0% or 100%. Consider those solutions on the edge of this triangle, when the overall percentage is growing between 0% and 50%, the fulfilled value for this customer stays at 0%. This is because the other customer's fulfillment is below 50%, if the fourth customer has 100% fulfillment then the standard deviation will increase and the solution will leave the edge. Thus, on the edge of the triangle leading up to 50% overall fulfillment, the standard deviation must increase if one of the customer's fulfillment remains at zero while the other customer's fulfillment increases.

The experiments show that as more requirements are fulfilled, less fairness is provided to the customers. This is

partly due to the high variation in the customers' number of requirements in the examined data sets. However, fortunately as the number of the selected requirements increases, the algorithm has more scope in which to search for optimally fair solutions. It was also observed that the quality of final solutions in terms of fairness is partly dependent upon the sparsity pattern of the Customer-Requirement matrices. This is also the case for the search algorithm, i.e. sparser customer-requirement matrixes tend to make problem more difficult for the search algorithm.

6 Related Work

In the area of requirements engineering, several related studies have been proposed for requirements analysis and optimisation. Karlsson [20, 21, 27] provided the methodologies for assigning priorities to requirements and developing strategies for selecting an optimal set of requirements for implementation. The Focal Point tool (marketed by Telelogic) is based on this work.

Bagnall et al. (2001) [2] suggested the term *Next Release Problem* for requirements planning and described the various metaheuristic algorithms to find a high quality, but possibly suboptimal, solution to balance customer requests. Van den Akker [32] study a variation of the problem using integer linear programming to find exact solutions within budgetary constraints.

Zhang et al. [33] considered value and cost as two separate criteria in their multi-objective next release problem (MONRP) formulation. They consider an integrated value function, comprising of the values associated with each customer using search-based techniques. Greer and Ruhe [16] address software release planning by minimizing total penalty and maximising total benefit in the form of an integrated objective function with user defined weights for each objective.

Problems associated with multiple customers with completing and conflicting view points has been known for some time [23]. Hoh In at el. [18, 19] proposed the WinWin model to help the stakeholders' negotiation process based on Multi-Criteria preference analysis. Another approach to resolve stakeholder conflicts is the ViewPoint approach [12, 13], which separates the different opinions among the stakeholders and can detect conflicts automatically. In the stakeholder requirements analysis problem, Robinson at el. [1, 25] worked on a requirements negotiation model which provided automated support to generate requirements resolutions.

However, the present paper is the first to introduce techniques for analysis of the trade-offs between different customers' notions of fairness in requirement allocation, where there are multiple customers with potentially conflicting requirement priorities and also possibly different views of what would constitute fair and equitable solution.

Evolutionary multicriteria optimisation has traditionally concentrated on problems comprising 2 or 3 objectives. Our formulation comprises a relatively large number of objectives. Such problems pose new challenges for algorithm design, visualization and implementation. In multi-objective evolutionary search the populations are likely to be largely composed of non-dominated solutions.

Fleming et al. [14] use progressive articulation of design preferences to assist in reducing the region of interest for the search and, thereby, simplifying the problem.

Corne and Knowles [8] compare a number of ranking methods to address the shortcoming of existing evolutionary algorithms for many-objective optimisation.

Deb and Kumar [10] suggest an interactive method to incorporate user preferences in guiding the multi-objective search. The idea is to reduce the search space by focusing on the more favourable regions of the Pareto front. This approach has potential application in the multi-objective next release problem provided that the user is prepared to identify their preferences during the search.

Though other authors have considered conflicts and negotiations, the present paper is the first to address the issue of "fairness" in requirements analysis.

7 Conclusions

The paper introduces the concept of fairness in requirements analysis and optimisation using a new formulation of Multi-Objective Next Release Problem. Three fairness models were introduced to balance the requirements fulfillments between the customers.

The work reported here is the first to address the issue of fairness balance among different definitions of fairness. The formulations adopted cover simplified scenarios. However, even with the relatively simple formulations adopted in this paper, it has been possible to use search based optimization technique to reveal tensions between fairness definitions.

The experiments upon which this paper reports demonstrate that search based techniques can be applied to real world data sets and illustrate the way in which they reveal hidden tensions implicit in these data sets. However, more work is required to extend and evaluate the techniques.

Future work will focus on adapting the search based formulations to cater for more 'messy' real world scenarios in which requirements are partially unclear and subject to change and for which the domain specific parameters are both constrained and subject to estimation error. These augmented scenarios would pose a significant challenge to any approach. However, there are grounds for optimism. We are also concerned to find ways to package and deliver our approach in such a way that they can be used by working requirements engineers in the context of existing tool sets.

References

[1] Automated support for requirements negotiation, Mar. 17 1994.

[2] A. Bagnall, V. Rayward-Smith, and I. Whittley. The Next Release Problem. *IEE Proceedings - Software*, 43(14):883–890, Dec 2001.

[3] P. Baker, M. Harman, K. Steinhöfel, and A. Skaliotis. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In *22nd International Conference on Software Maintenance (ICSM 06)*, pages 176–185, Philadelphia, Pennsylvania, USA, September 24-27 2006.

[4] P. Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 7(3):139–151, 2002.

[5] C. A. Coello Coello. Evolutionary multiobjective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, Feb. 2006.

[6] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.

[7] Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer, 2004.

[8] D. Corne and J. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *GECCO'07: Proceedings of Genetic and Evolutionary Computation Conference*, pages 773–780, 2007.

[9] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[10] K. Deb and A. Kumar. Interactive evolutionary multiobjective optimization and decision-making using reference direction method. In *GECCO'07: Proceedings of Genetic and Evolutionary Computation Conference*, pages 781–788, 2007.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr. 2002.

[12] S. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *Proc. 23rd International Conference on Software Engineering*, pages 411–420. IEEE Computer Society Press, May 2001.

[13] S. Easterbrook, A. Finkelstein, J. Kramer, and B. Nuseibeh. Co-ordinating distributed ViewPoints: the anatomy of a consistency check. Technical Report 333, School of Cognitive and Computing Sciences, University of Sussex, UK, 1994.

[14] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-Objective Optimization: An Engineering Design Perspective. *Lecture Notes in Computer Science*, 3410:14–32, 2005.

[15] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[16] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information & Software Technology*, 46(4):243–253, 2004.

[17] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report IllIGAL 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, IL, 1993.

[18] H. In, B. Boehm, T. Rodgers, and M. Deutsch. Applying winwin to quality requirements: A case study. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 555–564. IEEE Computer Society Press, May 2001.

[19] H. In, D. Olson, and T. Rodgers. A requirements negotiation model based on multi-criteria analysis. In *RE*, pages 312–313. IEEE Computer Society, 2001.

[20] J. Karlsson. Software requirements prioritizing. In *ICRE*, pages 110–116, 1996.

[21] J. Karlsson and K. Ryan. Supporting the selection of software requirements. In *IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design*, page 146, Washington, DC, USA, 1996. IEEE Computer Society.

[22] L. Karlsson, B. Regnell, and T. Thelin. Case studies in process improvement through retrospective analysis of release planning decisions. *International Journal of Software Engineering and Knowledge Engineering*, 16(6), 2006.

[23] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specifications. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994.

[24] A. Osyczka. In *Multicriteria optimization for engineering design*, pages 193–227. Design Optimization, 1985.

[25] W. N. Robinson and V. Volkov. Requirement conflict restructuring, Aug. 25 1999.

[26] G. Ruhe and M. O. Saliu. The art and science of software release planning. *IEEE Software*, 22(6):47–53, 2005.

[27] K. Ryan and J. Karlsson. Prioritizing software requirements in an industrial setting. In *ICSE*, pages 564–565, 1997.

[28] M. O. Saliu and G. Ruhe. Software release planning for evolving systems. *ISSE*, 1(2):189–204, 2005.

[29] M. O. Saliu and G. Ruhe. Bi-objective release planning for evolving software systems. In *Proceedings of ESEC/SIGSOFT FSE 2007*, 2007.

[30] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[31] F. Szidarovsky, M. E. Gershon, and L. Dukstein. *Techniques for multiobjective decision making in systems management*. Elsevier, New York, 1986.

[32] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization and what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.

[33] Y. Zhang, M. Harman, and S. A. Mansouri. The multi-objective next release problem. In *GECCO'07: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1129–1136. ACM Press, 2007.