Requirements Eng
DOI 10.1007/s00766-009-0075-y

RE'08 SPECIAL ISSUE

A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making

Anthony Finkelstein · Mark Harman · S. Afshin Mansouri · Jian Ren · Yuanyuan Zhang

Received: 19 October 2008 / Accepted: 13 January 2009

Abstract 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. The paper also reports on experiments to determine the most suitable algorithm for this problem, comparing the results of the NSGA-II algorithms, a widely used multi objective evolutionary algorithm, and the Two-Archive evolutionary algorithm, a recently proposed alternative.

Keywords Pareto optimality · Fairness analysis · Requirements assignment · Multi-objective genetic algorithms · Decision making

A. Finkelstein (✉)
University College London, Malet Place,
London WC1E 6BT, UK
e-mail: a.finkelstein@cs.ucl.ac.uk

M. Harman · J. Ren · Y. Zhang
King's College London, Strand, London WC2R 2LS, UK

M. Harman
e-mail: mark.harman@kcl.ac.uk

J. Ren
e-mail: jian.ren@kcl.ac.uk

Y. Zhang
e-mail: yuanyuan.zhang@kcl.ac.uk

S. A. Mansouri
Brunel University, Uxbridge, Middlesex UB8 3PH, UK
e-mail: afshin.mansouri@brunel.ac.uk

1 Introduction

This paper is concerned with fairness. That is, to what extent is it possible to say that an allocation of requirements to customers is fair, when there are multiple customers, each with their own idea of what the next set of requirements should be.

Of course, as soon as one embarks upon a discussion of fairness, the issue of what one means by fairness must be addressed. What might appear to be manifestly fair to one person, may seem exceedingly unjust to another. In requirements engineering, there are several possible ways in which fairness can be defined. Rather than picking one of these in a somewhat arbitrary fashion, this paper advocates a multi objective search based framework, in which a requirements engineer can incorporate all proposed models of fairness.

The framework proposes that each notion of fairness should form an objective in a multi objective, Pareto optimal Search Based Software Engineering (SBSE) setting. Pareto optimality is well-suited to this scenario, because it makes no assumptions about which objective takes priority. As will be seen, the approach advocated in the paper can be used to explore the extent to which a solution can be fair according to all definitions of fairness. The optimizing approach can also be said to reveal the inherent tensions and trade offs between the different notions of fairness.

In this way, SBSE is used not to provide an optimal solution to some particular instantiation of the choice of fairness. Rather, it is used to provide insight by exploring the space of possible solutions and the relationships between them. The aim of this work is thus not to replace a decision maker with an automated tool that allocates requirements, but to provide a new form of decision

Published online: 10 February 2009

support; one that searches for optimal balances, guiding the decision maker. The approach can be used to reveal the structure of the optimization problems that characterise the difficult balancing act faced by the requirements engineer.

The paper concerns the requirements analysis setting in which there are many customers, each with competing (and possibly conflicting interests). This is an increasingly prevalent scenario 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 paper also uses a data set from a previous study by Greer and Ruhe, together with synthetically created data that explore the behaviour of the optimization algorithms used over a range of possible data configurations.

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.

To address the fairness analysis 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 metaheuristic 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) [5, 27] and Release Planning [13, 33–35, 46–48]. 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 [5, 27, 53]. 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 [49, 56]. 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 [56] and the other in which the two objectives are implementation-based and business-based [49]. 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 where 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 addresses these issues. It is the first to introduce techniques for analysis of the trade-offs between different customers' notions of fairness in requirements allocation, where there are multiple customers with potentially conflicting requirement priorities and also possibly different views of what would constitute fair and equitable solution.

The 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.

Evolutionary multicriteria optimisation has traditionally concentrated on problems comprising two or three 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.

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 synthetic data set constructed to stress-test the approach.
4. The paper reports the results of a set of experiments which explore the suitability of two potentially applicable multi-objective evolutionary algorithms: the widely used NSGA-II, and the more recently proposed Two-Archive algorithm.
5. As a validation of the overall approach, the paper reports the results of a comparison with Random Search, showing the the results obtained by the 'intelligent' search based optimizing approaches significantly outperform Random Search.

The rest of the paper is organized as follows: in Sect. 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 on real world an synthetic data for two and four-objective instantiations of the problem using the NSGA II algorithm. Section 6 presents a comparison of NSGA-II, the Two-Archive algorithm and Random Search. Section 7 considers threats to validity of the findings. Section 8 describes the context of related work in which the current paper is located. Section 9 draws the conclusion and future work.

2 Problem formulation

This section gives definitions and characteristics of the MONRP problem as an extension of the traditional NRP model [5].

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 such that

$R_j \subseteq \Re, \quad \forall r \in R_j \quad value(r, c_j) > 0$

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:

   Maximize $\overline{NA}$

   Minimize $\sigma(NA)$

   where $\overline{NA}$ is the mean value of the vector NA.
   The vector $NA = \{NA_1, \ldots, NA_m\}$ 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:

   Maximize $\overline{VA}$

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

   The vector $VA = VA_1, \ldots, VA_m$ represents the fulfilled value for each customer. In this vector, similarly, $VA_j (1 \leq j \leq m)$ is the jth 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 $Cost\_C = \{Cost\_C_1, \ldots, Cost\_C_m\}$ represents the costs of fulfilled requirement for each customer. In this vector, $Cost\_C_j (1 \leq j \leq m)$ is the jth customer's fulfilled cost:

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

   The vector $VP = \{VP_1, \ldots, VP_m\}$ 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,

   Minimize $\sigma(Cost\_C)$

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

   Minimize $\sigma(VP)$

   to maximize the overall average fulfillment of each customers' objectives

   Maximize $\overline{VP}$

   and finally to minimize the overall cost of the next release

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

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. [52] and Collette and Siarry [16].

3.1 Pareto-optimal front

The multi-objective search algorithms used in the paper are based on the concept of dominance to solve MOOPs. In the algorithms, two solutions are compared on the basis of

whether one dominates the other solution or not [18]. We describe the domination concept below:

In a Multi-Objective Optimisation Problem, each solution here is defined as a vector of decision variables $\vec{x}$. It is assumed that there are M objective functions $f_i(\vec{x})$ where $i = 1, 2, \ldots, M$. The objective functions are a mathematical description of performance criteria. Often these criteria are in conflict with each other [43].

We wish to find a set of values for the decision variables that optimizes a set of objective functions. A decision vector $\vec{x}$ is said to dominate a decision vector $\vec{y}$ (also written as $\vec{x} \succ \vec{y}$) iff:

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

and

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

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.

The goal of the Multi-Objective Optimisation Problem (MOOP) is to find an ideal vector of decision variables $\vec{x}$, which optimizes a vector of M objective functions $f_i(\vec{x})$ subject to inequality constraints $g_j(\vec{x}) \geq 0$ and equality constraints $h_k(\vec{x}) = 0$ where $j = 1, 2, \ldots, J$ and $k = 1, 2, \ldots, K$.

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

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

subject to:

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

and

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

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

These objective functions constitute a multi-dimensional space which is called the objective space, Z. For each solution $\vec{x}$ in the decision variable space, there exists a point z^* in the objective space:

$$\exists z^* \in Z$$

$$z^* = \vec{f}(\vec{x}) = (f_1, f_2, \ldots, f_M)^T.$$

3.2 Evolutionary algorithms

Metaheuristics are a family of approximate optimisation techniques for solving the computational problem, which have received increasing attention in recent years. Among metaheuristics, 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 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 [14].

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 Sect. 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. [15] and Deb [18].

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 The NSGA-II algorithm

The Non-dominated Sorting Genetic Algorithm-II (NSGA-II), introduced by Deb et al. [20] is an extension to an earlier multi-objective EA called NSGA developed by Srinivas and Deb [51]. 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 [30].

Deb et al. [20] assumed that every individual i in the population has two attributes: non-domination rank (i_{rank}) and crowding distance (i_{distance}).

A partial order $\prec$ is defined as follows

$i \prec j$ if $(i_{\text{rank}} < j_{\text{rank}})$ or
$((i_{\text{rank}} = j_{\text{rank}})$ and $(i_{\text{distance}} > j_{\text{distance}}))$

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 [20].

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 [20]. The NSGA-II procedure executes the main loop described in Algorithm 1.

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

```
1  while t ≤ max_generation 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, ≺)
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
```

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

3.4 The Two-Archive algorithm

The Two-Archive algorithm was introduced by Praditwong and Yao [44]. It substitutes the new dominated solutions for the existing dominated solutions introduced by PESA [37]. Truncation is performed at the end of archiving non-dominated individuals, as it is in NSGA-II [20] and SPEA2 [57].

In the algorithm, there are two archives used to collect and record candidate solutions in the population: the Convergence Archive (CA) and the Diversity Archive (DA). If a non-dominated solution selected from the population dominates other members in the archives, it enters the CA and the dominated members are deleted, otherwise it enters the DA without deleting any members in the archives.

The total size of CA and DA is fixed, but the size of each archive varies. When the number of members in the archives exceeds the capacity of archives, the members of the DA with the shortest distance to the members in the CA are deleted until the total size falls within the threshold.

The details of the Two-Archive algorithm are shown by Algorithm 2 and Algorithm 3, which describe the top level and archiving processes, respectively.

Algorithm 2: The Two-Archive Algorithm (the main loop) Praditwong and Yao (2006)

```
1  Initialise the population
2  Initialise archives to the empty set
3  Evaluate initial population
4  Set t = 0
5  repeat
6      Archive non-dominated individuals
7      Select parents from archives
8      Generate new population using genetic
       operator
9      Evaluate the new population
10     t = t + 1
11 until t == MAX_GENERATION;
```

3.5 Random Search

The Random Search technique was also applied to the MONRP. This is merely a 'sanity check'; all metaheuristic algorithms should be capable of comfortably outperforming Random Search for a well-formulated optimization problem.

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.

Algorithm 3: Archive Non-Dominated Individuals Praditwong and Yao (2006)

```
for i = 1 to popsize do
    if individual(i) is non-dominated solution then
        if no member in both archives can dominate an individual(i) then
            Set individual(i).Dflag = 0
            if individual(i) dominates any member in both archives then
                Set individual(i).Dflag = 1
                Delete dominated member
            end
            if member(i).Dflag == 1 then
                Add individual(i) to Convergence Archive(CA)
            else
                Add individual(i) to Diversity Archive(DA)
            end
        end
    end
end
// Remove Strategy
if sizeof(CA) + sizeof(DA) > limit then
    for i = 1 to sizeof(DA) do
        DA(i).length = maxreal
        for j = 1 to sizeof(CA) do
            if DA(i).length > Dist(CA(j),DA(i)) then
                DA(i).length = Dist(CA(j),DA(i))
            end
        end
    end
    repeat
        Delete the member of DA with the shortest length
    until sizeof(DA) + sizeof(CA) == limit;
end
```

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 [6] 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 [27]. The Greer data set has 5 customers and 20 requirements. Greer's data do 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.

4.2 Algorithmic tuning

Each algorithm was run for a maximum of 10,000 function evaluations. The number of executions of each algorithm was 30 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 the probability of the crossover operator being applied was set to $P_c = 0.8$ and the probability of the mutation operator (per gene) to $P_m = 1/n$ (where n is the number of all possible requirements). In the Two-Archive algorithm, the total capacity of the archives was 200. The algorithm selects parents from both archives to the mating pool. An archive was chosen with a

Table 1 Feature data from Motorola

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

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

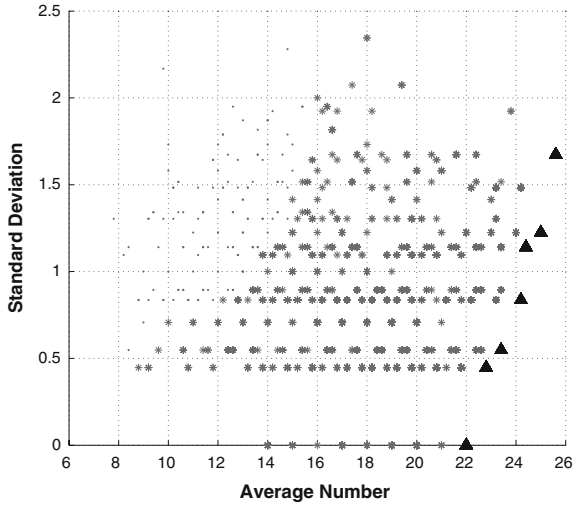
probability that is set to 0.6. The crossover and mutation probability in the Two-Archive algorithm are the same as NSGA-II's. Readers may refer to Goldberg [26] for detailed information about GAs and also to Deb [18] and Coello et al. [15] for a comprehensive review of multi-objective evolutionary algorithms.

5 Results and analysis for two and four objective formulations

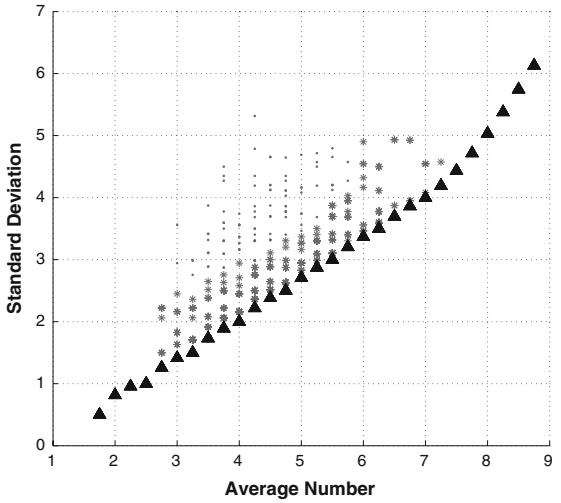
In this section, we present different fairness models in requirement assignments and the results of applying the NSGA-II algorithm to different problem instances. Three experiments were conducted and the results shown in Figs. 1, 2 and 3, respectively. In order to demonstrate the evolutionary process 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 Fig. 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.

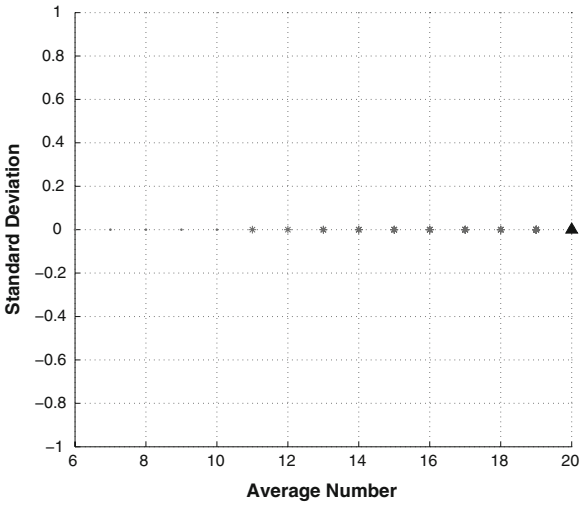
We observe that the search techniques guide the population towards the Pareto front. The optimal fronts are shown in the results for both random and the Motorola data set. On these two fronts, the standard deviation of fulfilled



(a) Result for Random Data Set

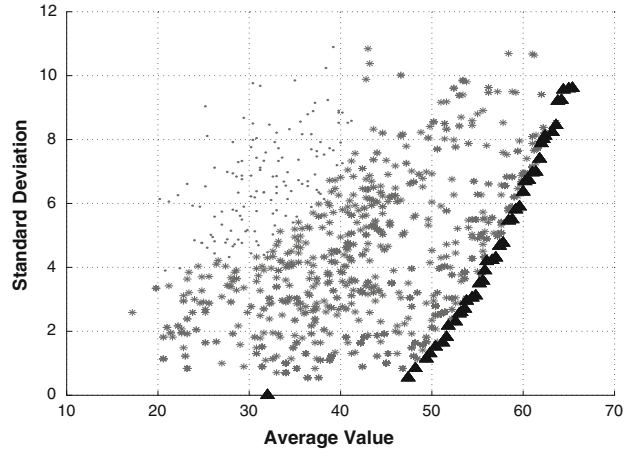


(b) Result for Motorola Data Set

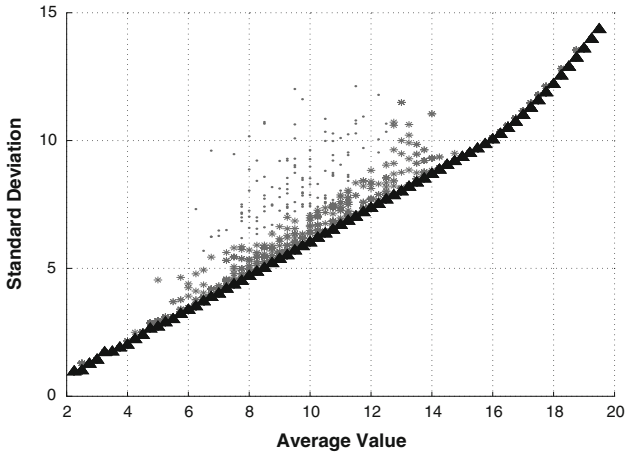


(c) Result for Greer Data Set

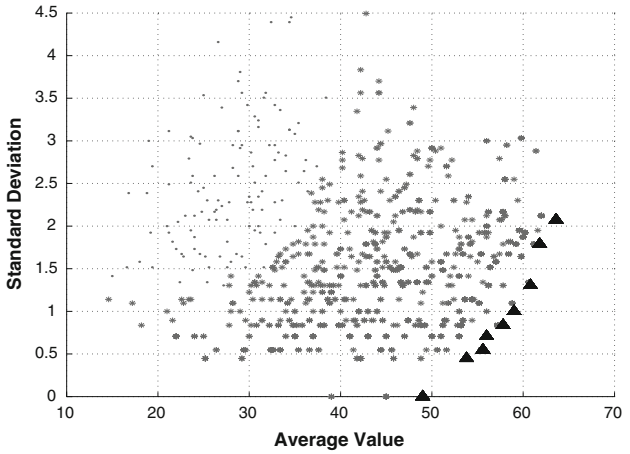
Fig. 1 Results of fairness on absolute *number* of fulfilled requirements



(a) Result for Random Data Set

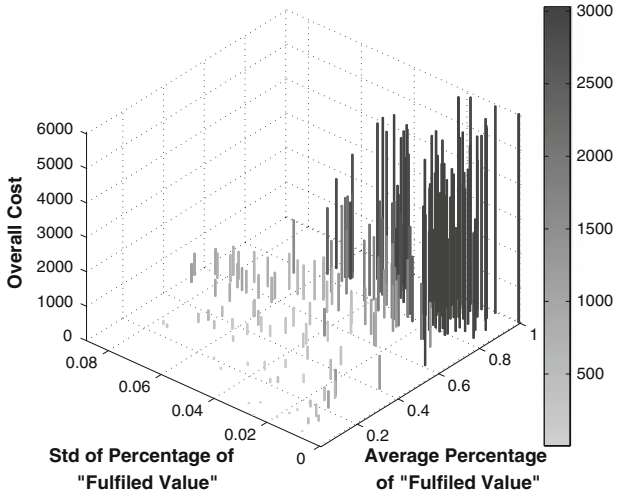


(b) Result for Motorola Data Set

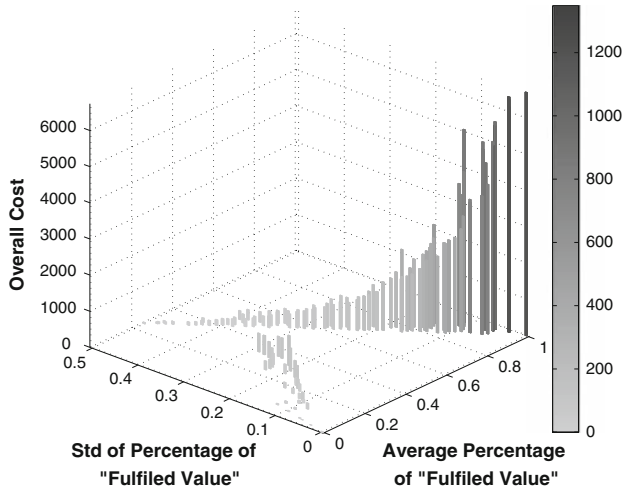


(c) Result for Greer Data Set

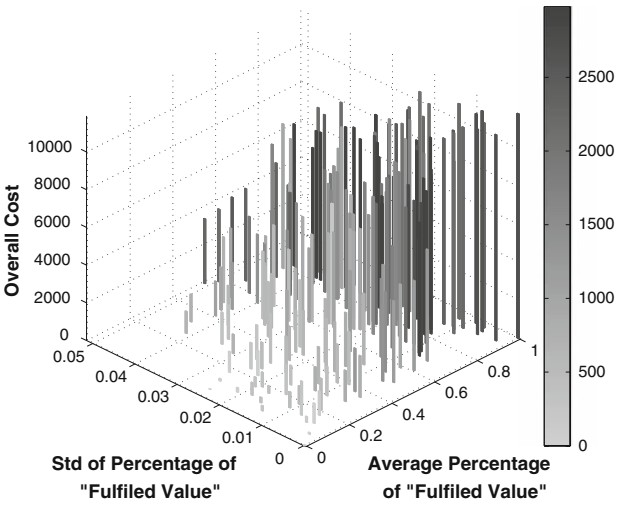
Fig. 2 Results of fairness on absolute *value* of fulfilled requirements



(a) Result for Random Data Set



(b) Result for Motorola Data Set



(c) Result for Greer Data Set

Fig. 3 Results of fairness on *percentage* of fulfilled value and cost

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 Fig. 1a, 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 Fig. 1b, 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, 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 Fig. 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.

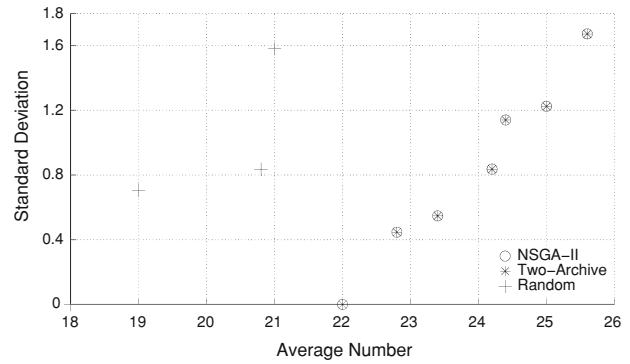
From the results for all the data sets, it can be seen that as the overall fulfilled value increases along the *X*-axis, the standard deviation of cost spend on the customers also increases. This observation replicates the previous experiments reported in this paper.

There is also an interesting observation in Fig. 3b. 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.

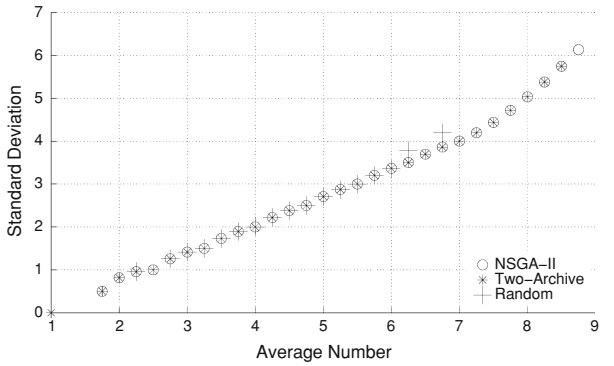
The experiments show that as more requirements are fulfilled, less fairness is provided to the customers. This is partly due to the high variation in the customers' number of requirements in the examined data sets. However, fortunately as the number of the selected requirements increases, the algorithm 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 Results of the experiment to compare algorithm performance

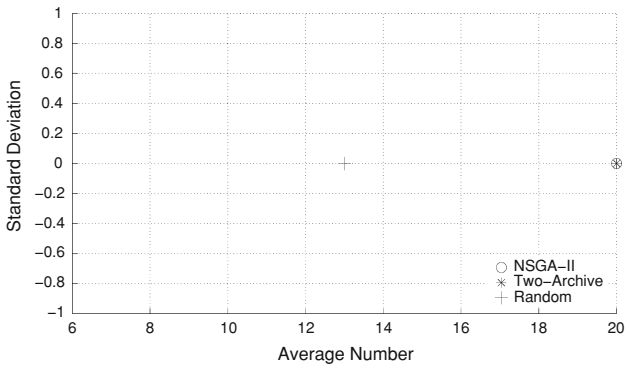
In this section, we present the results of applying the different optimisation algorithms to different problem instances. They are NSGA-II, Two-Archive and Random Search. In order to compare their performances, two experiments were conducted concerned with fairness on the percentage of fulfilled requirements and the number of fulfilled requirements. The results shown in Figs. 4 and 5, respectively. Each algorithm was executed 30 times for each data set. The final non-dominated solutions for each technique are represented by the '○', '*' and '+' symbols plotted in the figures. These solutions are the best ones over



(a) Result for Random Data Set

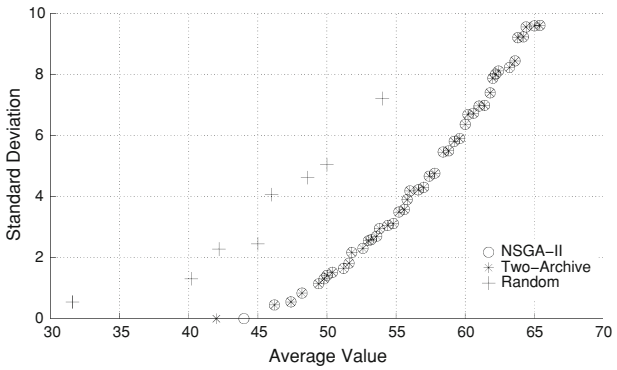


(b) Result for Motorola Data Set

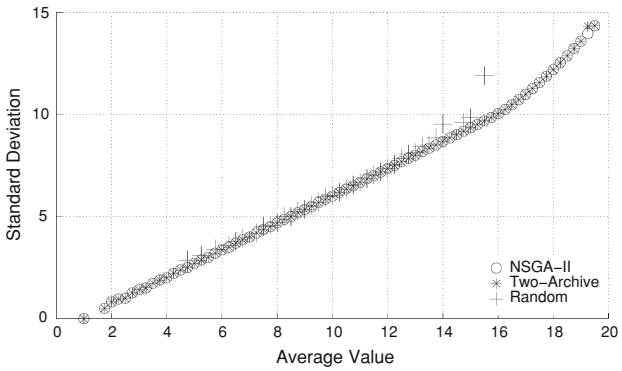


(c) Result for Greer Data Set

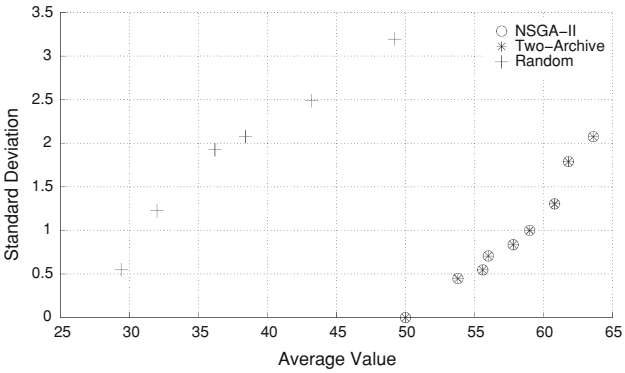
Fig. 4 Comparative results of fairness on absolute *number* of fulfilled value and cost



(a) Result for Random Data Set



(b) Result for Motorola Data Set



(c) Result for Greer Data Set

Fig. 5 Comparative results of fairness on absolute *value* of fulfilled requirements

all runs. Each point represents a optimal subset of requirements solution for the next release.

The two objectives in the experiments are the same as described in Sect. 2.2: (a) minimize the standard deviation of the absolute number or value of fulfilled requirements for each customer and (b) maximize the overall average number or value of fulfilled requirements for all customers.

The results of the first experiment are shown in Fig. 4. In Fig. 4a, the symbol '$\circ$' denotes the solutions produced by the NSGA-II algorithm, while the '$*$' symbol denotes the results generated by the Two-Archive algorithm. We observe that there is no gap between the two sets of solutions. The Pareto fronts of each solution share many common points and almost overlap completely. Therefore, from the figure we can see the Pareto front consists of the markers '$\circledast$'. This means that these two sets of solutions have a large intersection and the search techniques perform better than the Random Search. The results generated by the Random Search could not reach the Pareto front as can be seen by direct observation of the figures.

In terms of performance of two search algorithms, it is difficult to distinguish between them visually, simply by

Table 3 Percentage of solutions on the reference Pareto front (fairness on absolute number of fulfilled requirements)

	NSGA-II (%)	Two-Archive (%)	Random (%)
Random data set	100	100	0
Motorola data set	63.17	32.69	4.36
Greer data set	100	100	0

Table 4 Percentage of solutions on the Reference Pareto front (fairness on absolute value of fulfilled requirements)

	NSGA-II (%)	Two-Archive (%)	Random (%)
Random data set	100	98.08	0
Motorola data set	55.92	44.34	0.40
Greer data set	100	100	0

looking at the figures displayed. Therefore, a Reference Pareto front was constructed and used in this paper to compare the Pareto fronts produced by different algorithms. Consisting of the best solutions of each technique, the Reference Pareto front denotes the best available approximation to the *real* Pareto front. The Pareto fronts generated by the different search techniques may partly contribute to the Reference Pareto front. Therefore, one of the measurements to compare Pareto fronts is to count the number of solutions on the Reference Pareto front, namely the solutions that are not dominated by the Reference Pareto front. The results of this analysis are shown in Tables 3 and 4.

In Tables 3 and 4, the results provide a more quantitative analysis than the figures. For the all data sets, NSGA-II performs best, taking the lion's share of the Reference Pareto front. In the Random data set and Greer data set, NSGA-II and Two-Archive algorithms share almost the same amount of the Pareto front. Indeed, both succeed in covering almost the entire front, showing very similar performance.

However, the Motorola data set, reveals a slightly different story. There is no algorithm that covers the entire Reference Pareto front. The solutions from NSGA-II provide the largest share of the Reference front. Moreover, the results from the Two-Archive algorithm also contribute many solutions to the reference Pareto front. In addition, there is an interesting observation in the results from the application of the three algorithms to the Motorola data set: there are few duplicate solutions obtained by the three algorithms. In this case, there is almost no elements in common in the three sets of solutions and the methods preserve wide diversity of Pareto solutions. For this reason, in Tables 3 and 4, adding up the numerical values together in 'Motorola Data Set' row is almost equal to 100%.

Another observation shows that the Random Search even contributes the reference Pareto front in some cases. The algorithms having occasional good or bad performance may be due to the different characteristics and scale size of data sets. In terms of Random Search, this may occur when the size of data set is comparatively small and therefore denotes a relatively easy optimisation problem.

7 Threats to validity

The results of the current research are subject to limitations which are inherent in any empirical study. This section sets out these limitations, indicating how they may affect the degree to which it is possible to generalise the results. The two real data sets from industry and the literature might not be regarded as a fair representative of all real cases. In order to overcome this, a third set of synthetic data was generated to provide a better coverage of potential instances that remained uncaptured the two real world case studies. More real problems would result in a greater ability to generalise. However, it is known to be difficult to obtain real world data sets; they are typically considered confidential by the companies that own them.

The two algorithms (NSGA-II and Two-Archive) used in this study denote one popular and one comparatively new multi-objective search algorithm. Evolutionary algorithms have been shown effective and efficient in a wide range of problems but, like any other metaheuristic algorithm, there is no guarantee that they are the best available solution for a given problem. As a result, there might be better performing algorithms for fairness analysis. Although identification of the best algorithms for the problem could be the subject of a separate paper, we feel the chosen algorithms are appropriate to address the main research questions in this paper. Our results have demonstrated that the evolutionary optimisation algorithms are superior to Random Search. While this is a relatively low threshold to aim for, it does provide a validity check on the approach. Also, in the absence of any alternative technique, Random Search is a natural choice for a base line validity check. Future work may be able to define further improvements.

A number of assumptions have been made regarding the operators as well as parameter values of the search algorithms which can influence the results. However, all assumptions are the same for both algorithms to minimize environmental factors during the experimentations. Performance of the algorithms could have been improved by individual fine tuning empirically or through systematic experimentation. This is beyond the scope of the current research but could be suggested as an avenue for future research.

In addition, we do not consider dependencies that may exist between the requirements. For instance, some requirements may be coupled together. That is, if one requirement is selected in one release, then the others should be included as well in order to satisfy dependencies. This problem could be addressed, for example, by introducing new criteria as the constrains in our models. A chosen subset of requirements is a possible solution if it satisfies both the objectives and the dependence constrains.

8 Related work

In the area of requirements engineering, several related studies have been proposed for requirements analysis and optimisation. Karlsson [33, 34, 47] 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. [5] 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 et al. [53] study a variation of the problem using integer linear programming to find exact solutions within budgetary constraints.

Zhang et al. [56] 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. Also, the scalability of the approach (in terms of e.g. number of requirements and number of customer) was discussed in the paper. The authors have compared the performance of the techniques and the impact of requirement and customer set size on the different scale data sets.

Greer and Ruhe [27] 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.

More recently, there has more work on multi-objective formulations of the NRP [49, 55]. Ruhe and Omolade [49] showed how search based optimisation can balance the tension between user-level and system-level requirements and track dependencies from user requirements into their impact on system components. Zhang et al. [55] summarised existing achievements and described future challenges for Search Based Requirements Optimisation in recent years.

Problems associated with multiple customers with completing and conflicting view points has been known for some time [41]. In et al. [31, 32] 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 [22, 23], which separates the different opinions among the stakeholders and can detect conflicts automatically. In the stakeholder requirements analysis problem, Robinson et al. [1, 45] worked on a requirements negotiation model which provided automated support to generate requirements resolutions.

Fleming et al. [25] 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 [17] compare a number of ranking methods to address the shortcoming of existing evolutionary algorithms for many-objective optimisation. Deb and Kumar [19] 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. The paper is an extended version of a paper from the Requirements Engineering conference [24]. The primary technical extension in this paper lies in the results of the comparative algorithmic study reported in Sect. 6, though other sections have also been extended and/or rewritten to provide more detail.

The approach advocated in this paper is based on Search Based Software Engineering (SBSE) [28], which has been widely used to provide optimal and near optimal solutions to problems from a range of software engineering problems from software engineering project management [2–4, 12, 21, 36] through design [29, 39, 40, 50] to testing [7–11, 38, 54] and refactoring [42].

9 Conclusions

This 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 techniques 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.

The paper presents results of a comparative study of Random Search and two more sophisticated, evolutionary multi-objective search techniques. The results validate the overall approach, showing that the more sophisticated techniques comfortably outperform Random Search. The results also reveal that results across the more sophisticated techniques are reassuringly consistent in terms of their performance and somewhat complementary in terms of their diversity.

Future work will verify these findings by applying other search techniques and also classical optimisation techniques, such as, integer programming, nonlinear programming etc. This will provide valuable feedback to researchers and practitioners in the software engineering community.

References

1. Automated support for requirements negotiation, 17 March 1994
2. Aguilar-Ruiz J, Ramos I, Riquelme JC, Toro M (2001) An evolutionary approach to estimating software development projects. Inf Softw Technol 43(14):875–882
3. Antoniol G, Di Penta M, Harman M (2004) A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In: 10th international software metrics symposium (METRICS 2004), Los Alamitos, California, USA. IEEE Computer Society Press, Los Alamitos, pp 172–183
4. Antoniol G, Penta MD, Harman M (2005) Search-based techniques applied to optimization of project planning for a massive maintenance project. In: 21st IEEE international conference on software maintenance, Los Alamitos, California, USA. IEEE Computer Society Press, Los Alamitos, pp 240–249
5. Bagnall A, Rayward-Smith V, Whittley I (2001) The next release problem. IEEE Proc Softw 43(14):883–890
6. Baker P, Harman M, Steinhöfel K, Skaliotis A (2006) Search based approaches to component selection and prioritization for the next release problem. In: 22nd international conference on software maintenance (ICSM 06), Philadelphia, Pennsylvania, USA, pp 176–185
7. Baresel A, Binkley DW, Harman M, Korel B (2004) Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach. In: International symposium on software testing and analysis (ISSTA 2004), Omni Parker House Hotel, Boston, Massachusetts. Software Engineering Notes, vol 29, No. 4, pp 108–118
8. Baresel A, Sthamer H, Schmidt M (2002) Fitness function design to improve evolutionary structural testing. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, San Francisco, CA 94104, USA. Morgan Kaufmann Publishers, San Francisco, pp 1329–1336
9. Bottaci L (2002) Instrumenting programs with flag variables for test data search by genetic algorithms. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, New York. Morgan Kaufmann Publishers, New York, pp 1337–1342
10. Briand LC, Feng J, Labiche Y (2002) Using genetic algorithms and coupling measures to devise optimal integration test orders. In: SEKE, pp 43–50
11. Briand LC, Labiche Y, Shousha M (2005) Stress testing real-time systems with genetic algorithms. In: H-G Beyer, U-M O'Reilly (eds) Genetic and evolutionary computation conference, GECCO 2005, Proceedings, Washington DC, USA, June 25–29, 2005. ACM Press, New York, pp 1021–1028
12. Burgess CJ, Lefley M (2001) Can genetic programming improve software effort estimation? A comparative evaluation. Inf Softw Technol 43(14):863–873
13. Carlshamre P (2002) Release planning in market-driven software product development: provoking an understanding. Requir Eng 7(3):139–151
14. Coello Coello CA (2006) Evolutionary multiobjective optimization: a historical view of the field. IEEE Comput Intell Mag 1(1):28–36
15. Coello Coello CA, Van Veldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, New York
16. Collette Y, Siarry P (2004) Multiobjective optimization: principles and case studies. Springer, Berlin
17. Corne D, Knowles J (2007) Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: GECCO'07: proceedings of genetic and evolutionary computation conference, pp 773–780
18. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
19. Deb K, Kumar A (2007) Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In: GECCO'07: proceedings of genetic and evolutionary computation conference, pp 781–788
20. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Trans Evol Comput 6(2):182–197
21. Dolado JJ (2000) A validation of the component-based method for software size estimation. IEEE Trans Softw Eng 26(10):1006–1021
22. Easterbrook S, Chechik M (2001) A framework for multi-valued reasoning over inconsistent viewpoints. In: Proceedings of 23rd international conference on software engineering, May 2001. IEEE Computer Society Press, Oakland, pp 411–420
23. Easterbrook S, Finkelstein A, Kramer J, Nuseibeh B (1994) Co-ordinating distributed ViewPoints: the anatomy of a consistency check. Technical Report 333, School of Cognitive and Computing Sciences, University of Sussex, UK
24. Finkelstein A, Harman M, Mansouri SA, Ren J, Zhang Y (2008) "Fairness analysis" in requirements assignments. In: Proceedings of the 16th IEEE international requirements engineering conference (RE '08), Barcelona, Catalunya, Spain, 8–12 September
25. Fleming PJ, Purshouse RC, Lygoe RJ (2005) Many-objective optimization: an engineering design perspective. Lect Notes Comput Sci 3410:14–32
26. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
27. Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253
28. Harman M (2007) The Current State and Future of Search Based Software Engineering. In: 29th international conference on software engineering (ICSE 2007), Future of Software Engineering (FoSE), Minneapolis, USA, May 20–26
29. Harman M, Hierons R, Proctor M (2002) A new representation and crossover operator for search-based optimization of software modularization. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, San Francisco, CA 94104,

USA, 9–13 July 2002. Morgan Kaufmann Publishers, San Francisco, pp 1351–1358
30. Horn J, Nafpliotis N (1993) Multiobjective optimization using the niched pareto genetic algorithm. Technical Report IlliGAL 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, IL
31. In H, Boehm B, Rodgers T, Deutsch M (2001) Applying winwin to quality requirements: a case study. In: Proceedings of the 23rd international conference on software engineering, May 2001. IEEE Computer Society Press, Oakland, pp 555–564
32. In H, Olson D, Rodgers T (2001) A requirements negotiation model based on multi-criteria analysis. In: RE. IEEE Computer Society Press, Oakland, pp 312–313
33. Karlsson J (1996) Software requirements prioritizing. In: ICRE, pp 110–116
34. Karlsson J, Ryan K (1996) Supporting the selection of software requirements. In: IWSSD '96: proceedings of the 8th international workshop on software specification and design, Washington, DC, USA, 1996. IEEE Computer Society, Washington, p 146
35. Karlsson L, Regnell B, Thelin T (2006) Case studies in process improvement through retrospective analysis of release planning decisions. Int J Softw Eng Knowl Eng 16(6)
36. Kirsopp C, Shepperd M, Hart J (2002) Search heuristics, case-based reasoning and software project effort prediction. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, San Francisco, CA 94104, USA, 9–13 July 2002. Morgan Kaufmann Publishers, San Francisco, pp 1367–1374
37. Knowles JD, Corne DW, Oates MJ (2000) The pareto-envelope based selection algorithm for multiobjective optimization. In: Proceedings of the sixth international conference on parallel problem solving from nature (PPSN VI), Berlin, September 2000. Springer, Berlin, pp 839–848
38. Li Z, Harman M, Hierons R (2008) Meta-heuristic search algorithms for regression test case prioritization. IEEE Trans Softw Eng (to appear)
39. Mitchell BS, Mancoridis S (2002) Using heuristic search techniques to extract design abstractions from source code. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, San Francisco, CA 94104, USA, 9–13 July 2002. Morgan Kaufmann Publishers, San Francisco, pp 1375–1382
40. Mitchell BS, Mancoridis S (2006) On the automatic modularization of software systems using the bunch tool. IEEE Trans Softw Eng 32(3):193–208
41. Nuseibeh B, Kramer J, Finkelstein A (1994) A framework for expressing the relationships between multiple views in requirements specifications. IEEE Trans Softw Eng 20(10):760–773
42. O'Keeffe M, O'Cinneide M (2006) Search-based software maintenance. In: Conference on software maintenance and reengineering (CSMR'06), March 2006, pp 249–260
43. Osyczka A (1985) In: Multicriteria optimization for engineering design. Design Optimization, pp 193–227
44. Praditwong K, Yao X (2006) A new multi-objective evolutionary optimisation algorithm: the two-archive algorithm. In: Proceedings of the 2006 international conference on computational intelligence and security (CIS'2006). IEEE Press, NJ, pp 286–291
45. Robinson WN, Volkov V (1999) Requirement conflict restructuring, August 25, 1999
46. Ruhe G, Saliu MO (2005) The art and science of software release planning. IEEE Softw 22(6):47–53
47. Ryan K, Karlsson J (1997) Prioritizing software requirements in an industrial setting. In: ICSE, pp 564–565
48. Saliu MO, Ruhe G (2005) Software release planning for evolving systems. ISSE 1(2):189–204
49. Saliu MO, Ruhe G (2007) Bi-objective release planning for evolving software systems. In: Proceedings of ESEC/SIGSOFT FSE 2007
50. Seng O, Bauer M, Biehl M, Pache G (2005) Search-based improvement of subsystem decompositions. In: H-G Beyer, O'Reilly U-M (eds) Genetic and evolutionary computation conference, GECCO 2005, Proceedings, Washington DC, USA, June 25–29, 2005. ACM Press, New York, pp 1045–1051
51. Srinivas N, Deb K (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248, Fall
52. Szidarovsky F, Gershon ME, Dukstein L (1986) Techniques for multiobjective decision making in systems management. Elsevier, New York
53. van den Akker M, Brinkkemper S, Diepen G, Versendaal J (2008) Software product release planning through optimization and what-if analysis. Inf Softw Technol 50(1–2):101–111
54. Wegener J, Baresel A, Sthamer H (2001) Evolutionary test environment for automatic structural testing. Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms 43(14):841–854
55. Zhang Y, Finkelstein A, Harman M. Search based requirements optimisation: existing work and challenges. In: Proceedings of the 14th international working conference, requirements engineering: foundation for software quality (REFSQ '08), vol 5025 of LNCS, Montpellier, France, 16–17 June 2008. Springer, Berlin, pp 88–94
56. Zhang Y, Harman M, Mansouri SA (2007) The multi-objective next release problem. In: GECCO'07: proceedings of the genetic and evolutionary computation conference, ACM Press, pp 1129–1136
57. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001