The Multi-Objective Next Release Problem

Yuanyuan Zhang
King's College London
Strand, London
WC2R 2LS, UK
yuanyuan.zhang@kcl.ac.uk

Mark Harman
King's College London
Strand, London
WC2R 2LS, UK
mark.harman@kcl.ac.uk

S. Afshin Mansouri
King's College London
Strand, London
WC2R 2LS, UK
afshin.mansouri@kcl.ac.uk

ABSTRACT

This paper is concerned with the Multi-Objective Next Release Problem (MONRP), a problem in search-based requirements engineering. Previous work has considered only single objective formulations. In the multi-objective formulation, there are at least two (possibly conflicting) objectives that the software engineer wishes to optimize. It is argued that the multi-objective formulation is more realistic, since requirements engineering is characterised by the presence of many complex and conflicting demands, for which the software engineer must find a suitable balance. The paper presents the results of an empirical study into the suitability of weighted and Pareto optimal genetic algorithms, together with the NSGA-II algorithm, presenting evidence to support the claim that NSGA-II is well suited to the MONRP. The paper also provides benchmark data to indicate the size above which the MONRP becomes non–trivial.

Categories and Subject Descriptors

D.2.1 [**SOFTWARE ENGINEERING**]: Requirements/ Specifications—*Methodologies*

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Pareto optimality, next release problem, multi-objective genetic algorithms

1. INTRODUCTION

In this paper we address the Multi-Objective Next Release Problem (MONRP), in which a set of customers with varying requirements are targeted for the next release of an existing software system. Satisfying each requirement entails spending a certain amount of resources which can be translated into *cost* terms. In addition, satisfying each requirement provides some *value* to the software development company. The problem is to select the set of requirements that maximize total value and minimize required cost. These objectives are conflicting, so it would be difficult to assign weights to each. This makes it inappropriate to combine the two objectives into a single fitness function. Rather, a multi-objective formulation is better suited to this problem.

The single objective problem to maximize the value, subject to a limited amount of resources (a fixed budget constraint), is an instance a *Knapsack* Problem which is $\mathcal{NP}$-hard [22]. As a result, the multi-objective problem stated above is also $\mathcal{NP}$-hard, and therefore cannot be solved using exact optimization techniques for large scale problem instances.

As explained in Section 2, single objective formulations of the Next Release Problem (NRP) have been considered in the literature. Also, approaches have been considered, where the multiple objectives are combined into a single objective using weight. However, the multi-objective version of the problem, in which there are two or more competing objectives (which may conflict) has not been considered. The multi-objective formulation is important because, in practice, a software engineer is more likely to have many conflicting objectives to address when determining the set of requirements to include in the next release of the software. As such, the MONRP is more likely to be appropriate than the single objective NRP.

In this paper we apply metaheuristic search techniques to find approximations of Pareto-optimal set (or front) for the MONRP. This allows the decision maker to select the preferred solution from the Pareto-optimal set, according to their priorities. The Pareto front can also provide valuable insights into the outcome of the selected set of requirements to the software development company, because it captures the trade-offs between the two competing objectives. The results can also be used in 'what if?' analyses, for instance:

> "what would we gain if we could securely assign 10% more resources to the project?""

Or

> "what would we lose by reducing the project's budget by 20%?"

In such situations, a Pareto front is more useful in exploring the outcome of these changes to the scenario, because it captures the entire range of trade-off decisions, rather than fixing on a single point. The single objective formulation of the problem lacks the ability to provide such decision aids.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

The paper introduces a formal definition of the MONRP and develops four different multi-objective search-based solutions: Random Search, Single-Objective (Weighted) Genetic Algorithm, a Pareto GA and Non-dominated Sorting Genetic Algorithm II (NSGA-II) approach proposed by Deb et al. [8]. Random Search is included partly to provide a 'sanity check' that metaheuristic search techniques can outperform it (as would be expected) and partly to allow us to empirically identify the point at which the number of customers and requirements becomes sufficiently large to ensure that the problem is non–trivial. Results from applying the four algorithms are presented that compare their performance at finding approximations of the Pareto-optimal front.

The primary contributions of the paper are as follows:

1. The paper formulates the Next Release Problem as Multi-Objective Optimization Problem, treating the 'cost' constraint as an objective and combining it with the 'value' objective. This is the first paper to generalise the NRP to the Multi-Objective NRP (MONRP).

2. The paper presents the results of an empirical study into the relative performance of four different multi-objective search techniques for solving the problem. The results show:

   (a) That the weighted Single-Objective GA, Pareto GA and NSGA-II can all outperform Random search (passing the 'sanity check');

   (b) That NSGA-II outperforms the Pareto GA, both in terms of diversity of results and in terms of quality of the results (which dominate all results from the Pareto GA);

   (c) That the weighted Single-Objective GA can be helpful in finding extreme points on the Pareto front, by a suitable choice of weights.

3. The paper presents the results of a second empirical study, aimed at determining the size of problem for which the MONRP becomes non–trivial. This reveals that the number of requirements must be larger than approximately 20, while the number of customers need only be larger than 2 in order to produce a non–trivial instance of the MONRP.

The rest of the paper is organised as follows:

Section 2 describes the context of related work in which the current paper is located. Section 3 gives the definitions of the Multi-Objective Optimization Problem (MOOP) and the Pareto-optimal front. In Section 4 the research problem is defined formally, while Section 5 introduces the search algorithms studied and how they are tailored to the MONRP. Sections 6 and 7 present the results of the experiments and discuss the findings. Section 8 concludes.

2. RELATED WORK

The most closely related previous work to the content of the present paper is the work on the Next Release Problem (NRP) studied by several authors [2, 10, 14]. The term: 'Next Release Problem' was coined by Bagnall et al. [2]. In the NRP, as formulated by Bagnall et al., the goal is to find the ideal set of requirements that balance customer requests within resource constraints, so the problem is a constrained single objective optimization problem. They applied a variety of techniques (including search based and non search based algorithms) to a set of synthetic data. Greer and Ruhe also studied the NRP, proposing a Genetic Algorithm–based approach for planning software releases [10].

The NRP is an example of a Feature Subset Selection (FSS) search problem. Other FSS problems in previous work on SBSE include the problem of determining good quality predictors in software project cost estimation, studied by Kirsopp et al. [15] and chosing components to include in different releases of a system, studied by Harman et al. [11].

Previous work, both on the NRP and other FSS problems in SBSE has been solely concerned with single objective formulations of the problems concerned. The present paper is the first paper to generalise the NRP to the Multi-Objective NRP (MONRP). Indeed, much of the other existing work on SBSE has also tended to consider software engineering problems as single objective optimization problems. However, a recent trend appears to be developing, in which multiple objectives are considered. This would seem to be a natural and realistic extension of the initial work on SBSE, since so many software engineering problems are inherently multi-objective.

Other existing SBSE work that does consider multi-objective formulations of software engineering problems, tends to use the weighted approach to combine fitness functions for each objective into a single objective function using weighting coefficients to denote the relative importance of each individual fitness function. For example, in the seminal work of the Drexel group [9, 17, 18, 19] on search based clustering, the two objectives of cohesion and coupling are combined into a single objective by the Module Quality Metric (MQ), which has been widely studied, both by the Drexel group and also by other authors [5, 12, 16]. For search based refactoring, both Seng et al. [23] and O'Keeffe and O'Cinnéide [20] use a weighted multi-objective search, in which several metrics that assess the quality of refactorings are combined into a single objective function. In the domain of search based project planning [1], Chicano and Alba [3] combined several project management metrics into a single objective function, using weighting to guide resource allocation.

The present paper is one of the first papers on SBSE to consider the set of objectives independently in order to explore the search space towards the Pareto-optimal front (rather than simply weighting each objective to form a unified objective function which, at best, is capable of directing to a single Pareto-optimal solution).

The Pareto approach is more suitable when it is difficult to combine fitness functions into a single overall objective function. Such a combination is difficult in many situations, for example, because the fitness functions measure fundamentally different properties (so combining them would be to seek to 'compare apples and oranges'), because weights cannot be adequately determined or because the use of weights biases the search to a certain part of the solution space. As the results presented in the present paper indicate, for the MONRP, the weighting approach suffers from this problem; the weights force the search towards certain small areas of the Pareto front so several runs with different weights are required to find diverse approximations of Pareto fronts.

3. PARETO-OPTIMAL FRONT

The Multi-Objective Optimization 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 form a mathematical description of performance criteria that are usually in conflict with each other [21].

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 Optimization 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.

4. PROBLEM STATEMENT

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

4.1 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.

The set of possible software requirements is denoted by:

$$\mathcal{R} = \{r_1, \ldots, r_n\}$$

It's assumed that all the requirements are independent. 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. The resultant cost vector for the set of requirements $r_i(1 \leq i \leq n)$ is denoted by:

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

Each customer has a degree of importance for the company that can be reflected by a weight factor. The set of relative weights associated with each customer $c_j(1 \leq j \leq m)$ is denoted by:

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

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

It is assumed that all requirements are not 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 company. 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 has the requirement i and 0 otherwise.

Based on above, the overall *score* or importance of a given requirement $r_i(1 \leq i \leq n)$ can be calculated by:

$$score_i = \sum_{j=1}^{m} w_j \cdot value(r_i, c_j)$$

The '*score*' of a given requirement is represented as its overall '*value*' for the company.

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

4.2 MONRP Formulation

In the formulation of the MONRP, two objectives are taken into consideration in order to maximize customer satisfaction (or total *value* for the company) and minimize required cost. We treat cost in the current research as an objective instead of a constraint for the first time in our MONRP model. The reason is to explore the whole set of points on the Pareto-optimal front; this is a valuable source of information for the decision maker to understand the trade-offs inherent in meeting the conflicting objectives.

The following objective function is considered for maximizing total value:

$$\text{Maximize } \sum_{i=1}^{n} score_i \cdot x_i$$

The problem is to select a subset of the customers' requirements which results in the maximum value for the company.

The second objective function is defined as follows to minimize total cost required for the satisfaction of customer requirements:

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

In order to convert the second objective to a maximization problem in the MONRP, the total cost is multiplied by -1. Therefore, the MONRP model consisting of fitness functions can be represented as follows:

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

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

5. THE SOLUTION APPROACHES

This section describes the search algorithms used in this paper. As stated earlier, in the solution of 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 optimization, 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 Szidarovsky et al. [25] and Collette and Siarry [6].

Among meta-heuristics, Evolutionary Algorithms (EAs) seem particularly desirable to solve MOOPs. EAs are used to solve problems of this nature mainly because of the population based nature of EAs which 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, whereas these two issues are a real concern for mathematical programming techniques.

EAs usually contain several parameters that need to be 'tuned' for each particular application, which is, in many cases, highly time consuming. In addition, since the EAs are stochastic optimizers, 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 on application of EAs in multi-objective optimization see Coello et al. [4] and Deb [7].

To solve the MONRP, Multi-Objective EAs need to fulfill two major tasks:

1. Guiding the search towards the Pareto-optimal set to accomplish fitness assignment.

2. Maintaining a diverse population to achieve a well distributed non-dominated front.

We examine three search techniques namely: NSGA-II, a Pareto GA and a Single-Objective GA for the solution of the MONRP. These techniques are also compared with the Random Search to verify viability of applying computationally expensive search techniques for the MONRP.

5.1 NSGA-II

Non-dominated Sorting Genetic Algorithm-II (NSGA-II), introduced by Deb et al.[8] is an extension to an earlier Multi-Objective EA called NSGA developed by Srinivas and Deb [24]. 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. 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 better fitness compared to those with lower crowding distance, thereby avoiding the use of the fitness sharing factor [13].

Assume that every individual i in the population has two attributes:

1. nondomination rank (i_{rank});

2. crowding distance ($i_{distance}$).

We now define a partial order $\prec_n$ as

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

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

That is, between two solution with differing nondomination ranks, we prefer the solution with the lower(better)rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a lesser crowded region[8].

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

Algorithm 1: NSGA-II (main loop)

```
while not stopping rule do
    Let R_t = P_t ∪ Q_t;
    Let F = fast-non-dominated-sort (R_t);
    Let P_{t+1} = ∅ and i = 1;
    while |P_{t+1}| + |F_i| ⩽ N do
        Apply crowding-distance-assignment(F_i);
        Let P_{t+1} = P_{t+1} ∪ F_i;
        Let i = i + 1;
    end
    Sort(F_i, ≺_n);
    Let P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)];
    Let Q_{t+1} = make-new-pop(P_{t+1});

    Let t = t + 1;
end
```

5.2 Pareto GA

The Pareto GA algorithm used in this research is a variation of the simple genetic algorithm. A simple GA maintains the populations of candidate solutions that are evaluated according to a single fitness value. In the MONRP there are at least two fitness values for each solution which are used in tournament selection. The algorithm uses Pareto dominance relations between solutions to select candidates for the

mating pool. The new population is created by recombining the selected solutions through crossover and mutation operators. The main procedure of the Pareto GA is described in Algorithm 2.

Algorithm 2: Pareto GA

```
t = 0;
Generate population P_0;
while not stopping rule do
    Evaluate objective functions for ∀i ∈ P_t;
    Let t = t + 1;
    Let i = 0;
    while i < N do
        Consider two solutions x and y ∈ P_t at random;
        Let selected solution = ∅;
        if x ≻ y then
            Let selected solution = x;
            Let i = i + 1;
        end
        else if y ≻ x then
            Let selected solution = y;
            Let i = i + 1;
        end
        Add selected solution to mating pool;
    end
    Apply genetic operators (crossover and mutation) to P_t;
end
```

5.3 Single-Objective GA

To apply a Single-Objective GA to MONRP, we use the weighting method that combines the objective functions into a single objective using a weight factor $\omega(0 \leq \omega \leq 1)$. The fitness value of a given solution $\overrightarrow{x}$ in the Single-Objective GA is calculated as follows:

$$F(\overrightarrow{x}) = (1-\omega) \cdot f_1(\overrightarrow{x}) + \omega \cdot f_2(\overrightarrow{x})$$

By changing the weight factor $\omega \in [0, 1]$, one can explore various regions of the Pareto-optimal front. This approach is employed to compare the performance of the Single-Objective GA with other search techniques.

5.4 Random Search

We also applied the Random Search technique 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. Thus the Random Search technique was given the same number of fitness evaluations as the other algorithms to provide a useful lower bound benchmark for measuring the other algorithms' performance.

6. EXPERIMENTAL SET UP

In this section, we describe the test problems used to compare the performance of NSGA-II with Pareto GA, Random Search and Single-Objective GA.

The test problems were created by assigning random choices for value and cost. The range of costs were from 1 through to 9 inclusive (zero cost is not permitted). The range of values were from 0 to 5 inclusive (zero value is permitted, indicating that the 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. Each algorithm was executed 5 times for each data set.

The four algorithms were applied to two test problem sets, for two separate empirical study cases: Empirical Study 1 (ES1) and Empirical Study 2 (ES2). In the ES1 we report results concerning the performance of the four algorithms for what might be considered 'typical' cases, with the number of customers ranging from 15 to 100 and the number of requirements ranging from 40 to 140. In ES2, we are concerned with bounding the problem below, to determine the size of problem for which search is not appropriate; the point at which the problem becomes too small. In order to do this, we seek to find the point at which the metaheuristic techniques can (only just) outperform a Random Search, since we deem this to indicate that the problem is sufficiently large for it to be worth considering the application of metaheuristic search techniques.

All approaches were run for a maximum of 10,000 function evaluations. The initial population was set to 200. We used a simple binary GA encoding, with one bit to code for each decision variable (the inclusion or exclusion of a requirement). The length of a chromosome is thus equivalent to the number of requirements. Each experimental execution of each algorithm was terminated when the generation number reached 51 (i.e after 10,000 evaluations). All genetic approaches used the tournament selection (the tournament size is 5), single-point crossover and bitwise mutation for binary-coded GAs. The crossover probability was set to $P_c = 0.8$ and mutation probability to $P_m = 1/n$ (where n is the string length for binary-coded GAs) were used.

7. RESULTS AND ANALYSIS

7.1 Empirical Study 1—Scale Problem

In ES1 we investigate the relative performance of the four approaches to the MONRP for cases that we consider typical. We consider three 'scales' of problem that we hope are characteristic of some of the problems to which the approach may be applied. The number of customers and requirements for each scale problem is listed in Table 1:

Table 1: Scale test sets

Scale	Customer	Requirement
S1	15	40
S2	50	80
S3	100	140

The results of S1, S2 and S3 are shown in Figure 1(a-c). The figure shows typical results from the 5 runs of each algorithm. Space does not permit us to show all results, but the results from other runs were very similar to those shown in the figure. In each picture, the results show the non-dominated front obtained after 50 generations with NSGA-II, Pareto GA, Single-Objective GA, and all 10,000 solutions obtained by Random Search.

In the Single-Objective GA, there are nine different weight coefficients ω for each objective ranging from 0.1 to 0.9 in this paper. The step size of weight is 0.1. The algorithm was executed 9 times for different choices of weight coefficients to obtain different solutions within a single experiment.

The results lend evidence to support the following claims:

1. The NSGA-II algorithm performs the best in all three scale problems. Figures 1 (a), (b) and (c) show a smooth, non-interrupted Pareto-optimal front generated by NSGA-II. Clearly, it is able to find a better diversity of solution distribution and it also converges on results that are better than those for any of the other algorithms, because the front line dominates those produced by the other approaches. This provides empirical evidence that the NSGA-II algorithm is effective in solving the MONRP and that it may outperform the Pareto GA and the Single-Objective GA (as well, of course, as random).

2. The solutions obtained using the weight-based Single-Objective GA where ω was close to 0.0 or 1.0 drive the search towards the extreme ends of the Pareto front. These extreme parts of the Pareto front do not appear to be explored by NSGA-II, even though it does produce a good spread of results. In such extreme solutions, one objective is maximized to the almost total exclusion of the other. Although the Single-Objective GA may have difficulties in finding a good spread of solutions near the Pareto-optimal front, these extreme non-dominated solutions which the Single-Objective GA managed to find provide a necessary supplement to the NSGA-II solutions. They may be useful in real world scenarios, because they allow the software engineer to explore the extremes of the Pareto front, where one objective dominates.

3. The trend we observed from the three experiments is that the larger the scale of the test problem, the wider the gap in performance, both between metaheuristic techniques and Random Search and between the leader (NSGA-II) and the runner up (Pareto GA). This indicates that as the problem scales, the performance improvement offered by the use of NSGA-II also scales, making it an increasingly preferred choice of solution algorithm.

7.2 Empirical Study 2—Boundary Problem

In ES2, we want to explore the boundaries of MONRP. For the lower boundary, there are two extreme situations, namely many customers with very few requirements and vice versa. In this section, we are interested in obtaining the 'critical mass' of both customers and requirements. Once the critical mass exceeds a predetermined critical point, it becomes worthwhile applying metaheuristic techniques to the MONRP; below this point, the problem is too trivial to merit a metaheuristic solution.

In the first situation, there are many customers and few requirements. We set the number of customers to 100, with the aim of finding the smallest set of requirements to meet the critical point.

Figure 1(d) illustrates this critical point. From the figure we can see that there is no gap between the solutions generated by the Random Search and the Pareto-optimal front produced by NSGA-II. The number of requirements is set to 20 in Figure 1(d). The NSGA-II and the Random Search solutions share several common points. However, when we increase the number of requirements to 25, as shown in Figure 1(e), we can see that there is a clear gap between the solutions produced by the Random Search and the NSGA-II. If we increase the number of requirements further, the gap continues to widen.

In the second situation, there are few customers and many requirements. We assume that the smallest number of customers is 2. However, even with only 2 customers, with sufficient requirements, there is a noticeable difference in the performance of the metaheuristic search algorithms and Random Search. This is illustrated in Figure 1(f), where the number of requirements is 200. In this figure, NSGA-II clearly outperforms other algorithms.

Thus it can be seen that the primary determinant of the 'tipping point' is a function of the number of requirements, which should exceed about 20 requirements. By contrast there is no number of customers that is 'too small' for the problem to be worthwhile.

8. CONCLUSIONS AND FUTURE WORK

In this paper we address the Multi-Objective Next Release Problem (MONRP) for the first time, in order to optimize both value and cost simultaneously. These objectives are naturally conflicting so attaining a single optimal solution may not be possible in many cases. Instead, the set of Pareto-optimal solutions are to be sought that enables decision makers to select the best solutions in different circumstances based on their priorities. Four search techniques namely: Random Search, Single-Objective GA, Pareto GA and NSGA-II were examined to find approximations of the Pareto front in different problem instances.

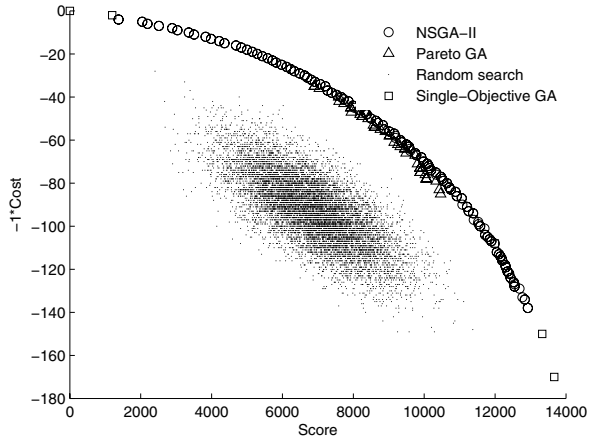
It was observed that the NSGA-II outperforms other techniques in finding a large and important part of the Pareto front in large problems. However, for these problems, a Single-Objective GA (applied to the unified objective function) performs better in finding extreme regions along the front when is run iteratively, using varying weights for the two objectives.

In small problem instances, no major difference was observed between the solution techniques in terms of quality of the Pareto front found. In this way, the paper provides an empirically determined lower bound benchmark on problem size, indicating the point below which the MONRP becomes trivial.

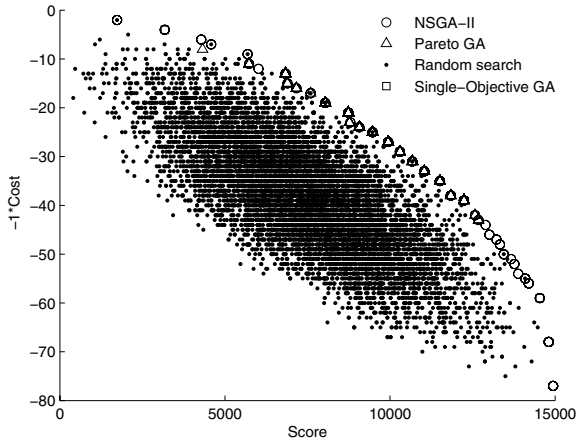
Future work will verify these findings by applying search techniques to real world problems. This will provide valuable feedback to researchers and practitioners in search techniques as well as software engineering communities. Other reformulations of the MONRP considering different sets of objectives and constraints including dependency relationship between requirements will be experimented with. This in turn may give rise to the need for the development of more efficient solution techniques.

9. REFERENCES

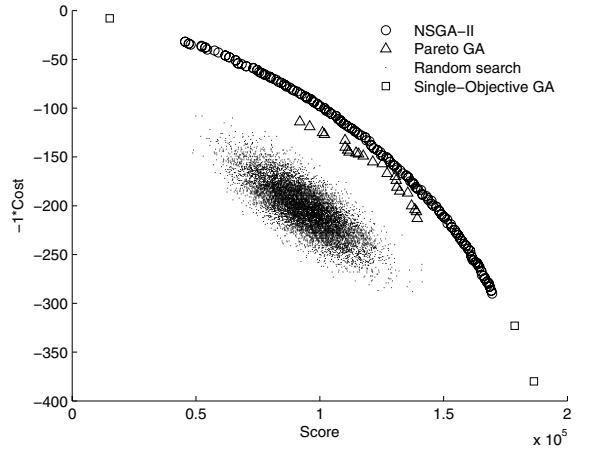
[1] Antoniol, G., Penta, M. D., and Harman, M. Search-based techniques applied to optimization of project planning for a massive maintenance project. In 21^{st} *IEEE International Conference on Software*



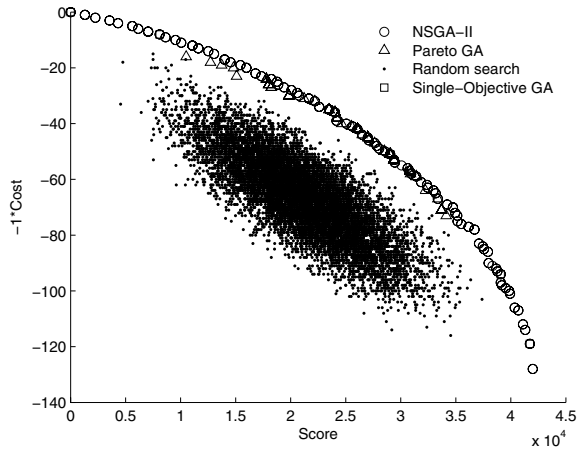
(**a**) 15 customers; 40 requirements



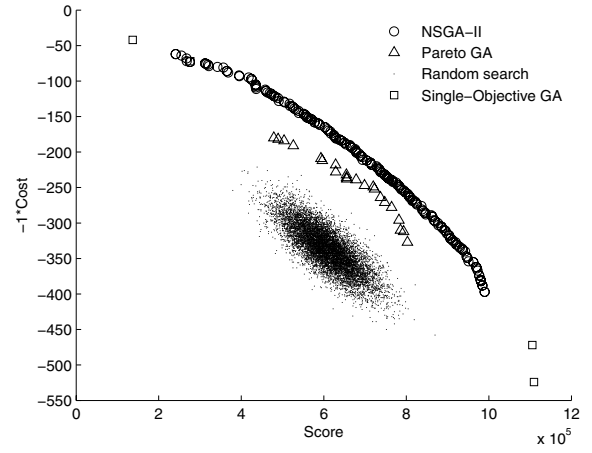
(**d**) 100 customers; 20 requirements



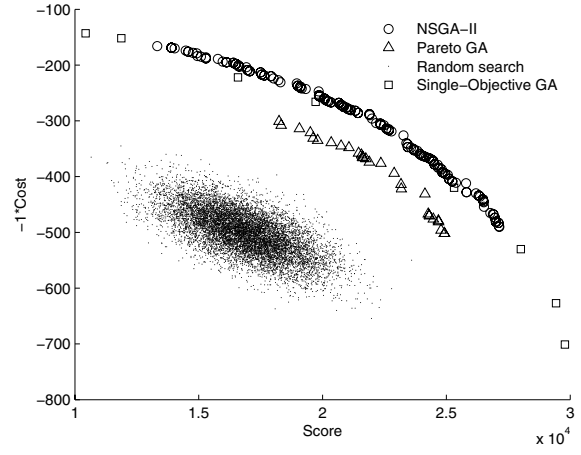
(**b**) 50 customers; 80 requirements



(**e**) 100 customers; 25 requirements



(**c**) 100 customers; 140 requirements



(**f**) 2 customers; 200 requirements

Figure 1: In (a), (b) and (c) metaheuristic search techniques have outperformed Random Search. The NSGA-II performed better or equal to others for a large part of the Pareto front while Single-Objective performed better in the extreme regions. The gap between search techniques and Random Search became larger as the problem size increased. (d) shows the boundary case concerning the number of requirements beyond which, Random Search fails to produce comparable results with metaheuristic search techniques. (e) shows 25% increase in the number of requirements, the gap became significant. The NSGA-II performed the best, and Pareto GA shared part of the front with NSGA-II. (f) shows that the gap was obviously large. The NSGA-II has outperformed Pareto GA but not the Single-Objective GA in the extreme ends of the front.

Maintenance (Los Alamitos, California, USA, 2005), pp. 240–249.

[2] Bagnall, A., Rayward-Smith, V., and Whittley, I. The next release problem. *Information and Software Technology 43*, 14 (Dec. 2001), 883–890.

[3] Chicano, F., and Alba, E. Management of software projects with gas. In 6^{th} *Metaheuristics International Conference (MIC2005)* (Vienna, Austria, Aug. 2005).

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

[5] Cohen, M., Kooi, S. B., and Srisa-an, W. Clustering the heap in multi-threaded applications for improved garbage collection. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (Seattle, Washington, USA, 8-12 July 2006), M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds., vol. 2, ACM Press, pp. 1901–1908.

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

[7] Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

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

[9] Doval, D., Mancoridis, S., and Mitchell, B. S. Automatic clustering of software systems using a genetic algorithm. In *International Conference on Software Tools and Engineering Practice (STEP'99)* (Pittsburgh, PA, 30 August - 2 September 1999).

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

[11] Harman, M., Steinhöfel, K., and Skaliotis, A. Search based approaches to component selection and prioritization for the next release problem. In 22^{nd} *International Conference on Software Maintenance (ICSM 06)* (Philadelphia, Pennsylvania, USA, Sept. 2006). To appear.

[12] Harman, M., Swift, S., and Mahdavi, K. An empirical study of the robustness of two module clustering fitness functions. In *Genetic and Evolutionary Computation Conference (GECCO 2005)* (Washington DC, USA, June 2005), pp. 1029–1036.

[13] Horn, J., and Nafpliotis, N. Multiobjective optimization using the niched pareto genetic algorithm. Tech. Rep. IllIGAL 93005, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1993.

[14] Karlsson, J., Wohlin, C., and Regnell, B. An evaluation of methods for priorizing software requirements. *Information and Software Technology 39* (1998), 939–947.

[15] Kirsopp, C., Shepperd, M., and Hart, J. 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), W. B. Langdon, E. Cant-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, Eds., Morgan Kaufmann Publishers, pp. 1367–1374.

[16] Mahdavi, K., Harman, M., and Hierons, R. M. A multiple hill climbing approach to software module clustering. In *IEEE International Conference on Software Maintenance* (Los Alamitos, California, USA, Sept. 2003), pp. 315–324.

[17] Mancoridis, S., Mitchell, B. S., Chen, Y.-F., and Gansner, E. R. Bunch: A clustering tool for the recovery and maintenance of software system structures. In *Proceedings; IEEE International Conference on Software Maintenance* (1999), IEEE Computer Society Press, pp. 50–59.

[18] Mancoridis, S., Mitchell, B. S., Rorres, C., Chen, Y.-F., and Gansner, E. R. Using automatic clustering to produce high-level system organizations of source code. In *International Workshop on Program Comprehension (IWPC'98)* (Los Alamitos, California, USA, 1998), pp. 45–53.

[19] Mitchell, B. S., and Mancoridis, S. On the automatic modularization of software systems using the bunch tool. 193–208.

[20] O'Keeffe, M., and O'Cinneide, M. Search-based software maintenance. In *Conference on Software Maintenance and Reengineering (CSMR'06)* (Mar. 2006), pp. 249–260.

[21] Osyczka, A. In *Multicriteria optimization for engineering design* (1985), Design Optimization, pp. 193–227.

[22] Papadimitriou, C. H., and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[23] Seng, O., Stammel, J., and Burkhart, D. Search-based determination of refactorings for improving the class structure of object-oriented systems. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (Seattle, Washington, USA, 8-12 July 2006), M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds., vol. 2, ACM Press, pp. 1909–1916.

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

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