Noname manuscript No.
(will be inserted by the editor)

A Study of the Bi-Objective Next Release Problem

Juan J. Durillo · Yuanyuan Zhang · Enrique Alba · Mark Harman · Antonio J. Nebro

the date of receipt and acceptance should be inserted later

Abstract One important issue addressed by software companies is to determine which features should be included in the next release of their products, in such a way that the highest possible number of customers get satisfied while entailing the minimum cost for the company. This problem is known as the Next Release Problem (NRP). Since minimizing the total cost of including new features into a software package and maximizing the total satisfaction of customers are contradictory objectives, the problem has a multi-objective nature. In this work, we apply three state-of-the-art multi-objective metaheuristics (two genetic algorithms, NSGA-II and MOCell, and one evolutionary strategy, PAES) for solving NRP. Our goal is twofold: on the one hand, we are interested in analyzing the results obtained by these metaheuristics over a benchmark composed of six academic problems plus a real world data set provided by Motorola; on the other hand, we want to provide insight about the solution to the problem. The obtained results show three different kinds of conclusions: NSGA-II is the technique computing the highest number of optimal solutions, MOCell provides the decision maker with the widest range of different solutions, and PAES is the fastest technique (but with the least accurate results). Furthermore, we have observed that the best solutions found so far are composed of a high percentage of low-cost requirements and of those requirements that produce the largest satisfaction on the customers as well.

Juan J. Durillo
Computer Science Department, University of Málaga (Spain)
E-mail: durillo@lcc.uma.es

Yuanyuan Zhang
King's College London CREST centre
E-mail: yuanyuan.zhang@kcl.ac.uk

Enrique Alba
Computer Science Department, University of Málaga (Spain)
E-mail: eat@lcc.uma.es

Mark Harman
King's College London CREST centre
E-mail: mark.harman@kcl.ac.uk

Antonio J. Nebro
Computer Science Department, University of Málaga (Spain)
E-mail: antonio@lcc.uma.es

1 Introduction

Traditionally, Search Based Software Engineering (SBSE) has been widely used in Software Testing [Afzal et al., 2009, Harman et al., 2009, Harman, 2007, McMinn, 2004] and comparatively less so in other fields of Software Engineering. However, there is evidence that SBSE techniques are starting to find their way into all aspects of software engineering activity from the earliest phases of the software development lifecycle concerned with requirements, management and planning right through to the post delivery phases of maintenance and re-engieeering [Harman et al., 2009]. This paper focuses on one of these early lifecycle problems, centred on requirements analysis.

In more traditionally considered areas of SBSE, the goal has tended to be one of finding the optimal or near optimal solution to the problem in hand, with respect to a single objective. For example, the very first work on SBSE [Korel, 1990, Miller and Spooner, 1976, Xanthakis et al., 1992] concerned itself with finding optimal or near optimal test sets.

However, more recently it has been realized that SBSE can also be used as a tool for decision support, using multi-objective approaches. In this mode, the search based approach can be used to provide insight to the Software Engineer, allowing him or her to explore the possible space of candidate solutions with certain properties, revealing structure in the solution space and potential points of attractive trade off. For example, recent work has considered multi-objective formulations of problems in testing [Del Grosso et al., 2005, Everson and Fieldsend, 2006, Lakhotia et al., 2007, Walcott et al., 2006, Yoo and Harman, 2007], quality assurance [Khoshgoftaar et al., 2004], refactoring [Harman and Tratt, 2007] and project management [Alba and Chicano, 2007] as well as requirements engineering [Finkelstein et al., 2008, Saliu and Ruhe, 2007, Zhang et al., 2007].

This focus on decision support has been technically underpinned by the re-formulation of many problems in SBSE as multi-objective problems, to which a Pareto optimal approach can be readily applied. In Pareto optimal approaches, the outcome of the search is not a single (optimal or near optimal solution). It is a *set of candidate solutions*, each of which cannot be improved upon according to one of the multiple objectives to be optimized without a negative impact on another. This set of solutions is called a 'non–dominated' set of solutions, because each is incomparable; no one solution dominates any other in terms of meeting the multiple objectives. In the Pareto optimal approach, all objectives are considered incomparable, so that weighting the different objectives in order to combine them into a single weighted sum objective, is impossible.

Any problem involving some form of cost–benefit analysis can be thought of as a canonical instance of the general class of problems for which a Pareto optimal approach is attractive. In any cost–benefit analysis, it will be hard to determine to what degree a decision maker can yield up a perceived benefit in order to reduce cost. Likewise, such a decision maker will not be able to readily decide, *a priori*, how much cost increase they would be prepared to tolerate for a commensurate increase in benefit. Cost and benefit simply are not like that; they require subjective human judgements to be made and depend on circumstances.

In such a potentially vague scenario space, in which the exercise of human judgement remains paramount, it may, at first, be difficult to see how one might successfully apply a search based technique. However, the key lies in the manner in which a Pareto approach presents a Pareto front of non–dominated solutions, each of which denotes a cost–benefit pair for which no other pair can be found which improves upon *both*

cost and benefit. The shape of this Pareto front gives insight to the decision maker. Though he or she may not be able to determine, *a priori*, the trade off they are prepared to accept between these two incompatible objectives, the shape of the front can, *a posteriori* direct them to points at which the trade off is most attractive.

In this paper we focus on the problem of helping the decision maker who is concerned with requirements analysis. Specifically, we seek to provide decision support for what has been termed the Next Release Problem (NRP). The problem here is *not* to tell the decision maker what requirement should be in the next release of the software; no self-respecting software engineer would entirely trust and rely upon an automated tool to make such a decision. Rather, the problem is to provide decision support, to help the manager to identify those solutions that best balance the competing concerns of cost and benefit. The approach we adopt is a Pareto optimal one, in which the decision maker supplies the tool with estimates of cost and assessments of benefit and the automated part of the analysis uses SBSE to search for a good Pareto front. A 'good' Pareto front is one which provides accuracy and diversity, as will be explained more formally later.

Furthermore, NRP has been shown to be an instance of the *Knapsack* problem, which is $\mathcal{NP}$-hard [Papadimitriou and Steiglitz, 1982], and, as a consequence, it cannot be solved efficiently by using exact optimization methods for large problem instances (e.g., artificial intelligence techniques such as A*). This situation makes necessary the application of techniques such as metaheuristics [Glover and Kochenberger, 2003], commonly used in SBSE. Although these kinds of algorithms do not ensure to find optimal solutions, they are able to obtain near-optimal solutions in a reasonable amount of time.

Metaheuristics can be integrated in automated tools for searching the enormous space of possibilities in order to present the decision maker with a front of candidate solutions that collectively denote the set of complementary best solutions that can be found. After that, the software engineer makes the ultimate decision as to how to proceed with and with which requirement set to proceed. Thus, this SBSE application scenario aims to draw on the complementary abilities of automated search and human domain knowledge.

This approach is not novel in requirements analysis, since multi-objective formulations of requirements analysis problems have been proposed by Zhang et al. [Zhang et al., 2007] and Saliu and Ruhe [Saliu and Ruhe, 2007]. However, there has been no previous in–depth empirical analysis of different algorithms to determine which provides the best results for this problem in terms of the efficiency (time to compute Pareto fronts) and effectiveness (quality and diversity of fronts). Clearly the quality and diversity of the Pareto front are vital to the approach if the decision maker is to be able to rely on an accurate assessment of the trade offs inherent in their particular instance of the problem. Furthermore, the performance of the algorithms will be important; it should any prove to be too slow to produce fronts in a reasonable amount of time and the decision maker will be unable to ask repeated 'what if' questions. This paper aims to provide just such an empirical study that answers these questions of efficiency and effectiveness for the Multi Objective Next Release Problem (MONRP). To come with this issue, three different multi-objective algorithms have been evaluated thorough this paper: NSGA-II [Deb et al., 2002] which is the reference algorithm in the field of multi-objective optimization, MOCell [Nebro et al., 2006] which has outperformed to NSGA-II in several studies, and PAES [Angeline et al., 1999] which is one

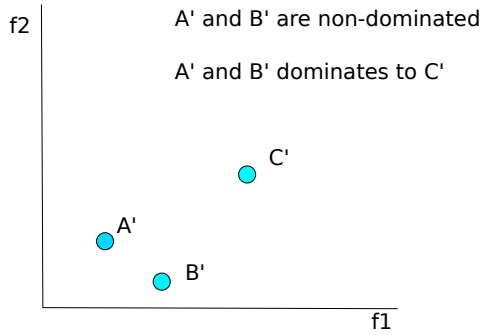


Fig. 1 Examples of dominated and non-dominated solutions.

of the simplest techniques in the field and does not require the adjustment of many parameters.

The remainder of this work is structured as follows: The next section contains some background about multi-objective optimization. Section 3 presents the Next Release Problem formally. The algorithms used in this work are described in Section 4. Section 5 is devoted to experimentation. We describe the obtained results in section 6, and we study the obtained solutions from the point of view of NRP in Section 7. In section 8, we describe and analyze the results obtained using a real world instance of the problem. Section 9 presents related work. Finally, Section 10 draws the main conclusions and lines of future work.

2 Multi-Objective Background

In this section, we provide the definition of some concepts for a better understanding of this work. In particular, we define the concept of multi-objective optimization problem (MOP), Pareto dominance and Pareto front. In these definitions we are assuming, without loss of generality, that minimization is the goal for all the objectives.

A general MOP can be formally defined as follows: find a vector $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ which satisfies the m inequality constraints $g_i(\mathbf{x}) \geq 0, i = 1, 2, \ldots, m$, the p equality constraints $h_i(\mathbf{x}) = 0, i = 1, 2, \ldots, p$, and minimizes the vector function $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T$, where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

The set of all values satisfying the constraints defines the *feasible region* Ω and any point $\mathbf{x} \in \Omega$ is a *feasible solution*.

Taking into account this definition of a MOP, a solution $x^1 = [x_1^1, x_2^1, ..., x_n^1]$ is said to dominate a solution $x^2 = [x_1^2, x_2^2, ..., x_n^2]$ if and only if $f_i(x^1) <= f_i(x^2)$ for $i = 1, 2, ..., m$, and there exist at least one j $(1 \leq j \leq m)$ such that $f_i(x^1) < f_i(x^2)$. Conversely, two points are said to be non-dominated whenever none of them dominates the other. Fig. 1 depicts some examples of dominated and non-dominated solutions. In this figure, A' dominates to C' because $f_1(A') < f_1(C')$, and $f_2(A') < f_2(C')$. Meanwhile, A' and B' are non-dominated because A' is better than B' in the first objective function $(f_1(A') < f_1(B'))$, but B' is better in the other objective function $(f_2(A') > f_2(B'))$.

The solution of a given MOP is usually a set of solutions (referred as Pareto optimal set) satisfying:

– Every two solutions into the set are non-dominated.
– Any other solution, y, is dominated by at least one solution in the set.

The representation of this set in the objective space is referred as *Pareto front*. Generating *Pareto front* is the main goal of multi-objective optimization techniques.

In theory, a Pareto front could contain a large number (or even infinitely many) points. In practice, a usable approximate solution will only contain a limited number of them; thus, an important goal is that they should be as close as possible to the exact Pareto front and uniformly spread, otherwise, they would not be very useful to the decision maker. Closeness to the Pareto front ensures that we are dealing with optimal solutions, while a uniform spread of the solutions means that we have made a good exploration of the search space and no regions are left unexplored.

Fig. 2 depicts these issues of convergence and diversity. The uppermost front depicts an example of good convergence and bad diversity: the approximation set contains Pareto optimal solutions but there are some unexplored regions of the optimal front. The approximation set depicted in the middle illustrates poor convergence but good diversity: it has a diverse set of solutions but they are not Pareto optimal. Finally, the lowermost front depicts an approximation front with both good convergence and diversity.

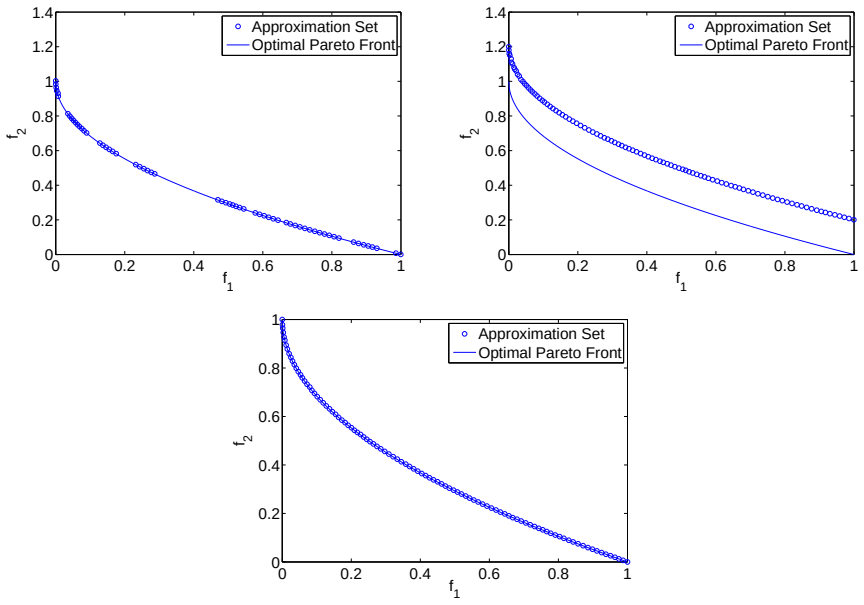


Fig. 2 Examples of Pareto fronts. From top to bottom, from left to right: (a) good convergence and bad diversity, (b) bad convergence and good diversity, and (c) good convergence and diversity.

3 Problem Statement

This section formalizes MONRP.

Given a software package, there is a set, C, of m customers whose requirements have to be considered in the development of the next release of this system. Each customer has associated a value, c_i, which reflects the customers' degree of importance to the software development company.

There is also a set, R, of n requirements to complete. In order to meet each requirement, resources must be spent, which can be transformed into an economical cost: the cost of satisfying the requirement. We denote this as r_j, $(1 \leq j \leq n)$ the economical cost of achieving the requirement j.

We also assume that more than one customer can be concerned with any requirement, and that all the requirements are not equally important for all the customers. In this way, associated with each customer and each requirement, there is a value v_{ij}, which represents the importance of the requirement j for the customer i. All these values can be represented by a matrix. Associated with the set R, there is a directed acyclic graph $G = (R, E)$, where $(r_i, r_k) \in E$ if and only if $r_i \in R$ is a prerequisite of $r_k \in R$ (i.e., it is mandatory to fulfill r_i before to r_k).

G is also transitive; then, if $(r_k, r_j) \in E$, and $(r_j, r_i) \in E$, the requirement r_k must be also fulfilled in order to afford r_i. In the special case where no requirement has any prerequisite, $E = \emptyset$, we say that the problem is basic.

The MONRP problem is to find a subset, R', of requirements which minimizes the cost and maximizes the total satisfaction of the customers with the finally included requirements. Thus the multi-objective Next Release Problem can be formalized as

$$minimize\ f_1 = \sum_{r_i \in R'} r_i \tag{1}$$

$$maximize\ f_2 = \sum_{i=1}^{n} c_i \sum_{r_j \in R'} v_{ij}. \tag{2}$$

Since minimizing a given function f is the same as maximizing $(-f)$, in this work we have considered the maximization of $(-f1)$ (i.e., the economical cost for companies), and $f2$ (i.e, the customer satisfaction).

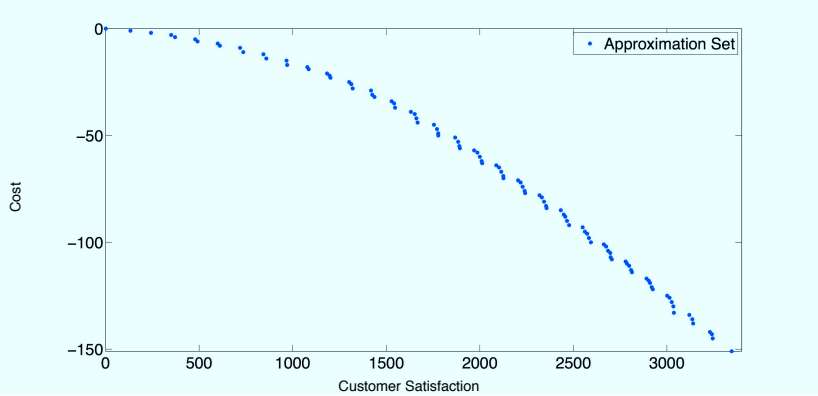


Fig. 3 Examples of Pareto front for NRP.

The advantages of considering NRP to be a multi-objective optimization problem, instead of a weighted single objective one, can be drawn from Fig. 3, which shows an example of front obtained for NRP. In this figure "Customer Satisfaction" means the total satisfaction of the customers, and "Cost" represents the economical cost of a requirement set. (Note that a negative cost means an investment made by the company). The decision maker is provided with a set of non-dominated solutions (all the solutions are equally important *a priori*) instead of a single solution.

This allows the decision maker to choose the best solution, depending on different situations and taking into account factors about which only he or she may be aware. For example, in difficult economic periods the decision maker would select solutions involving a lower cost for the company (solutions in the left part of the front); on the one hand, in times of high competition with other companies, the choice could be to select a solution which provides a high degree of customer satisfaction (solutions in the right part of the front). Meanwhile, in a single objective formulation of the problem, dealing with changes on these external situations may require a redefiniton of the considered weights, and to recompute the solution. Of course, these are relatively over–simplified scenarios, intended merely for illustrative purposes.

Moreover, the shape of the Pareto front could help the decision maker in working out relationships between the objectives. These relationships may yield insights into the nature of the problem, and also it can give a number of choices to select the most adequate solutions.

In practice, the decision maker will bring to the scenario a complex interwoven set of managerial, technical, sociological, economic and political concerns, for which it would be impractical to seek any machine–readable formulation. Nevertheless, the search based approach can complement this rich human domain knowledge, by setting out the best choices available, based solely on the cost–benefit analysis information provided. In this way, the machine and human work hand–in–hand. The machine focuses on what it does best (unbiased consideration of an enormous number of potential solutions, guided by cost–benefit data supplied by the human). The human considers the range of options selected in this purely mechanistic manner and uses the shape of the Pareto front to locate interesting locations in the solution space to which to direct further attention and consideration.

4 Solver Algorithms

In this section we describe the three algorithms which will be evaluated for solving NRP.

NSGA-II, proposed by K. Deb *et al.* [Deb et al., 2002], is a genetic algorithm which is the 'reference algorithm' in multi-objective optimization (with over 2,500 citations at the time of writing[1]). Its pseudocode is presented as Algorithm 1. NSGA-II makes use of a population (P) of candidate solutions (known as individuals). In each generation, it works by creating new individuals after applying the genetic operators to P, in order to create a new population, Q (lines 5 to 8). Then, both the current (P) and the new population (Q) are joined; the resulting population, R, is ordered according to a ranking procedure and a density estimator known as crowding distance (line 13) (for further details, please see [Deb et al., 2002]). Finally, the population P is updated

[1] Data from Google Scholar: 2,616 citations on 20^{th} September 2009.

Algorithm 1 Pseudocode of NSGA-II.

```
1: proc Steps_Up(nsga-II)        //Algorithm parameters in 'nsga-II'
2: P ← Initialize_Population() // P = population
3: Q ← ∅                          // Q = auxiliar population
4: while not Termination_Condition() do
5:    for i ← 1 to (nsga-II.popSize / 2) do
6:       parents←Selection(P);
7:       offspring←Recombination(nsga-II.Pc,parents);
8:       offspring←Mutation(nsga-II.Pm,offspring);
9:       Evaluate_Fitness(offspring);
10:      Insert(offspring,Q);
11:   end for
12:   R ← P ∪ Q
13:   Ranking_And_Crowding(nsga-II, R);
14:   P ←Select_Best_Individuals(nsga-II, R)
15: end while
16: end_proc Steps_Up;
```

with the best individuals in R (line 14). These steps are repeated until a termination condition is fulfilled.

MOCell (Multi-Objective Cellular Genetic Algorithm), introduced by Nebro *et al.* [Nebro et al., 2009], is a cellular genetic algorithm (cGA) which has proven to outperform NSGA-II in some studies [Nebro et al., 2009] [Nebro et al., 2007]. In cGAs, the concept of (small) *neighborhood* is paramount. This means that an individual may only cooperate with its nearby neighbors in the breeding loop. Overlapped small neighborhoods of cGAs help in exploring the search space because they induce a slow diffusion of solutions through the population, providing a kind of exploration (diversification). Exploitation (intensification) takes place inside each neighborhood by applying the typical genetic operations (crossover, mutation, and replacement).

MOCell includes an external archive to store the non-dominated solutions found as the algorithm progresses. This archive is limited in size and uses the crowding distance of NSGA-II to maintain diversity. The pseudocode of MOCell is presented as Algorithm 2, which corresponds with the version called aMOCell4, described in [Nebro et al., 2007].

We can observe that, in this version, for each individual we select one parent from its neighborhood and one from the archive, in order to guide the search towards the best solutions found (lines 5 to 8). Then a new solution is created by applying the genetic operators to these parents. The new solution is used to replace the current solution (line 11), and it is considered for inclusion in the archive (line 12). This constitutes a single iteration of the algorithm. The overall algorithm iterates until a termination condition is fulfilled.

PAES is a metaheuristic proposed by Knowles and Corne [Angeline et al., 1999]. The algorithm is based on a simple (1+1) evolution strategy. To find diverse solutions in the Pareto optimal set, PAES uses an external archive of nondominated solutions, which is also used to make decisions about new candidate solutions [Angeline et al., 1999]. An adaptive grid is used as a density estimator in the archive. The most remarkable characteristic of PAES is that it does not make use of any recombination operators (crossover). New solutions are generated only by modifying the current solution. The pseudocode of PAES is presented as Algorithm 3. It commences with a random solution (line 3). In each iteration, a new solution is produced by modifying the current solution (line 5). This new solution is included in the archive and it is considered as a potential

Algorithm 2 Pseudocode of MOCell.

```
1: proc Steps_Up(mocell)        //Algorithm parameters in 'mocell'
2: archive ← ∅ //Creates an empty archive
3: while not Termination_Condition() do
4:    for individual ← 1 to mocell.popSize do
5:       n_list←Get_Neighborhood(mocell,position(individual));
6:       parent1←Selection(n_list);
7:       parent2←Selection(archive);
8:       offspring←Recombination(mocell.Pc,parent1, parent2);
9:       offspring←Mutation(mocell.Pm,offspring);
10:      Evaluate_Fitness(offspring);
11:      Replacement(position(individual),offspring,mocell);
12:      Insert_Pareto_Front(offspring, archive);
13:   end for
14: end while
15: end_proc Steps_Up;
```

Algorithm 3 Pseudocode of PAES.

```
1: proc Steps_Up(paes)        //Algorithm parameters in 'paes'
2: archive ← ∅
3: currentSolution ← Create_Solution(paes) // Creates an initial solution
4: while not Termination_Condition() do
5:    mutatedSolution←Mutation(currentSolution);
6:    Evaluate_Fitness(mutatedSolution);
7:    if IsDominated(currentSolution, mutationSolution) then
8:       currentSolution ← mutatedSolution
9:    else
10:      if Solutions_Are_Nondominated(currentSolution, mutationSolution) then
11:         Insert(archive, mutatedSolution)
12:         currentSolution ← Select(paes, archive)
13:      end if
14:   end if
15: end while
16: end_proc Steps_Up;
```

replacement for the current solution (lines 7 to 14). These steps are repeated until the maximum number of evaluations is reached.

We have included PAES in our study because of its simplicity. PAES does not use any recombination operator, and its only parameter is the number of partitions of the adaptive grid of the archive. Its relative simplicity makes it attractive since there are comparatively few parameters that require tuning in order to know that the algorithm is being applied properly (e.g., population size, crossover probability, mutation probability).

5 Experimental Method

This section is aimed at presenting the indicators used to measure the quality of the obtained results and the benchmark problems we have used. It also describes how the solutions of the problem have been encoded and the genetic operators employed, the configuration of the algorithms, and the methodology we have followed.

5.1 Quality Indicators

Two different issues are normally taken into account for assessing the quality of the results computed by a multi-objective optimization algorithm:

1. To minimize the distance of the computed solution set by the proposed algorithm to the optimal Pareto front (convergence towards the optimal Pareto front).
2. To maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible (diversity).

A number of quality indicators have been proposed in the literature. Among them, we can distinguish between *Pareto compliant* and *non Pareto compliant* indicators [Knowles et al., 2006]. Given two Pareto fronts, A and B, if A dominates B, the value of a Pareto compliant quality indicator is higher for A than for B; meanwhile, this condition is not fulfilled by the non–compliant indicators. Thus, the use of Pareto compliant indicators should be preferable; however, non Pareto compliant indicators can also be used for measuring some particular features of a front. In this work, we apply an indicator of each type: Hypervolume [Zitzler and Thiele, 1999] (Pareto compliant), which takes into account the convergence as well as the diversity of the solutions; and Spread [Deb, 2001] (non Pareto compliant), which measures the distribution of solutions into a given front. Both indicators are defined as follows:

– **Hypervolume (HV).** This indicator calculates the volume (in the objective space) covered by members of a non-dominated set of solutions Q (the region enclosed into the discontinuous line in Figure 4, $Q = \{A, B, C\}$) for problems where all objectives are to be minimized. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume (HV) is calculated:

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{3}$$

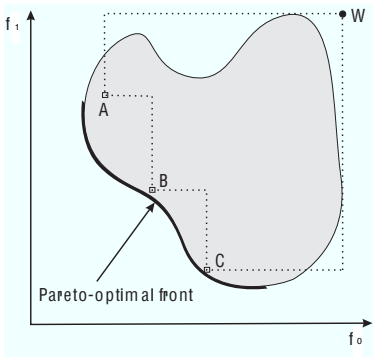


Fig. 4 The hypervolume enclosed by the non-dominated solutions.

In this case, we also apply this metric after a normalization of the objective function values to the range [0..1]. A Pareto front with a higher HV than another one could be due to two things: some solutions in the better front dominate solutions in the other, or, solutions in the better front are more widely distributed than in the other. Since both properties are considered to be good, algorithms with larger values of HV are considered to be desirable.

– **Spread or Δ.** The *Spread* indicator is a diversity quality indicator that measures the extent of spread by the set of computed solutions. Δ is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \quad , \tag{4}$$

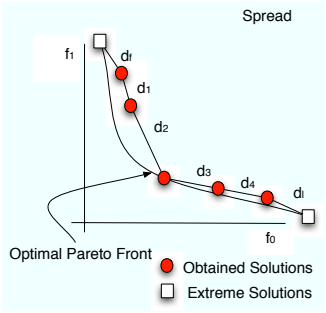


Fig. 5 Distances from the extreme solutions.

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* solutions of the optimal Pareto front in the objective space (see Figure 5 for a pictorial description). This metric takes a zero value for an ideal distribution, denoting a perfect spread of the solutions in the Pareto front. We apply this metric after a normalization of the objective function values to the range [0..1]. Pareto fronts with a smaller Δ value are considered more desirable:

To apply these quality indicators, it is usually necessary to know the optimal Pareto front. Of course, typically, we do not know the location of the optimal front. Therefore, we employ as a 'reference Pareto optimal front' the front composed of all the non-dominated solutions out of all the executions carried out (i.e., the best front known for these problems until now). We also consider the number of solutions that are non-dominated with respect to all the solutions computed by all the algorithms.

5.2 Test Problems

In this section, we describe the test problems used to evaluate the performance of NSGA-II, MOCell, and PAES.

The three algorithms were applied to a set composed of six test problems. These problems have been aimed at covering both 'typical' and non 'typical' cases of NRP. Thus, we have generated instances ranging from 2 to 100 customers, and from 20 to 200 requirements. All the values related to each instance have been generated by sampling a random uniform distribution. We have not considered dependencies among requirements.

All the instances have the nomenclature *c_r*, where *c* is the number of customers, and *r* the maximum number of requirements. Specifically, we have considered here the same instances proposed by hang et al. [Zhang et al., 2007]: 15_40, 50_80, 2_200, 100_20, 100_25, and 100_40.

5.3 Solution Encoding and Genetic Operators

As described in Section 3, a solution to the problem consists in selecting a subset of requirements to be included in the next release of the software package. In this work, each solution is encoded as a binary string, s, of length n (the maximum number of requirements), where $s_i = 1$ means that the requirement i is included in the next release of the software.

As to the genetic operators, we have used *binary tournament* as the selection scheme. This operator works by randomly choosing two individuals from the population and the one dominating the other is selected; if both solutions are non-dominated one of them is selected randomly. We also use *single point crossover* as crossover operator. It works by creating a new solution in which the binary string from the beginning of a parent solution to a crossover point, randomly chosen, is copied from that parent while the rest is copied from other parent. Finally, the mutation operator used is *random mutation* using which some random bits of the string are flipped.

5.4 Configuration

All approaches were run for a maximum of 25,000 function evaluations, and the results are analyzed when 5,000, 10,000, and 25,000 evaluations have been performed.

The initial population was set to 100 in NSGA-II and MOCell. In MOCell, the archive size was also limited to 100 solutions. In both algorithms the probability of the crossover operator was set to $P_c = 0.9$ and the probability of the mutation operator to $P_m = 1/n$, being n the number or requirements. In PAES, the maximum size of the archive was also set to 100, and the number of divisions of the adaptive grid to 5.

All the algorithms have been implemented using jMetal [Durillo et al., 2006], a Java framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems.

5.5 Methodology

We have executed 100 independent runs for each algorithm and each problem instance. Since we are dealing with stochastic algorithms, we need to perform a statistical analysis of the obtained results to compare them with a certain level of confidence. Next, we describe the statistical test that we have carried out for assuring this. First, a Kolmogorov-Smirnov test is performed in order to check whether the values of the results follow a normal (Gaussian) distribution. If so, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is performed; otherwise we perform a Welch test. For non-Gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms. All the tables include the mean and standard deviation of the evaluated indicator.

We always consider in this work a confidence level of 95% (i.e., significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Those tests in which the statistical confidence has been achieved are marked with "+" symbols in the last

row in the tables containing the results; conversely, "−" means that we cannot assure anything about the statistical confidence of the results (p-value > 0.05).

For the sake of a better visual comprehension, the best result for each problem is depicted with a grey background.

6 Experimental Analysis

In this section we present the obtained results by the three evaluated algorithms. We start by describing the values of the HV and Δ, the two quality indicators used. Then we consider how many of the computed solutions are among the best solutions found so far. Finally, we have also included the running time of the algorithms.

6.1 Hypervolume Results

Tables 1, 2 and 3 contain the mean and standard deviation for the HV indicator when 5,000, 15,000, and 25,000 function evaluations have been performed, respectively. For this indicator, the higher the value, the better the quality of the obtained results. Thus, by looking at the tables, we can see that NSGA-II has been the algorithm computing the best results regarding to this indicator when only 5,000 function evaluations have been performed. However, when the number of evaluations increase, the differences between it and MOCell reduce. After 25,000 evaluations, MOCell outperforms NSGA-II for half of the problem instances. PAES is the worst algorithm according to this comparison. In all cases, the difference in the performance of the best algorithm and that of the others is statistically significant.

Table 1 Mean ($\bar{x}$) and standard deviation (σ) of the results of the HV quality indicator after 5,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$6.55e-01_{3.8e-03}$	$6.54e-01_{3.3e-03}$	$6.33e-01_{1.5e-02}$	+
50_80	$5.62e-01_{7.1e-03}$	$5.59e-01_{6.5e-03}$	$5.24e-01_{1.4e-02}$	+
2_200	$4.61e-01_{1.0e-02}$	$4.58e-01_{1.0e-02}$	$4.17e-01_{1.6e-02}$	+
100_20	$6.12e-01_{9.9e-04}$	$6.12e-01_{5.7e-04}$	$6.04e-01_{6.6e-03}$	+
100_25	$6.31e-01_{2.4e-03}$	$6.30e-01_{1.8e-03}$	$6.13e-01_{1.1e-02}$	+
100_140	$5.01e-01_{9.2e-03}$	$4.96e-01_{7.7e-03}$	$4.46e-01_{1.8e-02}$	+

Table 2 Mean ($\bar{x}$) and stantard deviation (σ) of the results of the HV quality indicator after 10,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$6.63e-01_{1.6e-03}$	$6.63e-01_{1.1e-03}$	$6.45e-01_{1.6e-02}$	+
50_80	$5.88e-01_{4.9e-03}$	$5.87e-01_{4.1e-03}$	$5.40e-01_{1.5e-02}$	+
2_200	$5.22e-01_{7.6e-03}$	$5.13e-01_{6.8e-03}$	$4.50e-01_{1.0e-02}$	+
100_20	$6.13e-01_{2.3e-04}$	$6.13e-01_{3.1e-04}$	$6.05e-01_{6.8e-03}$	+
100_25	$6.35e-01_{7.6e-04}$	$6.35e-01_{4.5e-04}$	$6.18e-01_{1.1e-02}$	+
100_140	$5.41e-01_{7.1e-03}$	$5.34e-01_{5.2e-03}$	$4.69e-01_{1.4e-02}$	+

Table 3 Mean ($\bar{x}$) and standard deviation (σ) of the results of the HV quality indicator after 25,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$6.65e-01_{4.4e-04}$	$6.66e-01_{1.6e-04}$	$6.51e-01_{1.7e-02}$	+
50_80	$6.04e-01_{1.4e-03}$	$6.04e-01_{1.2e-03}$	$5.60e-01_{1.0e-02}$	+
2_200	$5.78e-01_{3.3e-03}$	$5.68e-01_{3.8e-03}$	$4.83e-01_{1.2e-02}$	+
100_20	$6.13e-01_{4.6e-05}$	$6.13e-01_{5.8e-05}$	$6.08e-01_{5.6e-03}$	+
100_25	$6.36e-01_{1.5e-04}$	$6.36e-01_{9.1e-05}$	$6.23e-01_{1.1e-02}$	+
100_140	$5.73e-01_{3.2e-03}$	$5.68e-01_{3.1e-03}$	$4.95e-01_{1.3e-02}$	+

As described in Section 5.1, the HV indicator measures the non-dominated area covered by a front; thus, the higher the HV value, the larger this area, and, hence, the number of solutions dominated by it. Obviously, the optimal Pareto front has the highest HV value, and the fronts computed by an algorithm should converge towards that value. If we analyze the HV obtained by NSGA-II and MOCell when 10,000 and 25,000 evaluations have been performed, we observe that the differences of the HV value are smaller in the instances with 40 or fewer requirements than in the instances with more requirements. This means that in the first group of instances both algorithms have converged towards an optimal (local or global) Pareto front of the problem. Meanwhile, in the second group, it is still possible to improve the computed fronts.

Fig. 6 clarifies this point. It shows the evolution of the HV of the approximated fronts computed by NSGA-II over the number of evaluations carried out in the instances 100_20 and 2_200. In this figure, we can observe that, in the instance with only 20 requirements, the HV indicator has converged towards a fixed value when approximately 4,500 evaluations have been carried out. On the other hand, in the 2_200 instance, the HV indicator increases with the number of evaluations and has not converged towards a fixed value in 25,000 function evaluations; therefore it is possible to compute better fronts by performing a higher number of evaluations. This observation suggests that the instances with more requirements need a higher number of evaluations than the smaller instances, in order to converge towards the optimal Pareto front.

Thus, in summarizing all these results, some conclusions regarding the HV indicator emerge:

- NSGA-II has been the overall best algorithm.
- NSGA-II has been the fastest algorithm in obtaining an accurate set of solutions (it has obtained the best values taking into account only 5,000 function evaluations).
- When the number of solutions increases, the differences between NSGA-II and MOCell reduce, and MOCell matches the effectiveness of NSGA-II.
- PAES has computed the least accurate results in this comparison.
- The higher the number of requirements, the harder the problem, with some evidence that the number of requirements has more bearing on problem difficulty than the number of customers.

6.2 Δ Results

We focus now in analyzing the Δ quality indicator, whose values when 5,000, 10,000, and 25,000 evaluations have been performed are included in Tables 4, 5 and 6, respectively. In this indicator, lower values denote better results. The tables show that

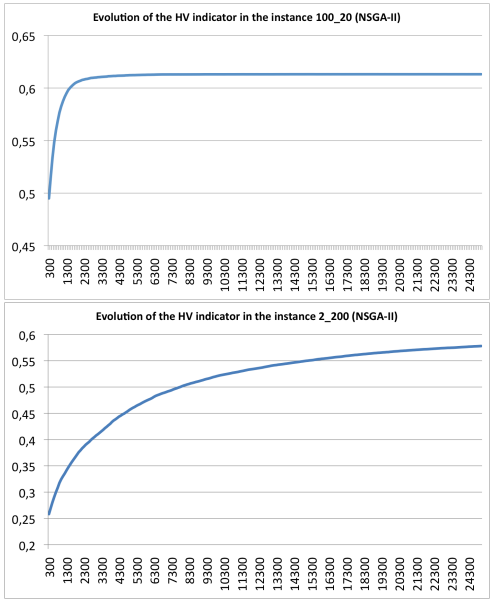


Fig. 6 Evolution of the HV indicator over the number of evaluations in instances 100_20 (top) and 2_200 (bottom).

Table 4 Mean ($\bar{x}$) and standard deviation (σ) of the results of the Δ quality indicator after 5,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$6.28e-01_{3.4e-02}$	$5.11e-01_{4.0e-02}$	$9.50e-01_{6.0e-02}$	+
50_80	$5.91e-01_{4.0e-02}$	$5.68e-01_{5.1e-02}$	$9.35e-01_{3.7e-02}$	+
2_200	$7.69e-01_{4.9e-02}$	$7.24e-01_{3.7e-02}$	$9.60e-01_{3.3e-02}$	+
100_20	$8.23e-01_{2.2e-02}$	$6.25e-01_{1.5e-02}$	$1.09e+00_{5.0e-02}$	+
100_25	$6.60e-01_{3.1e-02}$	$6.39e-01_{2.4e-02}$	$9.66e-01_{5.9e-02}$	+
100_140	$6.67e-01_{5.2e-02}$	$6.49e-01_{4.6e-02}$	$9.48e-01_{3.9e-02}$	+

Table 5 Mean ($\bar{x}$) and standard deviation (σ) of the results of the Δ quality indicator after 10,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$5.04e-01_{3.6e-02}$	$3.84e-01_{3.5e-02}$	$8.98e-01_{6.0e-02}$	+
50_80	$4.90e-01_{3.8e-02}$	$4.04e-01_{3.1e-02}$	$8.84e-01_{4.6e-02}$	+
2_200	$6.09e-01_{3.3e-02}$	$5.67e-01_{3.8e-02}$	$9.16e-01_{3.1e-02}$	+
100_20	$7.99e-01_{1.1e-02}$	$6.15e-01_{5.4e-03}$	$1.05e+00_{4.8e-02}$	+
100_25	$5.85e-01_{2.6e-02}$	$5.38e-01_{1.9e-02}$	$9.31e-01_{6.7e-02}$	+
100_140	$5.54e-01_{3.5e-02}$	$4.74e-01_{3.5e-02}$	$9.07e-01_{3.6e-02}$	+

MOCell is the algorithm obtaining the best results in all the cases. After MOCell, NSGA-II is second best. All result comparisons are statistically significant.

In summary, some observations about the results emerge:

– MOCell was the algorithm computing the fronts with the best distribution of solutions in all the cases.

Table 6 Mean ($\bar{x}$) and standard deviation (σ) of the results of the Δ quality indicator after 25,000 evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$4.67e-01_{3.8e-02}$	$2.49e-01_{2.2e-02}$	$8.40e-01_{8.1e-02}$	+
50_80	$4.07e-01_{3.5e-02}$	$2.33e-01_{2.7e-02}$	$8.37e-01_{4.4e-02}$	+
2_200	$4.74e-01_{3.3e-02}$	$3.28e-01_{3.3e-02}$	$8.84e-01_{2.7e-02}$	+
100_20	$7.93e-01_{5.5e-03}$	$6.15e-01_{7.0e-04}$	$1.02e+00_{3.7e-02}$	+
100_25	$5.35e-01_{1.6e-02}$	$4.96e-01_{1.1e-02}$	$8.84e-01_{6.7e-02}$	+
100_140	$4.34e-01_{3.8e-02}$	$2.93e-01_{2.8e-02}$	$8.72e-01_{3.9e-02}$	+

Table 7 Mean ($\bar{x}$) and standard deviation (σ) of the number of non dominated solutions found after 5,000 function evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$1.85e+01_{5.8e+00}$	$1.25e+01_{5.5e+00}$	$3.03e+00_{2.4e+00}$	+
50_80	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	-
2_200	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	-
100_20	$6.54e+01_{2.5e+00}$	$6.39e+01_{3.0e+00}$	$3.33e+01_{6.8e+00}$	+
100_25	$4.39e+01_{5.6e+00}$	$4.09e+01_{7.4e+00}$	$1.15e+01_{5.0e+00}$	+
100_140	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	-

– PAES has been again the algorithm obtaining the poorest results in accuracy.

An example of the computed fronts for the instances 15_40 and 2_200 by the different algorithms is depicted in Fig. 7. In the instance with 40 requirements (Fig. 7 (top)), we see that the solutions computed by PAES are close to the ones computed by the other techniques; however, NSGA-II and MOCell cover a large number of configurations. In the instance 2_200 (Fig. 7 (bottom)), we observe that the fronts computed by NSGA-II and MOCell dominate the one provided by PAES; furthermore, in this case we can observe the advantages of a front with a good diversity: MOCell has produced a better spread of solutions over the entirety of the Pareto front, while also covering a higher range of different configurations.

Looking again to those figures, we can see that MOCell has been able of computing non-dominatd solutions in areas where NSGA-II and PAES have not found any of them (solutions in the extremes of the Pareto front). This is related to a better exploration of the search space by MOCell. In fact, this is one of the properties of the cellular GA model, in which MOCell is based in, that has been reported in many studies on single-objective optimization (see [Alba and Dorronsoro, 2008]).

6.3 Number of Non Dominated Solutions Found

The instances used in this work have been hand-generated, and the optimal solutions to them are, *a priori*, unknown. Thus, we cannot be certain that the solutions computed by the algorithms evaluated in this work are optimal; hereinafter we refer to the set of all the non dominated solutions found 'so far' as final solutions.

Tables 7, 8 and 9 contain the number of final solutions computed by each algorithm for different degrees of effort (measured in terms of number of fitness evaluations 'so far'). Starting with Table 7, which shows that information when only 5,000 function

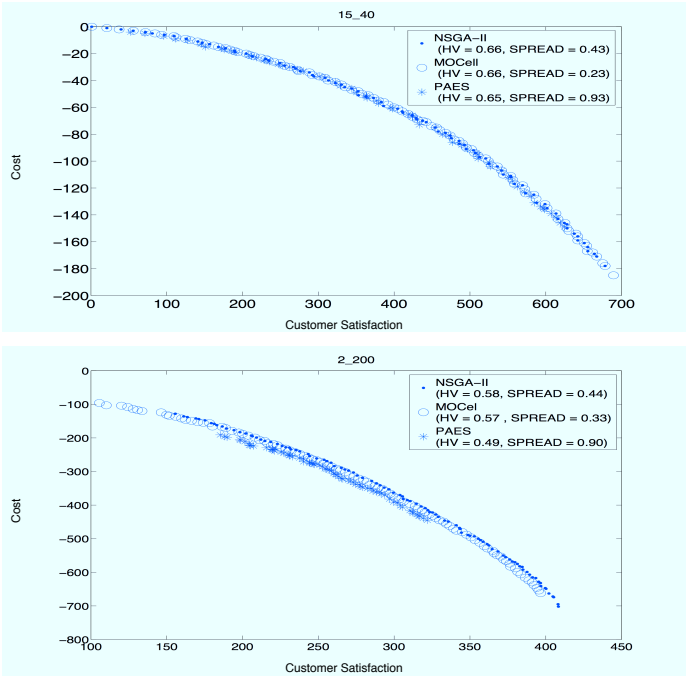


Fig. 7 Examples of the obtained front of solutions in instances 15_40 (15 customers and 40 requirements) and 2_200 (2 customers and 200 requirements).

Table 8 Mean ($\bar{x}$) and standard deviation (σ) of the number of non dominated solutions found after 10,000 function evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$4.38e+01_{5.5e+00}$	$3.65e+01_{6.9e+00}$	$7.98e+00_{3.8e+00}$	+
50_80	$2.20e-01_{6.3e-01}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	+
2_200	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	-
100_20	$6.80e+01_{1.7e+00}$	$7.26e+01_{1.7e+00}$	$3.99e+01_{6.1e+00}$	+
100_25	$6.22e+01_{4.8e+00}$	$6.59e+01_{4.7e+00}$	$1.84e+01_{5.3e+00}$	+
100_140	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	-

Table 9 Mean ($\bar{x}$) and standard deviation (σ) of the number of non dominated solutions found after 25,000 function evaluations.

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$4.83e+01_{4.6e+00}$	$5.98e+01_{4.7e+00}$	$1.53e+01_{4.4e+00}$	+
50_80	$1.91e+00_{3.4e+00}$	$8.30e-01_{1.1e+00}$	$0.00e+00_{0.0e+00}$	+
2_200	$1.02e+00_{8.9e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	+
100_20	$6.83e+01_{1.5e+00}$	$7.49e+01_{2.6e-01}$	$4.45e+01_{4.3e+00}$	+
100_25	$7.28e+01_{3.3e+00}$	$8.27e+01_{3.9e+00}$	$2.72e+01_{6.1e+00}$	+
100_140	$1.02e+00_{3.2e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	+

evaluations have been carried out, we observe that none of the algorithms has computed final solutions. It is also worth noting that the number of computed final solutions diminish when the number of requirements increases. Among the three algorithms, NSGA-II has been the one computing a higher number, followed by MOCell. The same comments are applicable when 10,000 function evaluations have been carried out (Table 8). However, in this case some final solutions have been computed by NSGA-II in the instance with 80 requirements. Statistical confidence has been found in all the cases where final solutions are obtained.

Finally, consider Table 9, where the number of final solutions found after 25,000 evaluations is included. In this table, we observe that NSGA-II was the only algorithm computing final solutions to the instances with more than 80 requirements. Meanwhile, in the instances with 40 or fewer requirements, MOCell has outperformed NSGA-II for the first time taking into account this indicator.

6.4 Running Time

We have also analyzed the running time required by the algorithms. All the time values are included in Tables 10, 11, and 12, which show the time in milliseconds to perform 5,000, 10,000, and 25,000 function evaluations, respectively. These values refer to execution on an Intel iQ7 processor at 2.8 GHz, with 6 GB RAM memory, running linux (kernel version 2.6.28-15) and the Java Virtual Machine provided by Sun (jdk version 1.6.0_14).

Considering the values shown in these tables, we observe that PAES was the fastest algorithm in our comparison in practically all the instances; only in the instance with 140 requirements and 100 customers the running times of PAES and NSGA-II are comparable. Notwithstanding this, it is important to notice that, in each case, all values are under one second. Looking at the time in each instance, we observe that the higher the number of requirements and customers, the higher the required time. The last column shows that the differences between the results are statistically significant.

Table 10 Mean ($\bar{x}$) and standard deviation (σ) of the running time after 5,000 function evaluations (ms).

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$4.57e+02_{3.3e+01}$	$5.01e+02_{2.6e+01}$	$2.36e+02_{1.9e+01}$	+
50_80	$6.17e+02_{2.7e+01}$	$7.84e+02_{3.3e+01}$	$4.70e+02_{1.9e+01}$	+
2_200	$1.16e+03_{2.8e+01}$	$1.40e+03_{4.3e+01}$	$1.01e+03_{2.1e+01}$	+
100_20	$4.72e+02_{3.7e+01}$	$5.07e+02_{1.6e+01}$	$2.44e+02_{1.8e+01}$	+
100_25	$5.10e+02_{3.7e+01}$	$5.92e+02_{3.4e+01}$	$2.99e+02_{2.7e+01}$	+
100_140	$1.25e+03_{5.9e+01}$	$1.41e+03_{4.5e+01}$	$1.16e+03_{4.6e+01}$	+

7 Studying the Computed Solutions

In the previous section, we have analyzed the quality of the solutions obtained when three different multi-objective optimizers are applied for solving NRP; in this section,

Table 11 Mean ($\bar{x}$) and standard deviation (σ) of the running time after 10,000 function evaluations (ms).

	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$6.21e+02_{2.6e+01}$	$7.54e+02_{3.0e+01}$	$3.43e+02_{2.0e+01}$	+
50_80	$1.01e+03_{4.5e+01}$	$1.23e+03_{4.4e+01}$	$8.22e+02_{4.1e+01}$	+
2_200	$2.20e+03_{4.8e+01}$	$2.58e+03_{5.3e+01}$	$2.01e+03_{3.2e+01}$	+
100_20	$6.49e+02_{3.2e+01}$	$6.90e+02_{3.5e+01}$	$3.80e+02_{3.5e+01}$	+
100_25	$7.36e+02_{4.2e+01}$	$8.63e+02_{3.8e+01}$	$4.82e+02_{4.0e+01}$	+
100_140	$2.22e+03_{7.2e+01}$	$2.57e+03_{7.0e+01}$	$2.23e+03_{1.3e+02}$	+

Table 12 Mean ($\bar{x}$) and standard deviation (σ) of the running time after 25,000 function evaluations (ms).

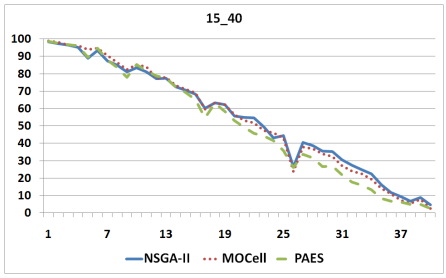
	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
15_40	$1.00e+03_{3.3e+01}$	$1.16e+03_{2.6e+01}$	$6.19e+02_{3.4e+01}$	+
50_80	$2.06e+03_{8.0e+01}$	$2.38e+03_{7.7e+01}$	$1.82e+03_{8.1e+01}$	+
2_200	$5.18e+03_{5.7e+01}$	$5.87e+03_{7.2e+01}$	$4.92e+03_{4.8e+01}$	+
100_20	$1.10e+03_{5.4e+01}$	$1.11e+03_{4.4e+01}$	$7.59e+02_{5.7e+01}$	+
100_25	$1.31e+03_{5.6e+01}$	$1.46e+03_{6.2e+01}$	$9.74e+02_{7.7e+01}$	+
100_140	$5.16e+03_{1.7e+02}$	$6.01e+03_{2.1e+02}$	$5.16e+03_{2.6e+02}$	+

we are interested in analyzing which requirements are included in the best solutions found by these algorithms.

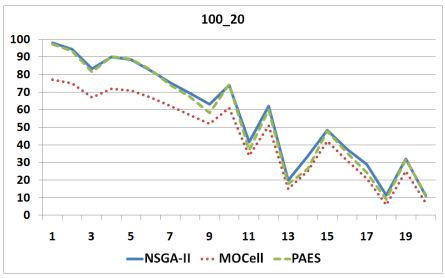
Let us suppose that a software engineer has to select a subset of requirements to be included in the next release of a software package and only wants to optimize the cost of fulfilling these requirements. In this situation, it is clear that the software engineer should include, in this set, those requirements that are cheaper. Something similar happens if the interest is only to maximize the satisfaction of the users of that system. Those requirements which satisfy the customer the most should be selected. But which requirements should be considered if the goal is to optimize both objectives? Intuitively, in this case, the optimal Pareto front should be composed of a set of solutions with different numbers of requirements, where those requirements offering a higher ratio of satisfaction per unit cost are more likely to be the candidate solutions found on the final Pareto front.

To analyze which requirements are included in the best solutions found by each algorithm, we have proceeded as follows. First, we sorted all the requirements by the ratio of satisfaction per unit cost (i.e., the satisfaction that each requirement provides to the customers divided by the cost of implementing it). Then, for each requirement we computed the mean of the times that it is included by each algorithm in each solution. Finally, we plotted this information in Fig. 8. The horizontal axis represents, in descending order, the requirements sorted by means of the ratio satisfaction by unit cost. The vertical axis represents the percentage of solutions which implement the requirement represented by the horizontal axis.

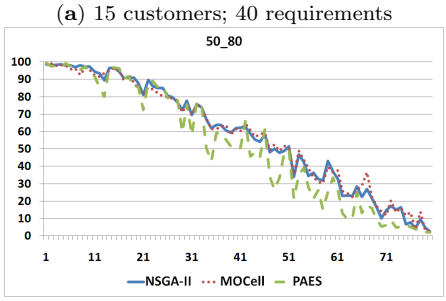
Let us start by analyzing the instance with the lowest number of requirements, i.e., that one known as 100_20. Fig. 8 -(d) shows the information related to this instance. Although there are some exceptions, there is also a clear tendency to include those requirements with a higher ratio of satisfaction per unit cost with more probability. Considering the differences between the algorithms, we make two further observations. On the one hand, NSGA-II and MOCell appear to have adopted consistently different



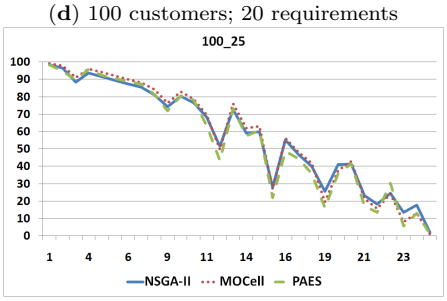
(**a**) 15 customers; 40 requirements



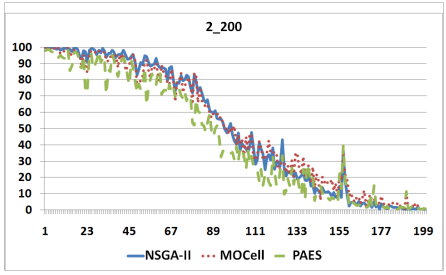
(**d**) 100 customers; 20 requirements



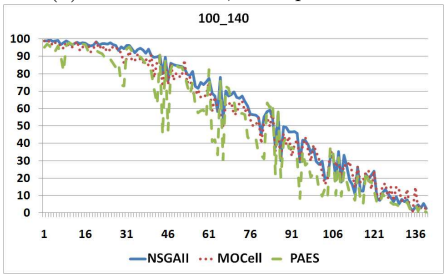
(**b**) 50 customers; 80 requirements



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



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



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

Fig. 8 Percentage of solutions which include each requirement (horizontal axis). Requirements are sorted by means of the ratio satisfaction provided by unit cost. In these graphics, the x-axis represents the requirements order by mean of ratio of provided customer satisfaction per unit cost; meanwhile, the y-axis represent the percentage of use of each requirements in the computed solutions.

design strategies. That is, each requirement appears in a higher number of configurations in NSGA-II than in MOCell. However, considering the results of the previous section, both have obtained very accurate results. On the other hand, NSGA-II and PAES have included each requirement the same number of times but, as Section 5 showed, NSGA-II has obtained fronts of better quality than PAES.

Figures 8 -(e), -(a), and, -(b) present the information belonging to the instances 100_25, 5_40, and 50_80, respectively. The requirements offering a higher ratio of satisfaction by unit cost are included in a higher number of solutions. Furthermore, NSGA-II and MOCell include each of these requirements as part of a solution more times than PAES. The requirements with the highest ratio have been used more by MOCell than by the other two algorithms, and the requirements with the smallest ratios were more used by NSGA-II.

Finally, the results for instances 100_140, and 2_200 are depicted in figures 8-(f), and -(c), respectively. We observe that there is also a tendency to use those requirements with a better ratio of satisfaction/cost in a higher number of solutions. However, we also see that these figures present many oscillations. These oscillations could potentially indicate that the algorithms have yet to fully converge, as we stated in Section 5.

Thus, in all the instances, the best solutions found so far by each algorithm are composed by those requirements which more satisfy the customers by unit cost. This fact corroborates the observations we made in the conference version of this paper [Durillo et al., 2009], where we stated that the better solutions found were composed of a high percentage of the cheapest requirements, and those which more satisfy the customers are generally those providing the higher ratio satisfaction per unit cost.

8 A Case Study: the Motorola Data Set

This section is aimed at solving a real world instance of NRP problem provided by a large international company, Motorola. We start by presenting the problem. After that, we analyze the obtained results in terms of the quality of the computed fronts, and also in terms of the composition of the solutions. Finally, we deep in the analysis of the obtained results.

8.1 Obtained Results

The Motorola data set concerns a set of 35 requirements for hand held communication devices. The stakeholders 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 maintain cost data in the form of the estimated cost of implementation of each requirement. Each of these stakeholders is equally important for the company (i.e., the value c_i is the same for $i = 1..4$). There exists a main difference between this problem and the test instances studied in this work; each requirement is only desired by one customer.

Table 13 Mean ($\bar{x}$) and standard deviation (σ) of the results of the HV quality indicator in the problem provided by Motorola.

Number of Evaluations	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
5,000	$7.78e-01_{2.1e-03}$	$7.77e-01_{2.6e-03}$	$7.45e-01_{3.5e-02}$	+
10,000	$7.83e-01_{5.6e-04}$	$7.82e-01_{6.6e-04}$	$7.52e-01_{3.5e-02}$	+
25,000	$7.84e-01_{1.3e-04}$	$7.84e-01_{1.5e-04}$	$7.62e-01_{2.2e-02}$	+

Table 13 includes the results of the HV, when 5,000, 10,000, and 25,000 function evaluations have been performed. As happened with the test problems, NSGA-II and MOCell computed the best fronts regarding this indicator. Actually, although the results are quite similar, when 25000 evaluations have been performed MOCell has been able to compute better fronts than NSGA-II. In fact, if we show the boxplot distribution of the HV values obtained for both algorithms (see Fig. 9), we can see that the values obtained by MOCell have been higher (then better) than the ones obtained by

Table 14 Mean ($\bar{x}$) and standard deviation (σ) of the results of the Δ quality indicator in the problem provided by Motorola.

Number of Evaluations	NSGA-II	MOCell	PAES	
	$\bar{x}_\sigma$	$\bar{x}_\sigma$	$\bar{x}_\sigma$	
5,000	$9.10e-01_{4.3e-02}$	$5.46e-01_{3.1e-02}$	$1.10e+00_{6.2e-02}$	+
10,000	$8.42e-01_{3.3e-02}$	$5.02e-01_{2.3e-02}$	$1.07e+00_{6.3e-02}$	+
25,000	$8.12e-01_{2.4e-02}$	$4.73e-01_{6.7e-03}$	$1.05e+00_{6.4e-02}$	+

NSGA-II. Furthermore, the non-overlapped notches in each box means that there is statistical significance between the obtained HV by both algorithms. Additionally, as the last column in the table indicates, the differences in the distributions of results have been statistically significant in all the cases.

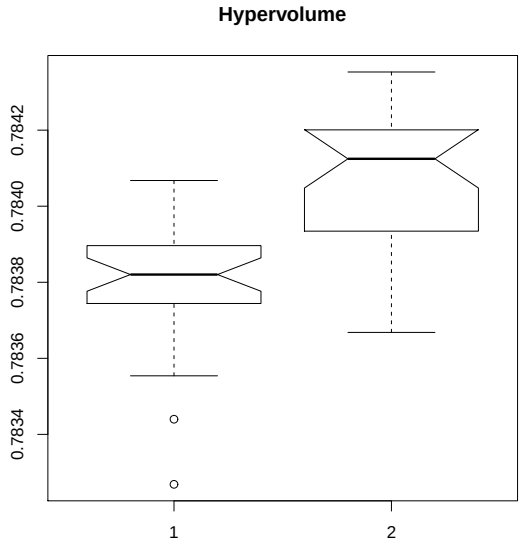


Fig. 9 Distribution of the HV value of the fronts computed by NSGA-II and MOCell.

We now consider the Δ values, presented in Tables 14. We observe that MOCell is the algorithm computing the fronts with the best value for this indicator; NSGA-II has obtained the second best values.

Thus, the values obtained by the algorithms in both indicators lead us to conclude that MOCell has been the most remarkable technique for solving this problem.

Fig. 10 shows examples of fronts computed by NSGA-II and MOCell after 25,000 evaluations. As we can observe, both algorithms have converged towards the same front. However, it is possible to see at a glance that MOCell has obtained a better distribution of solutions. These observations verify the results obtained by the quality indicators: both algorithms have obtained similar values of HV, but MOCell has outperformed to NSGA-II in the Δ indicator.

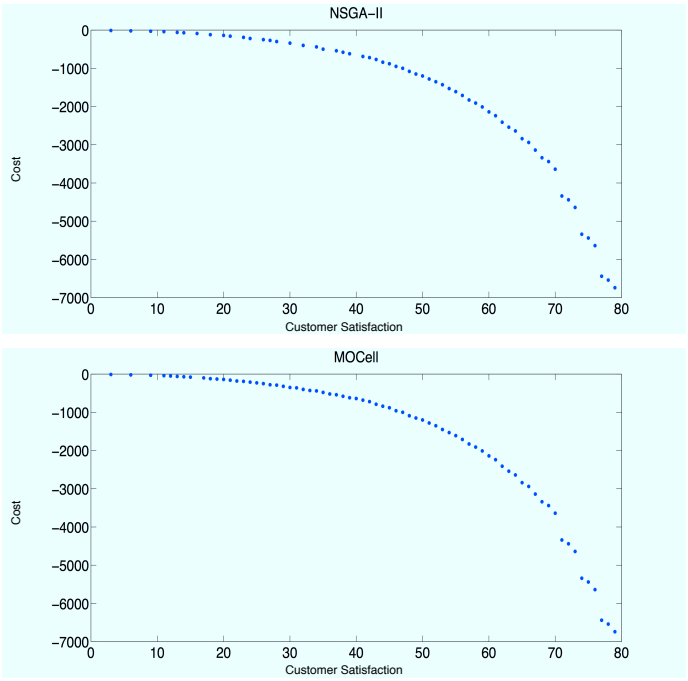


Fig. 10 Examples of fronts computed by NSGA-II (top) and MOCell (bottom) in the problem provided by Motorola.

Considering the shape of the fronts depicted in that figure, some conclusions can be drawn. On the one hand, we observe that by increasing the investment from 5 to 1000 cost units (vertical axis) it is possible to obtain a customer satisfaction between 0 and 50 satisfaction units. On the other hand, there is a section of the front along which an increase in customer satisfaction would require a much larger investment by the company. For example, increasing the customer satisfaction from 50 to 80 units requires an increase in investment of 700%.

With regard to the number of final solutions found by the algorithms, Table 15 summarizes the number of them found after performing 5,000, 10,000, and 25,000 evaluations. Also in this indicator, the algorithms behave in a similar manner to that in which they did for the test instances: when less than 10.000 function evaluations have been carried out, NSGA-II is the algorithm which finds the highest number of final solutions; however, when the number of evaluations increases MOCell outperforms to NSGA-II.

Table 15 Mean ($\bar{x}$) and standard deviation (σ) of the number of non dominated solutions found in the problem provided by Motorola.

Number of Evaluations	NSGA-II	MOCell	PAES	
5,000	$2.56e+01_{4.8e+00}$	$2.02e+01_{5.5e+00}$	$9.25e+00_{4.1e+00}$	+
10,000	$4.26e+01_{3.2e+00}$	$3.75e+01_{4.4e+00}$	$1.59e+01_{5.1e+00}$	+
25,000	$4.88e+01_{2.8e+00}$	$5.39e+01_{3.0e+00}$	$2.44e+01_{7.2e+00}$	+

The running times of the algorithms are presented in Table 16. In all the cases, the time required by PAES is less than half the time required by the other two algorithms. However, as we can see, all the times are under one second.

Table 16 Mean ($\bar{x}$) and standard deviation (σ) of the running time in the problem provided by Motorola (time is given in ms).

Number of Evaluations	NSGA-II	MOCell	PAES	
5,000	$3.53e+02_{1.5e+01}$	$3.71e+02_{5.9e+00}$	$1.68e+02_{8.1e+00}$	+
10,000	$4.81e+02_{6.3e+00}$	$4.96e+02_{6.3e+00}$	$2.42e+02_{9.4e+00}$	+
25,000	$8.00e+02_{5.7e+00}$	$8.01e+02_{1.6e+01}$	$4.37e+02_{1.4e+01}$	+

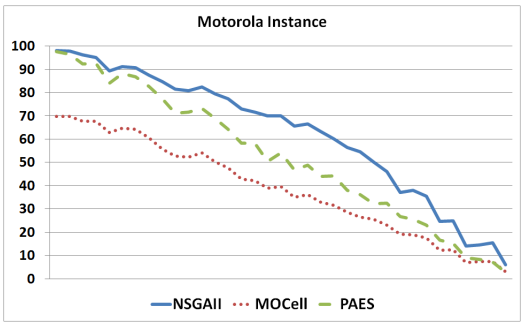


Fig. 11 Distribution of the use of requirements in the Motorola problem. Again, in this graphic, the x-axis represents the requirements order by mean of ratio of provided customer satisfaction per unit cost, and the y-axis represent the percentage of use of each requirements in the computed solutions.

Considering the composition of the computed solutions, Fig. 11 shows the distribution of the requirements used by each algorithm. We can see that those requirements with the better ratio satisfaction by unit cost are included in a higher number of solutions. This replicates the results observed for the test instances of the MONRP presented earlier.

Thus, the obtained results in this instance confirm the algorithm behavior observed in the previous test instances in terms of the quality of the solutions found, diversity of solutions and the composition of those solutions.

8.2 Post-Analysis of the Motorola Problem Results

In the last section, we focused on computing the Pareto front of solutions for the Motorola Problem and on analyzing the composition (in terms of the implemented requirements) of that front. In this section, we go an step forward on the analysis of the solutions computed by the most outstanding algorithm for that instance: MOCell. In particular, we want to analyze two different issues: on the one hand, we want to figure out if is there any relationship between a requirement and the cost and provided satisfaction of a solution which implements it; and, on the other hand, we are also

interested in the fairness (i.e., to what extent can a solution be shown to be a fair allocation of resources) of each solution into the computed Pareto front. We make use of the best front found so far by MOCell as the reference Pareto front.

The fairness of an allocation of resources has been previously studied in the SBSE field. Concretely, in [Finkelstein et al., 2008], a multi-objective algorithm has been used for computing solutions which maximizes the final satisfaction of customers, minimizes the cost, while maximizing the mean of fulfilled requirement for each customer, i.e., a new objective function is considered for measuring the fairness. The approach followed here is different: we compute the Pareto front of solutions only in terms of cost and satisfaction of the customers, and once it has been computed, we analyze how fair are those solutions.

Thus, summarizing, this section is aimed at answering to the next two questions, respectively:

- Does a requirement determine the place into the Pareto front where a solution implementing it is located?
- Can a given solution into the Pareto front benefiting a customer while damaging other ones?

For answering the first of the above suggested questions, we need to study if the implementation of a requirement and the objective functions (cost and provided customer satisfaction) are correlated. To come with this issue, we have made use of the Spearman's correlation coefficient.

The Spearman Rank Correlation [Kendall and Gibbons, 1990] statistical analysis test assess whether two measurement variables are correlated. The test consist in calculating the, so called, correlation coefficient, rs , which can take value between -1 and $+1$. A coefficient $rs = -1$ means two variables have a perfect negative correlation (as one increases, the other decreases). Accordingly, $rs = +1$ means two variables have a perfect positive correlation (as one increases, so does the other); $rs = 0$ means two variables are entirely independent; there is no correlation between them.

In our problem, a requirement can only have two states: it is implemented or not. As a consequence, to determine the correlation between each requirement and the objectives functions may have not sense. Thus, in order to study this issue we have group the requirements attending to the ratio of provided satisfaction by unit cost, and after that what we have done is to study the correlation between each group and the objective functions. This way, we can obtain insight knowledge about the relationship between each requirement and the objectives functions by means of its ratio.

Thus, taking into account the above, we have proceed as follows. First, we have sorted all the requirement by means of the ratio satisfaction per unit cost, and we have normalized the ratio to the interval $[0, 1]$; then, we have classify all the requirements in four different groups attending to the value of its ratio; this classification is based on the information provided by the instance and can be done before computing the optimal solutions. In particular, for this instance, we have computed the following groups: the first group, that we have called *very good*, consist of requirements having a ratio between 0.5 and 1; the second group, *good*, consist of requirements having a ratio between 0.1 and 0.5. The third group (*poor*) consist of those requirements whose ratio is lower than 0.1, and finally, the last group, that we have called *very poor*, consist of those requirements with a very small ratio.

Table 17 Spearman's Rank Correlation Between Requirements and Cost of Implementing

Group	correlation coefficient with cost	correlation coefficient with customer satisfaction
very good	0.19	0.4
good	0.53	0.83
poor	0.86	0.94
very poor	0.92	0.66

Once the groups have been determined, we have computed the correlation coefficient between each group and the solutions obtained. The obtained results are summarized in Table 17.

Attending to that table, we can observe that for the requirements classified as *very good*, the correlation coefficient with the cost and provided satisfaction are 0.19 and 0.4, respectively. This means that the use of requirements belonging to this group are very weakly correlated with cost and provided satisfaction, and hence, they can be part of whatever solution regardless of them. In that table, it is also possible to observe that the correlation coefficient increases when the ratio value of a group is decreased. This indicates that the lowest the ratio, the stronger the correlation. As a consequence, a requirement with a small ratio of satisfaction by unit cost will be rarely present on solutions of low cost; meanwhile, they will be found more often in those solutions of higher cost and provided satisfaction.

Accordingly, the answer to the first question is that, for requirements with a high ratio it is not possible to determine a priori the cost and customer satisfaction of a solution implementing it; meanwhile, for requirements with a small ratio it is possible to estimate them. This result was somehow expected because, from the last section, we have that requirements with higher ratio than others are implemented in a higher number of solutions. Furthermore, it is reasonable that a good requirement (in terms of its ratio) should be taken in many different configurations, while a bad one (also in terms of its ratio) should be implemented in those solutions which maximize the provided satisfaction (regardless of the cost).

We turn now to analyze the fairness of the computed solutions. The idea here is to determine the correlation between the satisfaction achieved for the different customers. Thus, if the satisfaction of two customers, A and B, are correlated, we can argue that whenever A gets satisfied, B also achieves a high grade of satisfaction.

For computing the correlation coefficient between the different customers we have proceed as follows. First, for each solution into the Pareto front, we have computed the number of implemented requirements targeted by each customer. Then, we have normalized those values to the $[0, 1]$ interval. These values give us a measure of how many requirements have been implemented satisfying each customer in each solution. After that, we have computed the correlation coefficient between each pair of customers. Table 18 summarizes the computed values for the correlation coefficient.

Looking at this table, we see that the coefficient correlation is higher than 0.84 in practically all the cases. Actually, the values under 0.91 involves a comparison with customer 4, which is the most difficult to satisfy due to it is only interested in one out of the 35 requirements. This means that there is a strong correlation between the satisfaction achieved for each pair of customers. As a consequence, the interpretation of those results is that each solution into the Pareto front try to satisfy the customers the same.

Consequently, the answer to the second proposed question is that the solutions computed by MOCell are not only good solutions in terms of the objective functions,

Table 18 Spearman's Rank Correlation Between the Satisfaction of Different Customers

	Customer 2	Customer 3	Customer 4
Customer 1	0.95	0.95	0.85
Customer 2		0.91	0.84
Customer 3			0.85

but in addition they seem to be fair in terms of the satisfaction provided to each customer. This is desirable but unexpected result due to neither of the three algorithms employed in this work makes use of any information about the fairness of a solution.

9 Related Work

From the industry point of view, many companies feel that they cannot control the release planning challenge, because many of them may only rely on the product or project manager to investigate the implicit characteristics of requirements and study the competing interests of the stakeholders.

In the literature, Yeh and Ng [Yeh and Ng, 1990] argued that a target system benefited directly from ranking and prioritising requirements in 1990. Karlsson [Karlsson, 1996] adopted two types of techniques for selecting and prioritising software requirements: Quality Function Deployment (QFD) [Sullivan, 1986] and Analytical Hierarchy Process (AHP) [Saaty, 1980] in 1996. In QFD the stakeholders prioritize the requirements on an ordinal scale (using a numerical assignment). The drawback of this is that there is no clear and obvious definition of the distinction among the absolute number assigned to each requirement. Moreover, relationships between requirements are not supported by QFD. The most serious drawback is that QFD cannot manage functional requirements, because there is no degree of fulfilment for functional requirements.

In 1997, Karlsson and Ryan [Karlsson and Ryan, 1997] proposed a cost-value approach, using AHP, applying it to compare all the candidate requirements in order to determine which of the two is of higher priority and to what extent its priority is higher. Moreover, in 1998, they evaluated six different methods for selecting and prioritising requirements [Karlsson et al., 1998] and found that AHP is the most promising method. However, the disadvantage of using a pairwise comparison technique is the huge number of required pairwise comparisons. The method becomes laborious and inefficient as the scale of the project increases. In addition, this prioritizing process has a lack of support for requirement interdependencies.

AHP caters for human judgements that may be both partial and inconsistent in order to arrive at a robust requirement prioritization. The prioritization problem is clearly related to the Next Release Problem (NRP), because one could select a subset simply as a prefix of the prioritized sequence. However, such a subset selection cannot, by definition, be better than that which can be located by selection of requirements within the same budget and is less amenable to multi-objective generalization.

This is a motivation for the consideration of the separate, but related problem of requirements selection. Compared with priority-based methods, we can provide more than one (usually many) optimal alternative solutions within a certain criterion (such as under specific project budget). As such, the requirements engineer has the opportunity to observe the impact of including or excluding certain requirements,

and can use this to choose the best from the different alternatives, without affecting the quality of solutions. An analogous choice between prioritization and selection formulations can be found in SBSE approaches to regression test case selection and prioritization [Harman et al., 2009].

The work of Karlsson et al. has had an enormous impact in the field of requirements engineering and is now the underpinning of the popular tool FocalPoint, marketed by TeleLogic, which is now subsidiary of IBM.

Aiming to reduce the complexity of applying AHP, in 1998, Jung [Jung, 1998] adopted linear programming techniques as a two-step process: firstly, maximizing the sum of the requirements' values within cost budgets; secondly, determining which requirements can be fulfilled to minimize the sum of the costs within maintaining the maximum value of the first step. The process was based on single objective formulation with a cost factor as constraint. In 1999, Wiegers [Wiegers, 1999] presented a semi-quantitative approach for requirements prioritization, which combined value, cost and risk criteria using weighting factors to evaluate requirements. This approach was limited by the ability to attach the weights for each objective. Robertson and Robertson [Robertson and Robertson, 2000,Robertson and Robertson, 2006] presented the Volere Requirements Specification Template (`http://www.volere.co.uk/template.htm`) for assessing requirements. The template provided the simple and effective starting point for quantifying requirements, but it did not refer to the requirements selection issue.

Within the SBSE community, a recent trend has emerged in which search-based optimization techniques have been used to solve requirements selection and optimization problems. This would seem to be a natural and realistic extension of the initial work on SBSE.

The NRP was first formulated as a single-objective SBSE problem by Bagnall et al. in 2001 [Bagnall et al., 2001]. The paper described various metaheuristic optimization algorithms, including greedy algorithms, branch and bound, simulated annealing and hill climbing. The authors did not give any *value* property to each requirement. They only used an associated *cost*. The task of the work was to find a subset of stakeholders, whose requirements are to be satisfied. The objective was to maximize the cumulative measure of the stakeholder's importance to the company under resource constraints.

Feather and Menzies [Feather and Menzies, 2002] built an iterative model to seek the near-optimal attainment of requirements. The authors proposed a Defect Detection and Prevention (DDP) process based on a real-world instance: a NASA pilot study. The DDP combined the requirements interaction model with the summarization tool to provide and navigate the near-optimal solutions in the risk mitigation/cost trade-off space. The paper was one of the first to use Pareto optimality in SBSE for requirements, though, unlike the work in our paper, the Pareto fronts were not produced using multi-objective optimization techniques (as with more recent work), but were produced using the iterative application of a weighting based single objective formulation by applying simulated annealing.

Ruhe et al. [Greer and Ruhe, 2004] [Ruhe and Greer, 2003] [Ruhe and Ngo-The, 2004] proposed the genetic algorithm based approaches known as the EVOLVE family which aimed to maximize the benefits of delivering requirements in an incremental software release planning process. Their approaches balance the required and available resources; assessing and optimizing the extent to which the ordering conflicts with stakeholder priorities. They also took requirement changes and two types of requirements interaction relationship into account and provided

candidate solutions for the next release in an iterative manner. As with previous work, this piece of work still adopted a single objective formulation, taking the resource budget as a constraint. Jalali et al. [Jalali et al., 2008] also considered the problem of optimizing requirements. They used a Greedy Algorithm to explore optimal solutions that take account of risk in the requirements analysis process.

Moreover, Carlshamre [Carlshamre, 2002] take requirements interdependencies into consideration by using Linear Programming techniques. Ruhe and Saliu [Ruhe and Saliu, 2005] also presented an Integer Linear Programming (ILP) based method which combined computational intelligence and human negotiation to resolve their conflicting objectives. Van den Akker et al. [Li et al., 2007, van den Akker et al., 2004, van den Akker et al., 2005, van den Akker et al., 2008] further extended the technique and developed an optimization tool based on integer linear programming, integrating the requirements selection and scheduling for the release planning to find the optimal set of requirements with the maximum revenue against budgetary constraints.

Using search-based techniques in order to choose components to include in different releases of a system was studied by Harman et al. [Baker et al., 2006, Harman et al., 2006]. The work considered requirements problems as feature (component) subset selection problems, like Feather et al., presenting results for a single objective formulation applied to a real world data set: the Motorola Data Set. The work of AlBourae et al. [AlBourae et al., 2006] was focused more on the requirements change handling. That is, re-planning of the product release. A greedy replan algorithm was adopted to reduce risks and increase the number of requirements achieved in the search space under change.

In addition, Cortellessa et al. [Cortellessa et al., 2008a, Cortellessa et al., 2006, Cortellessa et al., 2008b] described an optimization framework to provide decision support for COTS and in-house components selection. The Integer Linear Programming (LINGO model solver) based optimization models (CODER, DEER) were proposed to automatically satisfy the requirements while minimizing the cost.

The aforementioned work on this problem has tended to treat the requirements selection and optimization 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. Single objective formulations have the drawback that the maximization of one concern may be achieved at the expense of the potential maximization of another resulting in a bias guiding the search to a certain part of the solution space.

More recently, there has been work on multi-objective formulations of the problem. Zhang et al. [Zhang et al., 2007] proposed a multi-objective formulation of the next release problem (NRP) to optimise value and cost, upon which we base the formulation in the present paper. This was the first paper to use a Pareto optimal, multi-objective approach to the NRP, migrating it from NRP to MONRP. Independently, at the same time, Saliu and Ruhe [Saliu and Ruhe, 2007] also adopted a Pareto optimal multi-objective approach to the related problem of balancing implementation objectives and requirements objectives. This was the first work to establish and study the link between requirements optimization and the corresponding tension with the implementation. All previous work had considered requirements in isolation, independent from the architectural constraints that choices of requirements impose upon the implementation of the chosen requirement set.

Finkelstein et al. [Finkelstein et al., 2008, Finkelstein et al., 2009] considered the problem of fairness analysis in requirements optimization. This was the first paper to introduce techniques for analysis of the trade-offs between different stakeholders' notions of fairness in requirements allocation, where there are multiple stakeholders with potentially conflicting requirement priorities and also possibly different views of what would constitute fair and equitable solution.

In this work, like others on multi-objective solutions, each of the objectives to be optimized is treated as a separate goal in its own right; multiple objectives are not combined into a single (weighted) objective function. This allows the optimization 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 maximization of the others. Using Pareto optimal search it becomes possible to explore precisely the *extent* to which it is possible to satisfy "all of the people all of the time". Of course, this is unlikely to be completely achievable. However, the algorithm attempts to produce a set of non-dominated solutions that are as close as the stakeholders' prioritizations will allow to this ideal situation.

As can be seen, the field of multi-objective requirements analysis is growing and developing into a well-defined subfield of activity within the overall areas of SBSE. A position paper on recent trends in requirements analysis optimization can be found in the work of Zhang et al. [Zhang et al., 2008].

Much of the previous work has been concerned with development of new models, formulations and frameworks for search based requirements. This previous work suggests that requirement analysis can be transformed from a purely qualitative process of human value judgement to a decision support environment in which human judgement is informed by quantitative assessments of choices, optimized using metaheuristic techniques. This growing interest in SBSE for requirement necessitates a more detailed empirical analysis of the algorithmic choices available to engineers seeking to use SBSE techniques in requirements analysis.

The present paper seeks to take a step toward the provision of this detailed empirical analysis. In the conference version of this paper [Durillo et al., 2009], we provided an initial set of empirical results to investigate the effectiveness of NSGA-II and MOCell for the MONRP. In the present journal version of the paper, we extend these previous results in the following ways:-

- We broaden the scope to include another multi-objective optimizer, PAES [Angeline et al., 1999].
- We analyze the development of Pareto fronts as the algorithms progress. Since all the algorithms are essentially 'anytime' algorithms, this analysis explores how quickly the algorithms converge to a final Pareto front and the increase in Pareto front quality as the algorithms progress.
- We extend the study to consider the efficiency (run time performance) of each of the algorithms studied.
- We consider, additionally, a real world case study of the MONRP, to see whether the empirical results suggested by the detailed study of different problem instances are borne out in practice.
- We further analyze the obtained fronts, not only in terms of the composition of each solution but also considering the fairness of each point into the Pareto front.

10 Conclusions and Future Work

In this paper we have studied the Next Release Problem, with the intention of analyzing the performance of three different multi-objective algorithms, and the solutions they have provided over both test cases and a real instance of the problem.

To come with those issues, we have evaluated three state-of-the-art multi-objective optimization algorithms: NSGA-II, MOCell, and PAES. This comparison has been done on the basis of two quality indicators, HV and Δ, the number of non-dominated solutions obtained by those algorithms, and running time.

In terms of convergence towards the optimal Pareto front, NSGA-II and MOCell have been the best solvers in our comparison. The former algorithm has obtained the best results, performing a lower number of evaluations. Furthermore, it has obtained the best fronts in the two instances with the highest number of requirements. Regarding the distribution of solutions contained in the fronts computed by the algorithms, MOCell has been the most outstanding algorithm in our comparison.

As to the number of obtained solutions, NSGA-II is also the algorithm which has shown the best performance; it is the technique computing the best non-dominated solutions found so far in the two biggest instances. If we attend to the composition of those solutions, we have observe that they are composed in a high percentage of those requirements offering the highest ratio of customer satisfaction by unit cost.

The simplest algorithm in our comparison, PAES, has been the fastest algorithm in our comparison. However, it has obtained the least accurate results according to all the indicators. This highlights the importance of both a population and the use of a recombination operator in order to better explore the search space of MONRP.

The best solutions found so far by the algorithms are composed by those requirements which more satisfy the customers by unit cost. Furthermore, we have observed that algorithms using the same number of times the same requirements can provide the software engineer with solutions of different quality.

Additionally, we have also made use of the computed front as a tool for analyzing the fairness of the obtained solutions. Specifically, we have analyzed the satisfaction of customers provided by each solution into the Pareto fronts computed by MOCell, the best algorithm for that instance. The results have shown that solutions into those fronts try to satisfy the customers the same.

Thus, by considering a multi-objective approach, it is possible to allow the software engineers to use Pareto front evaluation as a comparative tool for a number of objectives. First of all, the analyst can pick up the best solutions on the Pareto front in different circumstances based on their priorities. For example, at a specific budget level for a project, one or more optimal solutions might be found when moving along the Pareto front. Meanwhile, the stakeholders' satisfaction can also be concerned. Each satisfaction level has its own corresponding cost (resource allocation, spending) according to the Pareto front. This can help the analyst make rough estimates and adjustments for a project budget.

Moreover, the Pareto front not only gives solutions themselves, but also may yield interesting insights into the nature of the problem. The shape of the Pareto front (concave, convex, discontinuous, knee point, nadir point, etc.) reflects the structure of data in an intuitive way, and provides the analyst with very valuable information about the trade-off among the different objectives and helps fully understand the problem and reach practical solutions. Particularly, in the real instance we have identified areas of

the fronts where a small increment in customers satisfaction demanded a huge one in the investment.

Future work will verify these findings by applying search techniques to a larger number of real word problems. This will provide valuable feedback to researchers and practitioners in search techniques and in software engineering communities. Other formulations of the problem considering different sets of objectives and constraints and the design of techniques which assist software engineers in the decision making are also issues to study. This, in turn, may give rise to the need for the development of more efficient solution techniques. It is also interesting to investigate how these techniques scale when the number of requirements and/or customer increases. In order to reach this goal, a procedure will be needed which allows the systematic creation of instances with the desired features; in this sense, we plan to design a problem generator for MONRP instances.

Acknowledgements J. J. Durillo, A. Nebro, and E. Alba acknowledge founds from the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract P07-TIC-03044 DIRICOM project (`http://diricom.lcc.uma.es`), and the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01 (the M* project). J. J. Durillo is also supported by grant AP-2006-03349 from the Spanish Ministry of Education and Science. Mark Harman is partly supported by EPSRC grants EP/G060525 (CREST: Centre for Research on Evolution, Search and Testing, Platform Grant), and EP/D050863 (SEBASE: Software Engineering By Automated SEarch), which also fully supports Yuanyuan Zhang.

References

[Afzal et al., 2009] Afzal, W., Torkar, R., and Feldt, R. (2009). A systematic review of search-based testing for non-functional system properties. *Information and Software Technology*, 51(6):957 – 976.

[Alba and Chicano, 2007] Alba, E. and Chicano, J. F. (2007). Software project management with gas. *Information Sciences*, 177(11):2380 – 2401.

[Alba and Dorronsoro, 2008] Alba, E. and Dorronsoro, B. (2008). *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces*. Springer-Verlag Heidelberg.

[AlBourae et al., 2006] AlBourae, T., Ruhe, G., and Moussavi, M. (2006). Lightweight Replanning of Software Product Releases. In *Proceedings of the 1st International Workshop on Software Product Management (IWSPM '06)*, pages 27–34, Minneapolis, MN, USA. IEEE Computer Society.

[Angeline et al., 1999] Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors (1999). *The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation*, volume 1, Mayflower Hotel, Washington D.C., USA. IEEE Press.

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

[Baker et al., 2006] Baker, P., Harman, M., Steinhofel, K., and Skaliotis, A. (2006). Search based approaches to component selection and prioritization for the next release problem. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 176–185, Washington, DC, USA. IEEE Computer Society.

[Carlshamre, 2002] Carlshamre, P. (2002). Release Planning in Market-Driven Software Product Development: Provoking an Understanding. *Requirements Engineering*, 7(3):139–151.

[Cortellessa et al., 2008a] Cortellessa, V., Crnkovic, I., Marinelli, F., and Potena, P. (2008a). Experimenting the Automated Selection of COTS Components Based on Cost and System Requirements. *Journal of Universal Computer Science*, 14(8):1228–1255.

[Cortellessa et al., 2006] Cortellessa, V., Marinelli, F., and Potena, P. (2006). Automated Selection of Software Components Based on Cost/Reliability Tradeoff. In *Proceedings of the 3rd European Workshop on Software Architecture (EWSA '06)*, volume 4344 of *LNCS*, pages 66–81, Nantes, France. Springer.

[Cortellessa et al., 2008b] Cortellessa, V., Marinelli, F., and Potena, P. (2008b). An Optimization Framework for "Build-or-Buy" Decisions in Software Architecture. *Computers & Operations Research*, 35(10):3090–3106.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 1 edition.

[Deb et al., 2002] Deb, K. D., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

[Del Grosso et al., 2005] Del Grosso, C., Antoniol, G., Di Penta, M., Galinier, P., and Merlo, E. (2005). Improving network applications security: a new heuristic to generate stress testing data. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1037–1043, New York, NY, USA. ACM.

[Durillo et al., 2006] Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B., and Alba, E. (2006). jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos.

[Durillo et al., 2009] Durillo, J. J., Zhang, Y., Alba, E., and Nebro, A. J. (2009). A study of the multi-objective next release problem. In *SSBSE '09: Proceedings of the 2009 1st International Symposium on Search Based Software Engineering*, pages 49–58, Washington, DC, USA. IEEE Computer Society.

[Everson and Fieldsend, 2006] Everson, R. M. and Fieldsend, J. E. (2006). Multiobjective optimization of safety related systems: An application to short-term conflict alert. *IEEE Trans. Evolutionary Computation*, 10(2):187–198.

[Feather and Menzies, 2002] Feather, M. S. and Menzies, T. (2002). Converging on the optimal attainment of requirements. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 263–272, Washington, DC, USA. IEEE Computer Society.

[Finkelstein et al., 2008] Finkelstein, A., Harman, M., Mansouri, S. A., Ren, J., and Zhang, Y. (2008). "fairness analysis” in requirements assignments. In *RE '08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, pages 115–124, Washington, DC, USA. IEEE Computer Society.

[Finkelstein et al., 2009] Finkelstein, A., Harman, M., Mansouri, S. A., Ren, J., and Zhang, Y. (2009). A Search based Approach to Fairness Analysis in Requirement Assignments to Aid Negotiation, Mediation and Decision Making. *Requirements Engineering Journal (RE '08 Special Issue)*.

[Glover and Kochenberger, 2003] Glover, F. W. and Kochenberger, G. A. (2003). *Handbook of Metaheuristics*. Kluwer.

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

[Harman, 2007] Harman, M. (2007). The current state and future of search based software engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 342–357, Washington, DC, USA. IEEE Computer Society.

[Harman et al., 2009] Harman, M., Mansouri, S. A., and Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03.

[Harman et al., 2006] Harman, M., Skaliotis, A., and Steinhöfel, K. (2006). Search-based Approaches to the Component Selection and Prioritization Problem. In *Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, pages 1951–1952, Seattle, Washington, USA. ACM.

[Harman and Tratt, 2007] Harman, M. and Tratt, L. (2007). Pareto optimal search based refactoring at the design level. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1106–1113, New York, NY, USA. ACM.

[Jalali et al., 2008] Jalali, O., Menzies, T., and Feather, M. (2008). Optimizing requirements decisions with keys. In *PROMISE '08: Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 79–86, New York, NY, USA. ACM.

[Jung, 1998] Jung, H.-W. (1998). Optimizing Value and Cost in Requirements Analysis. *IEEE Software*, 15(4):74–78.

[Karlsson, 1996] Karlsson, J. (1996). Software Requirements Prioritizing. In *Proceedings of the Second International Conference on Requirements Engineering (RE '96)*, pages 110–116, Colorado Springs, CO, USA. IEEE Computer Society.

[Karlsson and Ryan, 1997] Karlsson, J. and Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, 14(5):67–74.

[Karlsson et al., 1998] Karlsson, J., Wohlin, C., and Regnell, B. (1998). An Evaluation of Methods for Prioritizing Software Requirements. *Information & Software Technology*, 39(14-15):939–947.

[Kendall and Gibbons, 1990] Kendall, M. and Gibbons, J. D. (1990). *Rank Correlation Methods*. A Charles Griffin Title, 5 edition.

[Khoshgoftaar et al., 2004] Khoshgoftaar, T. M., Liu, Y., and Seliya, N. (2004). Module-order modeling using an evolutionary multi-objective optimization approach. In *Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS '04)*, pages 159–169. IEEE Computer Society.

[Knowles et al., 2006] Knowles, J., Thiele, L., and Zitzler, E. (2006). A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

[Korel, 1990] Korel, B. (1990). Automated software test data generation. *IEEE Trans. Softw. Eng.*, 16(8):870–879.

[Lakhotia et al., 2007] Lakhotia, K., Harman, M., and McMinn, P. (2007). A multi-objective approach to search-based test data generation. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1098–1105, New York, NY, USA. ACM.

[Li et al., 2007] Li, C., van den Akker, M., Brinkkemper, S., and Diepen, G. (2007). Integrated Requirement Selection and Scheduling for the Release Planning of a Software Product. In *Proceedings of the 13th International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ '07)*, volume 4542 of *LNCS*, pages 93–108, Trondheim, Norway. Springer.

[McMinn, 2004] McMinn, P. (2004). Search-based software test data generation: a survey: Research articles. *Softw. Test. Verif. Reliab.*, 14(2):105–156.

[Miller and Spooner, 1976] Miller, W. and Spooner, D. (1976). Automatic generation of floating-point test data. *IEEE Transactions on Software Engineering*, 2:223–226.

[Nebro et al., 2006] Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2006). A cellular genetic algorithm for multiobjective optimization. In Pelta, D. A. and Krasnogor, N., editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25–36, Granada, Spain.

[Nebro et al., 2007] Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2007). Design issues in a multiobjective cellular genetic algorithm. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 126–140. Springer.

[Nebro et al., 2009] Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2009). Mocell: A cellular genetic algorithm for multiobjective optimization. *Int. J. Intell. Syst.*, 24(7):726–746.

[Papadimitriou and Steiglitz, 1982] Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Robertson and Robertson, 2000] Robertson, S. and Robertson, J. (2000). Requirements Fundamentals: the Basis for Effective Testing.

[Robertson and Robertson, 2006] Robertson, S. and Robertson, J. C. (2006). *Mastering the Requirements Process*. Addison Wesley, 2nd edition.

[Ruhe and Greer, 2003] Ruhe, G. and Greer, D. (2003). Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE '03)*, pages 262–270, Rome, Italy. IEEE.

[Ruhe and Ngo-The, 2004] Ruhe, G. and Ngo-The, A. (2004). Hybrid Intelligence in Software Release Planning. *International Journal of Hybrid Intelligent Systems*, 1(1-2):99–110.

[Ruhe and Saliu, 2005] Ruhe, G. and Saliu, M. O. (2005). The Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53.

[Saaty, 1980] Saaty, T. L. (1980). *The Analytic Hierarchy Process, Planning, Priority Setting, Resource Allocation*. McGraw-Hill.

[Saliu and Ruhe, 2007] Saliu, M. O. and Ruhe, G. (2007). Bi-objective release planning for evolving software systems. In *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 105–114, New York, NY, USA. ACM.

[Sullivan, 1986] Sullivan, L. (1986). Quality Function Deployment. *Quality Progress*, June issue:39–50.

[van den Akker et al., 2004] van den Akker, M., Brinkkemper, S., Diepen, G., and Versendaal, J. (2004). Flexible Release Composition using Integer Linear Programming. Technical Report UU-CS-2004-063, Institute of Information and Computing Sciences, Utrecht University.

[van den Akker et al., 2005] van den Akker, M., Brinkkemper, S., Diepen, G., and Versendaal, J. (2005). Flexible Release Planning using Integer Linear Programming. In *Proceedings of the 11th International Workshop on Requirements Engineering for Software Quality (RefsQ '05)*, pages 247–262, Porto, Portugal. Essener Informatik Beitrage.

[van den Akker et al., 2008] van den Akker, M., Brinkkemper, S., Diepen, G., and Versendaal, J. (2008). Software Product Release Planning through Optimization and What-If Analysis. *Information and Software Technology*, 50(1-2):101–111.

[Walcott et al., 2006] Walcott, K. R., Soffa, M. L., Kapfhammer, G. M., and Roos, R. S. (2006). Timeaware test suite prioritization. In *ISSTA '06: Proceedings of the 2006 international symposium on Software testing and analysis*, pages 1–12, New York, NY, USA. ACM.

[Wiegers, 1999] Wiegers, K. E. (1999). First Things First: Prioritising Requirements. *Software Development*, 7(9):48–53.

[Xanthakis et al., 1992] Xanthakis, S., Ellis, C., Skourlas, C., Gall, A. L., Katsikas, S., and Karapoulios, K. (1992). Application of genetic algorithms to software testing. In *Proceedings of the 5th International Conference on Software Engineering and Applications*, pages 625–636, Toulouse, France.

[Yeh and Ng, 1990] Yeh, R. and Ng, P. (1990). Software Requirements – A Management Perspective. *System and Software Requirements Engineering*, pages 450–461.

[Yoo and Harman, 2007] Yoo, S. and Harman, M. (2007). Pareto efficient multi-objective test case selection. In *ISSTA '07: Proceedings of the 2007 international symposium on Software testing and analysis*, pages 140–150, New York, NY, USA. ACM.

[Zhang et al., 2008] Zhang, Y., Finkelstein, A., and Harman, M. (2008). Search based requirements optimisation: Existing work and challenges. In *REFSQ '08: Proceedings of the 14th international conference on Requirements Engineering*, pages 88–94, Berlin, Heidelberg. Springer-Verlag.

[Zhang et al., 2007] Zhang, Y., Harman, M., and Mansouri, S. A. (2007). The multi-objective next release problem. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1129–1137, New York, NY, USA. ACM.

[Zitzler and Thiele, 1999] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271.