

A Study of the Multi-Objective Next Release Problem

J. J. Durillo
University of Málaga (Spain)
durillo@lcc.uma.es

E. Alba
University of Málaga (Spain)
eat@lcc.uma.es

Y. Zhang
King's College London (UK)
yuanyuan.zhang@kcl.ac.uk

A. J. Nebro
University of Málaga (Spain)
antonio@lcc.uma.es

Abstract

One of the first issues which has to be taken into account by software companies is to determine what should be included in the next release of their products, in such a way that the highest possible number of customers get satisfied while this entails a 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 study the NRP problem from the multi-objective point of view, paying attention to the quality of the obtained solutions, the number of solutions, the range of solutions covered by these fronts, and the number of optimal solutions obtained. Also, we evaluate the performance of two state-of-the-art multi-objective metaheuristics for solving NRP: NSGA-II and MOCell. The obtained results show that MOCell outperforms NSGA-II in terms of the range of solutions covered, while this latter is able of obtaining better solutions than MOCell in large instances. Furthermore, we have observed that the optimal solutions found are composed of a high percentage of low-cost requirements and, also, the requirements that produce most satisfaction on the customers.

1 Introduction

Nowadays, many software companies are concerned with the development, maintenance, and enhancement of large and complex systems. Typically, customers are interested in buying and using these systems and each of these customers has its own necessities and/or preferences. Thus, one of the first issues which has to

be taken into account by software companies is to determine what should be included in the next release of their products, in such a way that the highest possible number of customers get satisfied while this entails a minimum cost for the company. This problem is known as the Next Release Problem (NRP) [1], and it is a widely known in Search Based Software Engineering [8] [9].

In engineering, a requirement is a singular documented need of what a particular product or service should be or do: it is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user. Thus, NRP consists in selecting a set among all the requirements of a software package such that the cost, in terms of money or resources, of fulfilling these requirements is minimum and the satisfaction of all the users of that system is maximum.

Since satisfying the highest number of customers and spending the lowest amount of money or resources are contradictory objectives, this problem can be treated as a multi-objective optimization problem (MOP) [17]. Contrary to single objective optimization, the solution of a MOP is not a single point, but a set of solutions known as the *Pareto optimal set*. This set is called *Pareto front* when it is plotted in the objective space. In these kinds of problems, there are two major issues to deal with. On the one hand, the solutions contained in the Pareto front should be as close as possible to the optimal Pareto front of the problem. In the context of NRP, this means to obtain solutions which produce the highest possible satisfaction of the customers as well as spending the lowest amount of money. On the other hand, the Pareto front should cover the maximum number of different situations, and provide a well distribution of solutions. In NRP, this means to provide the software engineer who takes the final decision

with a set of optimal solutions which contains a high number of different configurations.

NRP has been shown to be an instance of the *Knapsack* problem, which is \mathcal{NP} -hard [15]. As a consequence, it cannot be solved efficiently by using exact optimization techniques for large problem instances (e.g., an instance with 50 requirements has more than 10^{15} different configurations). In this situation, we need to make use of approximated techniques such as *metaheuristics* [6]. Although this kind of techniques does not ensure to find optimal solutions, they are able to obtain near-optimal solutions in a reasonable amount of time.

Metaheuristics are a family of approximate techniques which have received an increasing attention in the last years. Among them, evolutionary algorithms for solving MOPs are very popular in multi-objective optimization [2] [3], giving raise to a wide variety of algorithms, such as NSGA-II [4], and many others.

Despite that NRP can be considered as a MOP, as of today, most of the related works in the literature have considered it as a single-objective optimization problem [1] [7] [11]. One of the few proposals using a multi-objective formulation of the problem was proposed by Zhang *et al.* [17]. In this work, NSGA-II and other multi-objective algorithms (those using the Pareto dominance concept) were evaluated when solving NRP. This work showed that NSGA-II was able of comparable solutions with those obtained by the previous mono-objective approaches used in the literature, while in addition a set of non-dominated solutions were computed in a single run to help the decision maker at the company. However, this work did not pay special attention to the range of solutions covered by the resulting front, to the number of obtained solutions, nor to which algorithm obtained higher number of high quality solutions. The comparisons between the different evaluated algorithms were done in a visual way and no statistical analysis of the obtained results was provided.

In this work we fill the gap, and we advance in the results obtained in that last work: first, we want to prove that metaheuristics provide an intelligent guidance of the search. In order to achieve this goal, we evaluate NSGA-II and another multi-objective algorithm called MOCcell [12]. The latter algorithm has outperformed the former in terms of spread as well as convergence of solutions in several studies in continuous optimization [13] [14]. We take the same instances used in [17] and we compare the obtained results by both techniques with the ones obtained by a random search (RS) as a sanity check. All the comparisons are

given on the basis of two quality indicators, and the final number of obtained solutions. The chosen quality indicators are: Hypervolume [19] and Spread [3]. We apply a statistical methodology to ensure the significance of the results. Our second goal is to analyze the differences between the best solutions (those computed by the metaheuristics) and the rest of solutions.

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. The next sections is devoted to experimentation. We describe the obtained results in section 6, and we analyze them from the point of view of NRP in Section 7. Finally, Section 8 draws the main conclusions and lines of future work.

2 Multi-Objective Background

In this section we include some background on multi-objective optimization. We define the concepts of multi-objective optimization problem (MOP), Pareto optimality, Pareto dominance, Pareto optimal set, and Pareto front. In these definitions we are assuming, without loss of generality, the minimization of all the objectives. A general MOP can be formally defined as follows:

Definition 1 (MOP) Find a vector $\vec{x}^* = [x_1^*, x_2^*, \dots, x_n^*]$ which satisfies the m inequality constraints $g_i(\vec{x}) \geq 0, i = 1, 2, \dots, m$, the p equality constraints $h_i(\vec{x}) = 0, i = 1, 2, \dots, p$, and minimizes the vector function $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T$, where $\vec{x} = [x_1, x_2, \dots, x_n]^T$ is the vector of decision variables.

The set of all values satisfying the constraints defines the *feasible region* Ω and any point $\vec{x} \in \Omega$ is a *feasible solution*. As mentioned before, we seek for the *Pareto optima*. Its formal definition is provided below:

Definition 2 (Pareto Optimality) A point $\vec{x}^* \in \Omega$ is *Pareto Optimal* if for every $\vec{x} \in \Omega$ and $I = \{1, 2, \dots, k\}$ either $\forall_{i \in I} (f_i(\vec{x}) = f_i(\vec{x}^*))$ or there is at least one $i \in I$ such that $f_i(\vec{x}) > f_i(\vec{x}^*)$.

This definition states that \vec{x}^* is Pareto optimal if no other feasible vector \vec{x} exists which would improve some criteria without causing a simultaneous worsening in at least one other criterion. Other important definitions associated with Pareto optimality are the following ones:

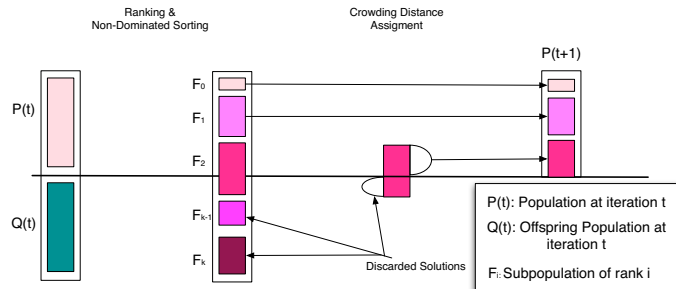


Figure 1. The NSGA-II procedure.

Definition 3 (Pareto Dominance) A vector $\vec{u} = (u_1, \dots, u_k)$ is said to dominate $\vec{v} = (v_1, \dots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if \vec{u} is partially less than \vec{v} , i.e., $\forall i \in \{1, \dots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \dots, k\} : u_i < v_i$.

Definition 4 (Pareto Optimal Set) For a given MOP $\vec{f}(\vec{x})$, the Pareto optimal set is defined as $\mathcal{P}^* = \{\vec{x} \in \Omega \mid \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \preceq \vec{f}(\vec{x})\}$.

Definition 5 (Pareto Front) For a given MOP $\vec{f}(\vec{x})$ and its Pareto optimal set \mathcal{P}^* , the Pareto front is defined as $\mathcal{PF}^* = \{\vec{f}(\vec{x}), \vec{x} \in \mathcal{P}^*\}$.

Obtaining the Pareto front of a MOP is the main goal of multi-objective optimization. In theory, a Pareto front could contain a large number of points (even infinite). In practice, a good 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, as well as they should be 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.

3 Problem Statement

This section is aimed at describing the mathematical model of NRP.

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 its degree of importance as a customer for the software development company.

There is also a set, R , of n requirements to complete. In order to meet each requirement it is needed

to expend a certain amount of resources, which can be transformed into an economical cost: the cost of satisfying the requirement. We denote 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 problem faced by the software company 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

$$\text{minimize } f_1 = \sum_{r_i \in R'} r_i \quad (1)$$

and,

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

Since minimizing a given function f is the same as maximizing $(-f)$, in this work we have considered the maximization of $(-f_1)$ (i.e., the economical cost for companies), and f_2 (i.e., the customer satisfaction).

4 Solver Algorithms

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

4.1 NSGA-II

NSGA-II, proposed by Deb *et al.* [4], is the reference algorithm in multi-objective optimization. It is based on a ranking procedure, consisting in extracting the non-dominated solutions from a population and assigning them a rank of 1. These solutions are removed from this population; the next group of non-dominated solutions have a rank of 2, and so on.

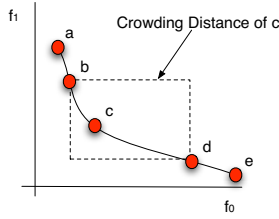


Figure 2. Measuring the crowding distance of a non-dominated point.

The algorithm has a current population that is used to create an auxiliary one (the offspring population); after that, both populations are combined to obtain the new current population. The procedure is as follows: the two populations are sorted according to their rank, and the best solutions are chosen to create the new population. In the case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance (see Figure 4) to the surrounding individuals belonging to the same rank is used to get the most promising solutions. Typically, both the current and the auxiliary populations have equal size. The procedure of NSGA-II is depicted in Figure 1.

4.2 MOCeII

MOCeII (Multi-Objective Cellular Genetic Algorithm), introduced by Nebro *et al.* [14], is a cellular genetic algorithm (cGA). In cGAs, the concept of (small) *neighborhood* is intensively used; this means that an individual may only cooperate with its nearby neighbors (see Figure 3) in the breeding loop. The overlapped small neighborhoods of cGAs help in exploring the search space because the induced slow diffusion of solutions through the population, providing a kind of exploration (diversification). Exploitation (intensification) takes place inside each neighborhood by genetic operations. MOCeII also includes an external archive to store the non-dominated solutions found so far. This archive is limited in size and uses the crowding distance of NSGA-II to keep diversity in the Pareto Front. We have used here an asynchronous version of MOCeII, called aMOCeII4 [13].

4.3 Random Search

We also apply a random search (RS) to NRP. This is merely a ‘sanity check’; all metaheuristic algorithms should be capable of comfortably outperforming random search for a well-formulated optimization prob-

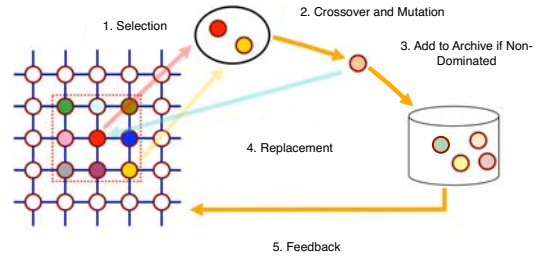


Figure 3. The MOCeII procedure.

lem. It consist in generating solutions randomly. The final results is the set of all the non-dominated solutions found.

5 Experimentation

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 as well as the genetic operators employed, the configuration of the algorithms, and the methodology we have followed.

5.1 Quality Indicators

For assessing the quality of the results obtained by the algorithms, two different issues are normally taken into account: (i) to minimize the distance of the Pareto front generated by the proposed algorithm to the optimal Pareto front (convergence), and (ii) to maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible (diversity). To determine these issues it is usually necessary to know the exact location of the optimal Pareto front. In the case of NRP, we are dealing with a problem whose optimal Pareto front is unknown. Thus, we have employed as Pareto optimal front the one composed of the non-dominated solutions out of all the executions carried out (i.e., the best front known for these problems until now).

A number of quality indicators have been proposed to be used for comparative studies among metaheuristics when solving MOPs. We use in this work one indicator which measures the diversity of the solution, and other one which measures both convergence and diversity.

- **Spread (Δ).** The *Spread* indicator [3] is a diversity quality indicator that measures the extent

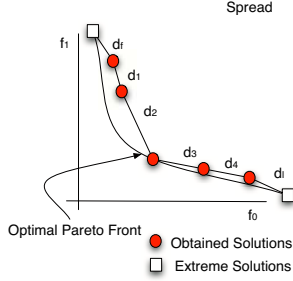


Figure 4. Distances from the extreme solutions.

of spread by the set of computed solutions. Δ is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}, \quad (3)$$

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 4 for further details). This metric takes a zero value for an ideal distribution, pointing out 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 desired.

- **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 5, $Q = \{A, B, C\}$) for problems where all objectives are to be minimized [19]. 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:

$$HV = \text{volume} \left(\bigcup_{i=1}^{|Q|} v_i \right). \quad (4)$$

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 former front dominate solutions in the second one, or, solutions in the first front are better distributed than in the second one. Thus, algorithms with larger values of HV are desirable.

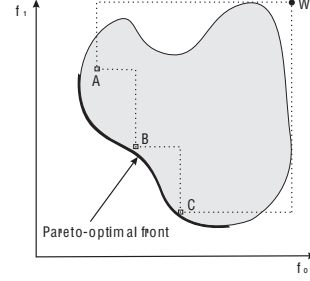


Figure 5. The hypervolume enclosed by the non-dominated solutions.

Another important topic when solving real problems is related to the number of obtained solutions. To this point, we also consider here the number of obtained solutions by the evaluated algorithms as well as the number of them which are Pareto optimal with regard to all the computed solutions.

5.2 Test Problems

The three algorithms were applied to two test problem sets, both of them composed of three problems, for two separate empirical study cases: Empirical Study 1 (ES1) and Empirical Study 2 (ES2). In the ES1 we report results concerning the performance of the three algorithms for what might be considered ‘typical’ cases, with the number of customers ranging from 15 to 100 and the number of requirements ranging from 40 to 140. In ES2, we are concerned with bounding NRP. This latter benchmark is composed of three instances: an instance with 20 requirements and 100 customers (few requirements, high number of customers), an instance with 25 requirements and 100 customers, and other with only 2 customers and 200 requirements (high number of requirements, few customers). All the instances have the nomenclature c_r , where c is the number of customers, and r the maximum number of requirements. We have not considered dependences among requirements [17].

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, previously chosen, is copied from that parent while the rest is copied from other parent. Finally, the mutation operator used is *random mutation* in which some random bits of the string are flipped.

5.4 Configuration

All approaches were run for a maximum of 10,000 function evaluations. The initial population was set to 100 in NSGA-II and MOCcell. In MOCcell, the archive size is 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 of requirements.

All the algorithms have been implemented using jMetal [5], a Java framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems.

5.5 Methodology

To assess the search capabilities of the algorithms, we have performed 100 independent runs of each experiment, and we show the mean, \bar{x} , and standard deviation, σ_n , as measures of location (or central tendency) and statistical dispersion, respectively. Since we are dealing with stochastic algorithms in order to provide confident results, we have also included a testing phase by performing a multiple comparison of samples [10]. We have used the Wilcoxon test for that purpose. We always consider a confidence level of 95% (i.e., significance level of 5% or p -value below 0.05) in the statistical tests.

Table 1. Results of the Δ quality Indicator

Problem	NSGA-II $\bar{x}\sigma_n$	MOCcell $\bar{x}\sigma_n$	RS $\bar{x}\sigma_n$
15_40	4.93e-1 \pm 4.2e-2	3.76e-1 \pm 3.3e-2	6.91e-1 \pm 3.8e-2
50_80	5.00e-1 \pm 3.5e-2	4.10e-1 \pm 3.9e-2	7.72e-1 \pm 2.9e-2
100_140	5.51e-1 \pm 3.7e-2	4.85e-1 \pm 3.8e-2	8.08e-1 \pm 2.4e-2
100_20	7.98e-1 \pm 9.1e-3	6.16e-1 \pm 4.9e-3	6.09e-1 \pm 5.3e-2
100_25	5.79e-1 \pm 2.4e-2	5.43e-1 \pm 2.6e-2	6.45e-1 \pm 4.2e-2
2_200	6.06e-1 \pm 3.6e-2	5.60e-1 \pm 4.7e-2	8.11e-1 \pm 3.2e-2

6 Obtained Results

In this section we present the obtained results by the three evaluated algorithms. We start by describing

Table 2. Results of the HV quality Indicator

Problem	NSGA-II $\bar{x}\sigma_n$	MOCcell $\bar{x}\sigma_n$	RS $\bar{x}\sigma_n$
15_40	6.63e-1 \pm 1.6e-3	6.63e-1 \pm 1.2e-3	5.27e-1 \pm 7.4e-3
50_80	5.88e-1 \pm 4.6e-3	5.85e-1 \pm 4.9e-3	4.35e-1 \pm 7.0e-3
100_140	5.28e-1 \pm 6.4e-3	5.20e-1 \pm 5.7e-3	3.65e-1 \pm 5.4e-3
100_20	6.13e-1 \pm 2.1e-4	6.13e-1 \pm 3.0e-4	5.72e-1 \pm 4.5e-3
100_25	6.35e-1 \pm 6.3e-4	6.35e-1 \pm 5.7e-4	5.45e-1 \pm 6.2e-3
2_200	5.26e-1 \pm 7.6e-3	5.17e-1 \pm 7.5e-3	3.02e-1 \pm 5.0e-3

the values obtained by the two quality indicators used. Then, we pay attention to the number of obtained solutions, and how many of these solutions are among the better solutions found so far.

The obtained values for Δ , HV, number of solutions, and number of Pareto optimal solutions among all the solutions found are shown in tables 1, 2, 4, and 5, respectively. In the case of the Δ indicator, the lower the value the better, while in the rest of indicators used, the higher the value the better.

We start by analyzing the Δ quality indicator (Table 1). If we focus on the instances belonging to the first problem set, ES1 (i.e., 15_40, 50_80, and 100_140), we see that MOCcell is the algorithm which has obtained the better values of the indicator. As to the ES2 benchmark (i.e., 100_20, 100_25, and 2_200), the results observed in ES1 hold the same for the instances with more than 20 requirements. As a general conclusion from the obtained values by this indicator, MOCcell has performed better than the other techniques.

In most of the instances the results obtained by the RS are far from those obtained by MOCcell and NSGA-II. However, it is remarkable the obtained results by RS in the instance 100_20 (the one with the smallest number of requirements).

Proceeding as before, we analyze the results obtained for the HV quality indicator, whose values are included in Table 2. Considering the instances belonging to ES1, the obtained results by MOCcell and NSGA-II has been quite similar in the instances with 40 and 80 requirements. However, in the instance with the highest number of requirements, NSGA-II has outperformed MOCcell. Concerning ES2, MOCcell and NSGA-

Table 3. Comparison NSGA-II vs MOCcell: Δ and HV Indicators

Problem	NSGA-II vs MOCcell	
	Δ Indicator	HV Indicator
15_40	MOCcell	•
50_80	MOCcell	NSGA-II
100_140	MOCcell	NSGA-II
100_20	MOCcell	•
100_25	MOCcell	•
2_200	MOCcell	NSGA-II

II have obtained similar results in the instances with 20 and 25 requirements, whereas in the instance with the highest number of requirements (200) NSGA-II has performed better than MOCcell.

For this indicator, all the results obtained by RS has been significantly worse than the obtained by MOCcell and NSGA-II.

In Table 3, we summarize the comparison between MOCcell and NSGA-II, considering Δ and HV. The second and third column show the name of the algorithm which has been significantly better in each indicator, respectively; a “•” symbol means that no statistical differences have been found between them. As general conclusion for the Δ indicator, we have that MOCcell has outperformed the results obtained by NSGA-II in all the instances. As to the HV indicator, both algorithms, MOCcell and NSGA-II, have behaved in a similar way in the instances with the smaller number of requirements (until 40 requirements); however, as the number of requirements increases NSGA-II has obtained better figures than MOCcell.

Regarding to the number of obtained solutions (Table 4), we observe that NSGA-II, and MOCcell have performed similarly: only in the instance with the smallest number of requirements of the ES2 benchmark, NSGA-II has obtained more solutions than MOCcell clearly. As to how many of the obtained solutions are Pareto optimal among all those obtained (Table 5), NSGA-II has outperformed MOCcell in all the cases. Thus, focusing in NSGA-II, we observe that in the instances with 20, 25, and 40 requirements, it has obtained from 42 to 20 Pareto optimal solutions, respectively. In the instances with 80 and 140 requirements, it has only found less than five Pareto optimal solutions, and, in the instance with the 200 requirements and only 2 customers, it has obtained around 50 Pareto optimal solutions per execution.

Finally, we see that RS has obtained a fewer number of solutions than NSGA-II and MOCcell in practically all the instances. Nevertheless, it is remarkable that RS has found some optimal solutions in the instance with the highest number of requirements.

In Table 6 we summarize the comparison between NSGA-II and MOCcell according to the number of found solutions. As commented above, the algorithm which obtains a higher number of solutions and Pareto optimal solutions is NSGA-II. In this case, all the results but two are given with statical confidence.

An example of the obtained fronts in the different instances is depicted in Figure 6. In this figure “Customer Satisfaction” means the total satisfaction of the

Table 4. Results of the Number of Obtained Solutions

Problem	NSGA-II \bar{x}_{σ_n}	MOCcell \bar{x}_{σ_n}	RS \bar{x}_{σ_n}
15_40	1.00e+2±0.0e+0	9.99e+1±3.4e-1	3.34e+1±3.7e+0
50_80	1.00e+2±0.0e+0	1.00e+2±1.4e-1	4.08e+1±4.5e+0
100_140	1.00e+2±0.0e+0	9.99e+1±3.4e-1	4.12e+1±5.5e+0
100_20	1.00e+2±0.0e+0	7.71e+1±1.0e+0	3.78e+1±4.2e+0
100_25	1.00e+2±0.0e+0	1.00e+2±2.2e-1	3.61e+1±3.9e+0
2_200	1.00e+2±0.0e+0	9.93e+1±1.8e+0	2.89e-1±4.2e+0

Table 5. Results of the Number of Solutions Contained in the Best Front Found

Problem	NSGA-II \bar{x}_{σ_n}	MOCcell \bar{x}_{σ_n}	RS \bar{x}_{σ_n}
15_40	1.98e+1±4.4e+0	1.39e+1±4.5e+0	3.40e-1±8.1e-1
50_80	2.11e+0±2.9e+0	8.00e-2±3.1e-1	0.0e+0±0.0e+0
100_140	1.19e+0±2.5e+0	2.00e-2±1.4e-1	0.0e+0±0.0e+0
100_20	4.23e+1±1.6e+0	3.03e+1±9.6e-1	9.20e-1±1.1e+0
100_25	3.77e+1±3.7e+0	3.32e+1±4.0e+0	3.00e-2±1.7e-1
2_200	4.90e+1±2.8e+0	0.0e+0±0.0e+0	1.30e+1±5.4e-1

customers, and “-cost” represents the economical cost for companies. In all the cases the metaheuristic techniques have outperformed RS. In the instance with the smaller number of requirements (Figure 6.b), we see that the results of RS are closer to the results of the other techniques than in the other instances. It is also worth mentioning how MOCcell is able to obtain a front which covers a high range of different solutions; however, NSGA-II have obtained solutions which dominate those obtained by the other techniques, specially in the instances with the higher number of requirements.

The advantages of using metaheuristic techniques can be easily drawn from Figure 6. Comparing the obtained results by these techniques against RS, we observe the benefits of using them for a software company. On the one hand, the cost is reduced; NSGA-II as well as MOCcell have obtained solutions with less economical cost and also higher satisfaction of their customers. On the other hand, multi-objective metaheuristics provide the software engineer with a set of

Table 6. Comparison NSGA-II vs MOCcell: Number of Solutions and Pareto Optimal Solutions

Problem	NSGA-II vs MOCcell	
	Solutions	Pareto Optimal Solutions
15_40	NSGA-II	NSGA-II
50_80	•	NSGA-II
100_140	NSGA-II	NSGA-II
100_20	NSGA-II	NSGA-II
100_25	•	NSGA-II
2_200	NSGA-II	NSGA-II

trade off optimal configurations instead a single one allowing him (or her) to choose the most appropriated solution in each different situation.

7. Analysis

Previous section has shown that multi-objective optimization algorithms are suitable techniques for solving NRP; in this section, we are interested in analyzing the obtained solutions from the point of view of the problem.

Let us suppose that a software engineer has to select a subset of requirements to including in the next release of a software package and he only wants to optimize the cost of fulfilling these requirements; in this situation, it is clear that he should include, in this set, those requirements which are cheaper of implementing than others. Something similar happens if he only wants to maximize the satisfaction of the users of that system: he should consider those requirements which satisfy the customer the most. But, what requirements should he consider if he wants to optimize both objectives?

We have analyzed which requirements are included in each solution. Thus, for each point in a front of solutions, we have computed the percentage of included requirements which are cheaper or equal expensive of implementing than all the requirements not included by this point. Similarly, we have also calculated the percentage of requirements included which satisfy the customers more or equal than all the requirements not included by this point. Let us call these percentages as P_1 and P_2 , respectively. They can be formalized as

$$P_1 = \frac{\|r_i \in R', r_i \leq r_j, \forall r_j \in (R - R')\|}{\|R'\|} \quad (5)$$

and,

$$P_2 = \frac{\|r_i \in R', \sum_{j=1}^n c_j * v_{ji} \geq \sum_{j=1}^n c_j * v_{jk}, \forall r_k \in (R - R')\|}{\|R'\|} \quad (6)$$

The mean and standard deviation of P_1 and P_2 in all the solutions in all the executions carried out are included in Tables 7 and 8, respectively. Regarding to Table 7, we observe that the computed solutions by the metaheuristic techniques, NSGA-II and MOCell, are composed of a higher percentage of cheapest requirements than the solutions computed by RS. For example, in the instance 15_40 the percentage of cheapest requirements in the solutions found by NSGA-II and MOCell are 78.9% and 77.2%, respectively, while in the solutions found by RS this percentage has only been of 23.4%. The same holds with the percentage

of requirements which most satisfy the customers (Table 8): this percentage is higher in the solutions found by the metaheuristic techniques than in the solutions found by RS. Thus, the better solutions are composed of a high number of low cost requirements, and also those requirements which most satisfy the customers.

Table 7. Percentage of Included Requirements with Cost Fewer of Equal that all the Requirements not Included

Problem	NSGA-II \bar{x}_{σ_n}	MOCcell \bar{x}_{σ_n}	RS \bar{x}_{σ_n}
15_40	7.89e+1±1.8e+0	7.72e+1±2.1e+0	2.34e+1±1.8e+0
50_80	7.22e+1±3.9e+0	6.55e+1±5.3e+0	1.18e+1±1.5e+0
100_140	5.48e+1±6.1e+0	4.45e+1±6.5e+0	7.85e+0±7.6e-1
100_20	8.18e+1±6.0e-1	8.07e+1±5.0e-1	5.35e+1±4.1e+0
100_25	8.15e+1±1.3e+0	8.10e+1±1.4e+0	3.40e+1±2.9e+0
2_200	2.04e+1±4.3e+0	1.87e+1±3.4e+0	1.12e+1±3.7e-1

Table 8. Percentage of Included Requirements with Customer Satisfaction Bigger of Equal that all the Requirements not Included

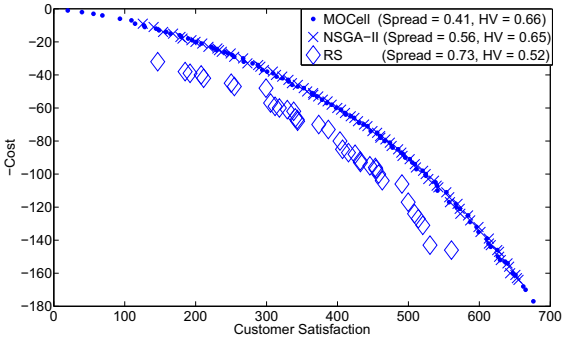
Problem	NSGA-II \bar{x}_{σ_n}	MOCcell \bar{x}_{σ_n}	RS \bar{x}_{σ_n}
15_40	4.80e+1±2.6e+0	4.50e+1±2.6e+0	1.27e+1±2.0e+0
50_80	1.54e+1±3.1e+0	1.39e+1±3.1e+0	4.51e+0±6.5e-1
100_140	3.42e+0±1.4e+0	3.42e+0±1.9e+0	1.44e+0±3.0e-1
100_20	4.85e+1±8.4e-1	4.79e+1±6.1e-1	2.99e+1±2.8e+0
100_25	3.41e+1±1.9e+0	3.28e+1±1.7e+0	1.64e+1±1.9e+0
2_200	1.19e+1±3.4e+0	1.05e+1±3.3e+0	2.49e+0±4.3e-1

8 Conclusions and Future Work

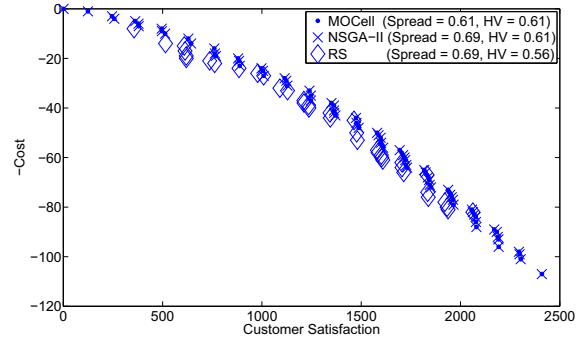
In this paper we have studied the Next Release Problem, with the intention of analyzing the performance of different multi-objective algorithms, and the solutions they have provide. We have evaluated two state-of-the-art multi-objective optimization algorithms and compared them against a random search. This comparison has been done in the basis of two quality indicators, HV and Δ , and the number of solutions obtained by those algorithms.

In the case of the Δ indicator, which measures how uniform is the distribution of computed solutions, MOCell is the algorithm which has worked out the better results in instances with more than 20 requirements. The random search algorithm has obtained the worst results in practically all the instances.

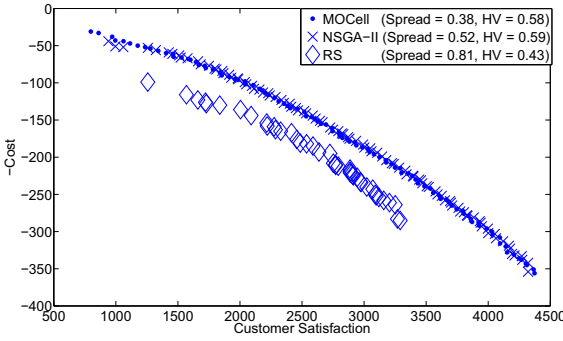
Regarding to the HV indicator, which measures the convergence and distribution of the fronts obtained by the algorithms, NSGA-II and MOCell have both shown a similar performance in instances with a small and



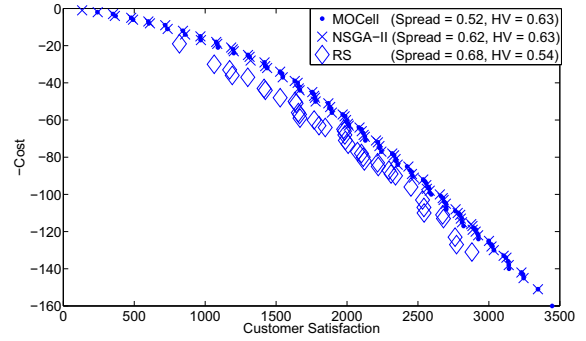
(a) 15 customers; 40 requirements



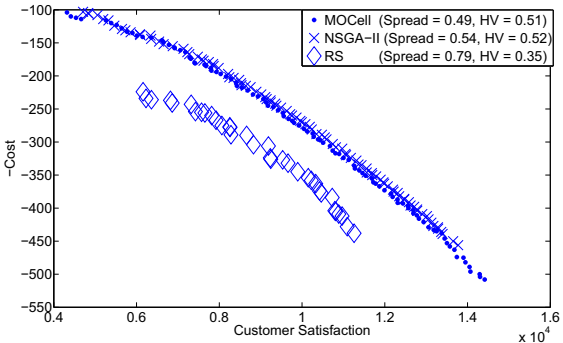
(d) 100 customers; 20 requirements



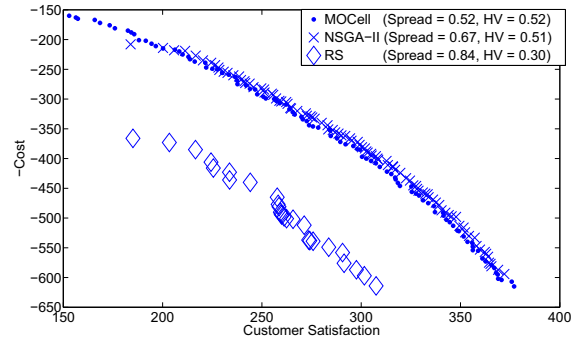
(b) 50 customers; 80 requirements



(e) 100 customers; 25 requirements



(c) 100 customers; 140 requirements



(f) 2 customers; 200 requirements

Figure 6. Examples of the Obtained Front of Solutions in each Instance.

medium number of requirements, i.e., from 20 to 40 requirements. As the number of requirements increases, NSGA-II provides better results than MOCeII. The random search has obtained the worst results in all the instances.

As to the number of obtained solutions, NSGA-II is also the algorithm which has shown the best performance, and it is also the technique which computes the highest number of solutions which are contained in the best front known for each instance.

Concerning the solutions of the problem, we have

observe that the best solutions are composed of a high percentage of low-cost requirements, and also, of requirements which most satisfy the customers.

The future work will verify these findings by applying search techniques to real world problems. This will provide valuable feedback to researchers and practitioners in search techniques and in software engineering communities. Other reformulations of the problem considering different sets of objectives and constraints including dependency relationship between requirements will be experimented with. This in turn may give rise

to the need for the development of more efficient solution techniques. It is also interesting to investigate how these techniques scale when the number of requirements and/or customer increases. In order to come to this goal, it is also needed to develop a procedure which allows the systematic creation of instances with the desired features; in this sense, we plan to design a problem generator for NRP instances.

Acknowledgements

J. J. Durillo, A. Nebro, and E. Alba acknowledge funds 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.

References

- [1] BAGNALL, A., RAYWARD-SMITH, V., AND WHITTLEY, I. The next release problem. *Information and Software Technology* 43, 14 (Dec. 2001), 883–890.
- [2] COELLO COELLO, C. A., VAN VELDHIJZEN, D. A., AND LAMONT, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.
- [3] DEB, K. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.
- [4] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE TEC* 6, 2 (Apr. 2002), 182–197.
- [5] DURILLO, J.J., NEBRO, A.J., LUNA, F., DORRONSORO, B., AND ALBA, E. jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Dept. de Lenguajes y Ciencias de la Computación, University of Málaga.
- [6] GLOVER, F. W., AND KOCHENBERGER, G. A. *Handbook of Metaheuristics*. Kluwer, 2003.
- [7] GREER, D., AND RUHE, G. Software release planning: an evolutionary and iterative approach. *Information & Software Technology* 46, 4 (2004), 243–253.
- [8] HARMAN, M. The current state and future of Search Based Software Engineering. In FOSE 2007.
- [9] HARMAN, M., AND JONES, F. B. Search Based Software Engineering. *Information and Software Technology* 43, 14 (2001), 833–839
- [10] HOCHBERG, Y., AND TAMHANE, A. C. *Multiple Comparison Procedures*. Wiley, 1987.
- [11] KARLSSON, J., WOHLIN, C., AND REGNELL, B. An evaluation of methods for prioritizing software requirements. *Information and Software Technology* 39, (1998), 939–947.
- [12] NEBRO, A. J., DURILLO, J. J., LUNA, F., DORRONSORO, B., AND ALBA, E. A Cellular Genetic Algorithm for Multiobjective Optimization. Proceedings of NCSO 2006.
- [13] NEBRO, A. J., DURILLO, J. J., LUNA F, DORRONSORO, B, AND ALBA, E. Design issues in a multi-objective cellular genetic algorithm. EMO 2007.
- [14] NEBRO, A. J., DURILLO, J. J., LUNA, F., AND DORRONSORO, B., AND ALBA, E. MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. *International Journal of Intelligent Systems* (to appear), 2008.
- [15] PAPADIMITRIOU, C. H., AND STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.
- [16] SZIDAROVSKY, F., GERSHON, M. E., AND DUKSTEIN, L. *Techniques for multi-objective decision making in systems management*. Elsevier, New York, 1986.
- [17] ZHANG, Y., HARMAN, M., AND MANSOURI, A. S. The Multi-Objective Next Release Problem. In GECCO 2007.
- [18] ZITZLER, E., LAUMANN, M., THIELE, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.
- [19] ZITZLER, E., THIELE, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE TEC* 3(4):257–271.