# Requirements Selection: Knowledge based optimization techniques for solving the Next Release Problem

José del Sagrado, Isabel M. del Águila, Francisco J. Orellana, and S. Túnez

Dpt. Languages and Computation,
Ctra Sacramento s/n, 04120 University of Almería, Spain
{jsagrado,imaguila,fjorella,stunez}@ual.es

**Abstract.** The requirements selection for the next software release is a problem always present in Software Engineering. The complex nature of this problem and the difficulty to address it using exact techniques has motivated the application of optimization techniques to obtain near optimal solutions. This work provides a review of the different optimization techniques proposed to accomplish the requirements selection problem. Moreover, it proposes the application of these techniques in a requirement tool in order to be used in real software developments.

**Keywords:** next release problem, optimization techniques, search based software engineering

## 1 Introduction

Software development organizations fail many times to deliverer its products within schedule and budget. Statistical studies, and all the Chaos Reports [16] published since 1994, reveal that, frequently, tasks related to requirements lead software project to the disaster. When requirement-related tasks are poorly defined or executed, the software product is typically unsatisfactory [23]. Software requirements express the needs and constraints fixed for a software product that contribute to the solution of some real world problem [18]. Usually stakeholders propose some desired functionalities that software managers must filter in order to define the set of requirements to include in the final software product. All new suggested functionalities cannot be selected to be implemented since resource constraints are always present in development companies; hence each new feature competes against each other to be included in the next release software product. The requirements selection is considered a complex task in every software development since many factors are involved in this decision. A bad or inappropriate choice of enhancements can turn into a source of problems during software development: scheduling problems, dissatisfied customers, and economic losses. This problem is known as next release problem (NRP) [2] and it is considered an optimization problem [2, 17, 12] within the Search Based Software Engineering (SBSE) discipline [13, 5, 14]. The SBSE area is a growing research field which proposes the application of search based optimization algorithms to tackle problems in Software Engineering (SE). The term SBSE was first used in 2001 by Harman and Jones [13] and has been successfully applied to different problems in SE such as requirements, design tools and techniques, software verification and testing and debugging among others [15]. Different approaches can be found in the literature to tackle with requirements selection problem, for example, [2] and [3] apply greedy and simulated annealing (SA) techniques, [12] use genetic algorithms (GAs) in software release planning and [21] propose the use of ant colony optimization (ACO). In this work we provide a comprehensive review of the AI techniques applied to solve the NRP. We also propose the inclusion of these techniques on a CARE (Case Aided REquirement) tool to guide the decision maker to select the best set of requirements for the next release. The rest of this paper is structured as follows. Section 2 introduces and provides the formal description of the NRP problem describing different approaches used when addressing the NRP as an optimization problem. Section 3 analyzes the existing techniques applied in the literature to address the NRP whereas Section 4 describes a proposal to integrate these optimization techniques in a CARE tool. Finally, Section 5 draws conclusions and future works.

## 2 The NRP Problem

The problem of selecting the subset of requirements among a whole set of candidate requirements proposed by a group of customers, that will be included in the next release of a software product is a well-known problem in Software Engineering. However, it is not a straightforward problem since many factors are involved in this selection problem. Customers, seeking their own interest, demand the set of enhancements they consider important, but not all customer needs can be satisfied; on the one hand, each requirement means a cost in effort terms that the company must assume but company resources are limited; on the other hand, neither all the customers are equally important for the company nor are the requirements equally important for the customers. Market factors can also drive this selection process; the company may be interested on satisfying the newest customers' needs, or they may consider desirable to guarantee every customer have fulfilled at least one of their proposed requirements. Two main goals are usually considered in this kind of problems: find a subset of requirements which maximize the customers' satisfaction and minimize the required effort to implement the subset of chosen requirements. The complexity of the problem increases as the number of customers and requirements grows. Therefore, optimization techniques can be used to find optimal or near optimal solutions in a reasonable amount of time. As Harman defined in [13], it is possible to apply metaheuristic search to numerous problems in SE, but that aim requires a reformulation of the problem which implies to define:

– a representation of the problem which is amenable to symbolic manipulation,
– a fitness function based on this representation and
– a set of manipulation operators.

These are the steps that we are going to follow in order to review how meta-heuristic search techniques had been applied to the NRP problem.

## 2.1 The NRP formulation

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of requirements that are proposed by the customers. These requirements represent enhancements to the current software system, suggested by a set of $m$ customers and candidates to be included in the next release. Customers are not equally important for the company. So, each customer $i$ will have an associated weight $w_i$, which measures its relative importance. Let $W = \{w_1, w_2, \ldots, w_m\}$ be the set of customers' weights. Each requirement $r_j \in R$ has an associated development cost $e_j$, which represents the effort needed in its development. Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of requirements' efforts. On many occasions, the same requirement is suggested by several customers. However, its importance or priority may be different for each customer. Thus, the importance that a requirement $r_j$ has for customer $i$ is given by a value $v_{ij}$. The higher the $v_{ij}$ value, the higher is the priority of the requirement $r_j$ for customer $i$. A zero value for $v_{ij}$ represents that customer $i$ has not suggested requirement $r_j$. All these importance values $v_{ij}$ can be arranged under the form of an $m \times n$ matrix. The global satisfaction, $s_j$, or the added value given by the inclusion of a requirement $r_j$ in the next release, is measured as a weighted sum of the its importance values for all the customers and can be formalized as: $s_j = \sum_i^m w_i v_{ij}$. In every SE project it is common to find dependencies among the features suggested by the customers. Requirements dependencies mean that a set of constraints has to be considered during the requirement selection task, since they force us to check whether conflicts are present whenever we intend to select a new requirement to be included in the next software release. Several kinds of dependencies related to this problem are proposed first in [1] and later in [4]:

– *Implication or precedence.* A requirement $r_i$ cannot be selected if a requirement $r_j$ has not been implemented yet.
– *Combination or coupling.* A requirement $r_i$ cannot be included separately from a requirement $r_j$.
– *Exclusion.* A requirement $r_i$ can not be included together with a requirement $r_j$.
– *Revenue-based.* The development of a requirement $r_i$ implies that some others requirements will increase their value.
– *Cost-based.* The development of a requirement $r_i$ implies that some others requirements will increase their implementation cost.

These kind of dependences, that are reviewed in [20], are taken into account in some works about NRP such as [2] and [12]. Thus, the NRP main goal is to search for a subset of requirements $\hat{R}$ within the set of all subsets of $n$ requirements $P(R)$, so the dimension of the search space is $2^n$. A subset of requirements $\hat{R}$ can be represented in this space as a vector $x_1, x_2, \ldots, x_n$, where $x_i \in 0, 1$. If requirement $r_i \in \hat{R}$, then $x_i = 1$ and otherwise $x_i = 0$. In this way, the NRP can be considered as an instance of the 0-1 knapsack problem, and in consequence is a NP-hard problem [2] (it is unfeasible to tackle it using exact techniques to find the best solution in a polynomial time).

## 2.2 Single-objective NRP or Multi-objective NRP

The main goal of optimization problems is to search for the best solution with respect to several objectives. The quality of a candidate solution with respect to each objective is measured throughout the use a previously fixed evaluation function. According to the number of objectives, the problem can be classified as single-objective or multi-objective. Generally, in order to define the next software release, the main goal that we pursuit is to select a subset of requirements $\hat{R}$ from the candidate requirement list $R$, which maximize satisfaction and minimize development effort. The satisfaction and development effort of this subset $\hat{R}$ can be obtained, respectively, as

$$\mathrm{sat}(\hat{R}) = \sum_{j \in \hat{R}} (s_j), \quad \mathrm{eff}(\hat{R}) = \sum_{j \in \hat{R}} (e_j) \tag{1}$$

where $j$ is an abbreviation for requirement $r_j$. As the resources available are limited, then development effort cannot exceed a certain bound $B$. First works [2, 12] in NRP tended to consider this problem as a single-objective problem: maximize customers' satisfaction within a certain development constraints. Their main goal is to find a subset of requirements that satisfies customer requests within a given resource constraints (i.e. availability of resources). That is to say, the selected subset of requirements $\hat{R}$ has to maximize customers' satisfaction within a given development effort bound $B$. Formally,

$$\begin{array}{l} \text{maximize } \mathrm{sat}(\hat{R}) \\ \text{subject to } \mathrm{eff}(\hat{R}) \leq B \end{array} \tag{2}$$

Most recent works [25, 9] consider NRP as a multi-objective problem (MONRP), since they consider at least two conflicting objectives; maximize customers' satisfaction and minimize the total effort involved in the development of the selected requirements. Formally, the NRP can be defined as the search for requirement subsets $\hat{R} \subseteq R$ such as

$$\begin{array}{l} \text{maximize } \mathrm{sat}(\hat{R}) \\ \text{minimize } \mathrm{eff}(\hat{R}) \end{array} \tag{3}$$

Other approaches (see [10]) formulate multi-objective in a different way, applying other criterion to measure customers' satisfaction or defining more than two objectives. In contrast to single objective optimization, which returns a unique solution, multi-objective optimization returns a set of solutions satisfying the proposed objectives. This means an advantage for the software developers as they can choose from a range of different alternatives. The set of non-dominated solutions that fulfill multiple objectives is denominated Pareto optimal front (see Fig. 1) . Whether any of the objectives of these solutions is improved, the others objectives will get worse.

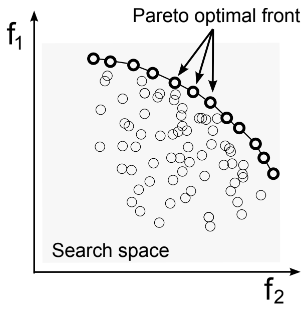

**Fig. 1.** Pareto optimal front considering two different objectives $f_1$ and $f_2$.

## 3 Analysis of Techniques

Once the problem has been formulated as a search problem and the fitness functions have been defined, metaheuristic techniques are be applied in order to find possible solutions. In the specific case of NRP, the metaheuristic techniques that can be found in the literature are: greedy algorithms, simulated annealing, genetic algorithms or ant colony systems. However, although all these approaches pursuit the same aim, not all of them deal with the NRP in the same way.

### 3.1 Simulated Annealing

Simulated annealing (SA) is an optimization algorithm which emulates the energy changes that occur in a system of particles when its temperature is reduced till the system reaches a state of equilibrium. At higher temperatures drastic changes in the system are allowed, whereas at lower temperatures only minor changes are allowed. This cooling scheduling has as goal to reduce the energy state of the system, taking the system from an arbitrary initial energy state to a final state with the minimum possible energy. Starting from an initial solution and an initial temperature $T_0$, the algorithm iterates following a cooling schedule function which decreases the temperature until it reaches a minimum $T_{end}$. Using some cooling functions, the algorithm stays at the same temperature for a certain number of iterations; then, it is decreased. In each algorithm iteration, a new solution from the neighbourhood is extracted and it can be accepted or not as the current solution. This technique allows to explore the search space at higher temperatures accepting poor solutions, whereas at lower temperatures only moves that improve the current solution are accepted. This algorithm uses an acceptance probability which determines whether a new solution found is accepted as the current one or not. Formally, let $S$ be the current solution and $S'$ be a new solution in the neighborhood of $S$, $S' \in nei(S)$ (it is said that $S'$ is a neighbour of $S$, if they differ exactly on one requirement). Let $T$ be the current temperature and $\Delta E = f(S') - f(S)$ the energy difference between $S$ and $S'$, obtained after applying a fitness function. The probability of making the transition from the current solution $S$ to the candidate solution $S'$, i.e. the acceptance probability, is denoted by

$$p(S, S', T) = \begin{cases} 1, & \text{if } \Delta E > 0 \\ e^{\frac{\Delta E}{T}}, & \text{otherwise.} \end{cases} \tag{4}$$

SA has been applied to NRP by Bagnall et al. [2] and Baker et al. [3]. In contrast to most of the NRP approaches in the literature, which are focused on finding the optimal subset of requirements, the main aim searched in [2] is to find a subset of customers whose needs will be fully covered. Only implication dependencies are considered (i.e. a requirement that needs some others requirements to be implemented), defining precedence relationships for the candidate requirements. Baker et al. [3] focuses on a component selection problem of a software system from a telecommunications company. The aim of this work is to compare the component selection obtained from a group of human experts with the results obtained applying a search technique such as simulated annealing. In this case dependencies among requirements are not considered.

**Table 1.** Simulated annealing techniques applied to the NRP

| | Fitness Function | Cooling Schedule | Initial Temperature | Parameters |
|---|---|---|---|---|
| [2] | $f(S) = \sum_{i \in S} w_i -$ $-\lambda \min\{0, B - \text{eff}(S)\}$ | Geometric $T_{i+1} = \alpha T_0$ | $T_0 = 100$ | $\alpha = 0.9995$ Iters. $= \{250, 500\}$ |
| | | Lundy and Mees $T_{i+1} = T_i/(1+\beta)T_i$ | $T_0 = 100$ | $\beta = \{5 \times 10^{-7}, 10^{-7}, 10^{-8}\}$ |
| [3] | $f(S) = \text{sat}(S)$ | Geometric $T_{i+1} = (1-\alpha)T_0$ | $-T_0 = \frac{\Delta E^{max}}{ln(1-p_1)}$ | $\alpha = 0.2$ Iters. $= 15000$ $p_1 = 0.8$ |

Table 1 provides details about the fitness and cooling schedule functions applied in both works, and the parameters used for the experiments. Bagnall's approach [2] combines into a single fitness function two objectives based on the customers' weights and the total effort of the solution, since the requirement priorities are not gathered in this work. By contrast, Baker et al. [3] only takes into account the total satisfaction given by a solution to measure its quality. Applying the specified parameters, the results obtained by Bagnall et al. [2] show that a search technique such as SA is the best choice among the studied alternatives (greedy algorithms or hill climbers). On the other side experiments performed in [2] using the Lundy and Mees cooling schedule slightly outperforms the geometric function approach. Bakeret al. [3] compares SA to a greedy algorithm, and a selection performed by human experts. Best results are also reached by SA. This technique yields the best score in every experiment, followed by the greedy algorithm. Finally, the component selection specified by the human experts demonstrates to be much worse than the returned using SA.

## 3.2 Genetic Algorithms

A Genetic Algorithm (GA) [11] is a bio-inspired search algorithm based on the evolution of collections of individuals (i.e. populations) as result of natural selection and natural genetics. Starting from an initial population, their individuals evolve into a new generation by means of selection, crossover and mutation operators. This technique emulates the evolution process where best fitted individuals survive through generations. This evolution (i.e. iteration of the algorithm) is performed selecting some individuals according to their quality (measured by a fitness function) from the population. Then some parents are chosen and combined using crossover to produce new individuals (children). Finally, all the individuals in the new population have a certain but very small probability of mutation, i.e. their hereditary structure may be altered. The crossover and mutation operators are in charge of producing new individuals and they are applied with different probabilities i.e. crossover probability and mutation probability, denoted by $P_c$ and $P_m$, respectively.

NRP addressed using GAs can be found in Greer and Ruhe [12], as a single-objective problem, whereas Zhang et al. [25] , Durillo et al. [9] and Finkelstein et al. [10] tackle the problem using a multi-objective approach, existing important differences among them.

Greer and Ruhe [12] addresses the requirement selection problem from a perspective based on agile methods, considering the iterations in the incremental software development. This work proposes an overall method for optimally allocating requirements to increments, which deals with a single-objective NRP as a combination of two different objectives: maximize the satisfaction and minimize the total cost of the solution. Precedence (implication) and coupling (combination) dependencies are considered and added to the problem as new constraints. The system provides the decision maker a small set of the most promising solutions that can be selected for the next software increment.

Zhang et al. [25] applies GAs to solve the NRP, using first synthetic data in [25] and real data in [24]. As Greer and Ruhe [12], two main goals related to benefit and effort are considered, although in this case the problem is addressed from a multi-objective perspective applying NSGA-II and ParetoGA algorithms. The first is a well-known multi-objective algorithm using an elitist strategy to preserve the solutions from the best front whereas the latter is an extension of the simple GA. Results reported by this work point to the NSGA-II method as the best choice; the solutions belonging to the Pareto front are better than the rest of methods evaluated and it offers a better diversity of solution distribution.

Durillo et al. [9] filled the gap left by this last work, arguing that the algorithms evaluation was performed in a visual way and no statistical analysis of the obtained results was provided. Using the same instances used by [25], they solve NRP by using a Random Search, and two multi-objective metaheuristics, NSGA-II and MOCell. In order to perform the analysis of the results, some quality indicators were used to measure the extent of spread of the set of solutions (i.e. spread) or the volume covered by the set of non-dominated solutions (i.e. hypervolume). According to the obtained results, Random Search results are generally poor, whereas NSGA-II and MOCell obtains good results presenting a similar performance in most of the cases. However NSGA-II outperforms MOCell when the experiment reaches the highest number of requirements.

Finkelstein et al. [10] focuses on satisfying the fairness term related to the requirements selection problem, whose main motivation is to "try to balance the requirement fulfillments between the customers". However, the task of finding this fairness does not result easy to achieve; hence three different multi-objective approaches are proposed. These proposals intend to maximize the satisfaction taking into account the number of fulfilled requirements per customer, the total satisfaction, or the percentage of satisfied requirement per customer. Two different algorithms, NSGA-II and the two-archive algorithm, are studied and applied on a set of real data from a telecommunication company.

Table 2 summarizes the techniques applied in each work and the parameters settings used by authors in the experimental evaluation.

**Table 2.** Genetic algorithms applied to the NRP

| | Techniques | Selection | Crossover | Mutation |
|---|---|---|---|---|
| [12] | Single-objective GA | Probability curve based on fitness value | Random selection $P_c = \{0.1, 0.2, 0.3 \cdots, 1\}$ | Random $P_m = \{0.05, 0.1, 0.15, \cdots, 1\}$ |
| [25] | NSGA-II, Pareto GA, Single-objective GA | Tournament | Single Point $P_c = 0.8$ | Bitwise $P_m = 1/n$ |
| [10] | NSGA-II, The Two Archive | Tournament | Single Point $P_c = 0.8$ | Bitwise $P_m = 1/n$ |
| [9] | NSGA-II, MOcell | Tournament | Single Point $P_c = 0.9$ | Random $P_m = 1/n$ |

## 3.3 Ant Colony Optimization

Ant colony optimization (ACO) is a meta-heuristic for combinatorial optimization problems proposed by Dorigo et al. [7], [6]. This technique emulates the cooperative behaviour of real ants in their task to find the shortest path from their colony to a source of food. This process is led by a substance called pheromone that ants leave on the floor as they move along their path. If other ants find and follow the same path, this pheromone trail will be stronger, attracting other ants to follow it. On the other side, the pheromone is periodically evaporated; therefore, the worse paths gradually lose their pheromone trail. Thus, what at first seems to be a random behaviour for ants, when no pheromone trail is present on the ground, turns into a movement influenced by the substance left by other ants in the colony.

The Ant System (AS) was the first ACO algorithm, proposed by Dorigo et al. [8]. Later, a new approach, called Ant Colony System (ACS) [7], was defined.

This approach introduced some changes related to the mechanism used by the ants to select the next vertex, and to update the pheromone.

In ACS, the NRP is represented as a fully connected directed graph. Vertexes represent the candidate requirements $r_i, r_2, \ldots, r_n$ and a pheromone $\tau$ is associated to the edges joining pairs of requirements. Ants traverse the graph vertex by vertex constructing a new solution, but their movement is driven by an equation based on the heuristic information and the pheromone values.

The pheromone update [6] is performed both locally and globally. The local update $\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0$ (where $\varphi \in (0,1]$ is a pheromone decay coefficient and $\tau_0$ is the initial pheromone value) is applied by each individual ant only to the last edge traversed, when searching for its solution. Its main goal is to expand the search of subsequent ants during one iteration of the colony. The global update $\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}$ (where $\rho$ is the pheromone evaporation rate and $\Delta\rho_{ij}$ is the amount of pheromone left in each arc), is performed by the ant that has found the best solution during an iteration of the colony. It is a kind of global memory of the colony that stores the best paths (solutions) found.

At the time of building a solution, the ants apply the pseudorandom proportional rule [6]: an ant moves from requirement $i$ to $j$, depending on a random variable $q$ (that is uniformly distributed on the 0 to 1 range) and a parameter $q_0$, such that if $q \leq q_0$, then $j = argmax_{l \in nei(i)} \tau_{ij}\eta_{ij}^{\beta}$, otherwise $j$ is selected with a probability [6]

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, & \text{if } j \in N_i^k, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

where the set of visible nodes, $nei(i)$, from the current vertex $i$ is denoted by $N_i^k$. The heuristic information is defined by $\eta_{ij}$, whereas the pheromone accumulated in the edge $i$,$j$ is represented by $\tau_{ij}$. On the other side, the parameters $\alpha$ and $\beta$, reflect the relative influence of the pheromone with respect to the heuristic information.

Del Sagrado and del Águila [21] propose applying ACO to the requirement selection problem in the incremental development proposed by agile methodologies. The NRP using a single-objective approach is afforded in [22], and later in [21] applying a multi-objective perspective, defining this problem as NI-RSP (Next Increment Requirement Selection Problem). Both approaches are based on ACS. NI-RSP formulates a multi-objective problem seeking to maximize the total score, and minimize the total effort needed to develop. This technique is compared to other multi-objective optimization techniques, such as GRASP (greedy randomize adaptive search procedure) and NSGA-II, using some indicators to measure the quality of the Pareto front. The obtained results indicate that ACS can be applied efficiently to solve the requirement selection problem; its performance is very similar to NSGA-II and considerably better than GRASP. However, according to the quality indicators, it presents less oscillation in the number of non-dominated solutions.

## 4 Practical Application

This work has shown how optimization techniques applied to NRP let us find high quality solutions, in order to help developers during the requirement selection tasks. Once the applicability of these techniques has been demonstrated, they still have to be put in practice in real world software development. We strongly believe that having these search techniques available in a CARE tool would be considerably helpful for any development team at the time of dealing with the requirements selection.

InSCo Requisite [19] is a web-based tool early developed by our research group to manage the requirements of software development projects. Therefore, we propose an architecture (see Fig. 2) that integrates these techniques in the InSCo Requisite tool. The tool allows that a group of customers and developers works simultaneously in the same project, specifying the requirements of the system. Each requirement has an associated form which gathers its features, priorities and even a scenario or storyboard.

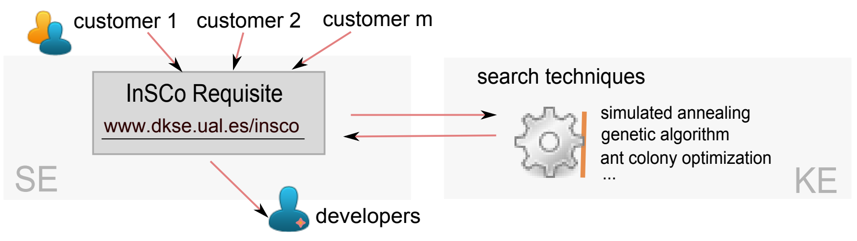

**Fig. 2.** Integration of search techniques in the architecture of the InSCo Requisite tool.

In order to facilitate the applicability of meta-heuristics algorithms, InSCo-Requisite must generate an interface file that contains all data needed during the execution of the simulated annealing, genetic or ant colony optimization algorithms.

The tool actually allows to export the whole set of specified requirements to an XML file. In a near future we plan to include development effort as a new property of each requirement. In this way, the resulting XML file could be easily used as input to any of the metaheuristic techniques applied in NRP. The result obtained by the metaheuristic techniques will be presented in the interface of InSCo Requisite and will serve as a feedback to developers when facing to the problem of planning the next software release.

## 5 Conclusions

The paper presented has provided a review of the metaheuristic techniques applied to the requirement selection problem, known as Next Release Problem

(NRP). This problem, within the Search Based Software Engineering (SBSE) discipline, was formulated in 2001 as a search problem and since then it has been addressed by many authors applying different search techniques: SA, GA and ACO. Table 3 summarizes the different works in the literature addressing the NRP, classified according to several factors as dependences are considered or not, and whether a single-objective or multi-objective perspective is applied.

**Table 3.** Classification of NRP related works

| NRP single-objective | | NRP multi-objective |
|---|---|---|
| With requirements dependences | Without requirements dependences | Without requirements dependences |
| *Greedy, SA*: Bagnall et al., 2001 [2] | *SA*: Baker et al., 2006 [3] | *GA*: Zhang et al., 2007 [25], Filkenstein et al., 2009 [10], Durillo et al., 2009 [9] |
| *GA*: Greer and Ruhe, 2004 [12] | *ACO*: del Sagrado et al., 2010 [22] | *ACO*: del Sagrado et al., 2009 [21] |

Although each technique has been reviewed in an isolated way, since the comparison is not feasible when different datasets are applied, the results obtained by all of them demonstrate their applicability to the NRP. Finally, an integration of these techniques in an existent requirement tool has been proposed in order to take advantage of these techniques in Software Engineering.

**Acknowledgments.** This work was supported by the Spanish Ministry of Education and Science under project TIN2007-67418-C03-02 and by the Junta of Andalucía under project P06-TIC-02411-02.

## References

1. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software product release planning through optimization and what-if analysis. Information and Software Technology 50(1-2), 101–111 (2008)
2. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Information and Software Technology 43(14), 883–890 (2001)
3. Baker, P., Harman, M., Steinhofel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: Procs. $22^{nd}$ IEEE Int. Conf. on Soft. Maintenance, 176–185. IEEE Computer Society (2006).
4. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.N.: An industrial survey of requirements interdependencies in software product release planning. In: Procs. $5^{th}$ IEEE Int. Symp. on Requirements Engineering. p. 84–91 (2001)
5. Clarke, J., Dolado, J.J., Harman, M., Hierons, R., Jones, B., Lumkin, M., Mitchell, B., Mancoridis, S., Rees, K., Roper, M., et al.: Reformulating software engineering as a search problem. IEE Proceedings-Software 150, 161–175 (2003)

6. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Computational Intelligence Magazine 1(4), 2839 (2006)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. On Evolutionary Computation 1(1), 53–66 (1997)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. on Sys., Man, and Cybernetics, Part B 26(1), 29–41 (1996)
9. Durillo, J.J., Zhang, Y.Y., Alba, E., Nebro, A.J.: A study of the multi-objective next release problem. In: Procs.1st Int. Symp. on Search Based Soft. Engineering. p. 49–58 (2009)
10. Finkelstein, A., Harman, M., Mansouri, S., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirements Engineering 14(4), 231–245 (2009)
11. Goldberg, D.E.: Genetic Algorithms in Search and Optimization. Addison-wesley (1989)
12. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology 46(4), 243–253 (2004)
13. Harman, M., Jones, B.F.: Search-based software engineering. Information and software technology 43(14), 833 (2001)
14. Harman, M.: The current state and future of search based software engineering. In: 2007 Future of Software Engineering, 342–357. IEEE Computer Society (2007)
15. Harman, M., Mansouri, S.A., Zhang, Y.: Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Tech. Rep. TR-09-03 (2009)
16. Johnson, J.: CHAOS chronicles v3.0. Tech. rep. (2003), `http://standishgroup.com/chaos/toc.php`
17. Karlsson, J., Ryan, K.: A Cost-Value approach for prioritizing requirements. IEEE Softw. 14(5), 67–74 (1997),
18. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. Wiley (Aug 1998)
19. Orellana, F.J., Canadas, J., del Águila, I.M., Túnez, S.: INSCO requisite - a Web-Based RM-Tool to support hybrid software development. In: ICEIS (3-1), 326–329 (2008)
20. Ruhe, G.: Software release planning. In: Handbook of software engineering and knowledge engineering, vol. 3, 365–394. S K Chang (2005)
21. del Sagrado, J., del Águila, I.M.: Ant colony optimization for requirement selection in incremental software development. Technical Report, University of Almería (2009)
22. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem. a comparative study. In: Procs. 2nd Int. Symp. on Search Based Software Engineering (2010)
23. Sommerville, I.: Software engineering (6th ed.). Addison-Wesley Longman Publishing Co., Inc. (2001)
24. Zhang, Y., Finkelstein, A., Harman, M.: Search based requirements optimisation: Existing work and challenges. In: Procs. 14th Int. Conf. on Requirements Engineering: Foundation for Soft. Quality, 88–94. Springer-Verlag, Montpellier, France (2008)
25. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Procs. 9th Ann. Conf. on Genetic and Evol. Computation, 1129–1137. ACM, London, England (2007)