Empir Software Eng (2015) 20:577–610
DOI 10.1007/s10664-013-9287-3

# Multi-objective ant colony optimization for requirements selection

**José del Sagrado · Isabel M. del Águila · Francisco J. Orellana**

Published online: 30 November 2013


**Abstract** The selection of a set of requirements between all the requirements previously defined by customers is an important process, repeated at the beginning of each development step when an incremental or agile software development approach is adopted. The set of selected requirements will be developed during the actual iteration. This selection problem can be reformulated as a search problem, allowing its treatment with metaheuristic optimization techniques. This paper studies how to apply Ant Colony Optimization algorithms to select requirements. First, we describe this problem formally extending an earlier version of the problem, and introduce a method based on Ant Colony System to find a variety of efficient solutions. The performance achieved by the Ant Colony System is compared with that of Greedy Randomized Adaptive Search Procedure and Non-dominated Sorting Genetic Algorithm, by means of computational experiments carried out on two instances of the problem constructed from data provided by the experts.

**Keywords** Software requirements · Search based software engineering · Ant colony optimization · Next release problem

Communicated by: Mark Harman

J. del Sagrado · I. M. del Águila (✉) · F. J. Orellana
Departamento de Informática, Universidad de Almería,
Ctra. Sacramento s/n, 04120 Almería, Spain
e-mail: imaguila@ual.es

J. del Sagrado
e-mail: jsagrado@ual.es

F. J. Orellana
e-mail: fjorella@ual.es

## 1 Introduction

Software development organizations fail many times to deliver its products within schedule and budget. Statistical studies, and CHAOS Reports (Johnson 2003) published since 1994, reveal that, frequently, tasks related to requirements lead software project to the disaster. As Kotonya and Sommerville (1998) suggest, one of the major problems we face when developing large and complex software systems is the one related with requirements. The concept of requirement, in its broadest sense, must be understood as a logical unit of behaviour that is specified by including functional and quality aspects; other authors as Ruhe and Saliu (2005) use instead the concept of feature. Stakeholders propose some desired functionalities that software managers must filter in order to define the set of requirements or features to include in the final software product.

Usually, during the lifetime of a software product, we are faced with the problem of selecting a subset of requirements from the whole set of candidate requirements (e.g. when a new release is being planned). Enhancements to include into the next software release cannot be randomly selected since there are many factors involved. Within this scenario, customers demand their own software enhancements, but all of them cannot be included in the software product, mainly due to the existence limited resources (e.g. availability of man-month in a given software project). In most cases, it is not feasible to develop all the new functionalities suggested. Hence each new feature competes against each other to be included in the next release.

This task of selecting a set of requirements, which until now only appeared when defining new versions of widely distributed software products, becomes important within the incremental approaches of software development, and specially in the agile approaches. Agile methods promote a disciplined project management process that encourages frequent inspection and adaption. These software development methodologies are based on iterative development, advocating for frequent "releases" in short development cycles, called timeboxes, in order to improve productivity and introduce checkpoints. Each iteration works through a full cycle, generating a software release that has to be shown to stakeholders. These approaches focus on the quick adaptation of software to the changing realities.

Within this prospective, the challenge of Software Engineering consists on defining specific techniques or methods that improve the way requirements are selected. The problem of choosing a set of requirements fulfilling certain criteria, such as minimal cost or maximal client satisfaction, is a good candidate for the application of metaheuristics. Specifically, this paper shows how Ant Colony Optimization (ACO) systems can be applied to problems of requirements selection. Our solution offers to developers and stakeholders a set of possibilities satisfying several objectives, i.e. the Pareto front. The idea is to help people to take a decision about which set of requirements has to be included in the next release during the software development applying both, agile or classical software development approaches. The proposed ACO system is evaluated by means of a compared analysis with Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb 2001; Deb et al. 2002) and Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende 1989; Pitsoulis and Resende 2003; Resende and Ribeiro 2003); both adapted to the problem of requirements selection.

The rest of the paper is organized in six sections. Section 2 summarizes the basic Requirements Engineering concepts, focusing in requirement selection process, together with the description and definition of the problem of selection of requirements to be included in the next software release, known as Next Release Problem (NRP) (Bagnall et al. 2001), and related works. Section 3 describes the metaheuristic technique applied in this work, Ant Colony Optimization. Specifically, we focus on its application in multi-objective optimization problems, on the study of how multi-objective ACO can be used to find a set of non-dominated solutions to NRP. In Section 4, the experimental evaluation is carried out by comparing ACO with GRASP and NSGA-II approaches. The analysis of the results is presented in Section 5. Section 6 considers threats to the validity of the findings in the study. Finally, in Section 7 we give some conclusions and the future works that can extend this study.

## 2 Requirements Selection

Requirements related tasks are inherently difficult Software Engineering activities. Descriptions of requirements are supposedly written in terms of the domain, describing how this environment is going to be affected by the new software product. In contrast, other software processes and artifacts are written in terms of the internal software entities and properties (Cheng and Atlee 2007).

The problem of selecting the subset of requirements among a whole set of candidate requirements proposed by a group of customers is not a straightforward problem, since there are many factors involved. Customers, seeking their own interest, demand the set of enhancements they consider important, but not all customers' 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 the requirements are 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 that every customer sees fulfilled at least one of their proposed requirements. Also, requirements show interactions that impose a certain development order or either conflicts between them, limiting the alternatives to be chosen.

During the software development process many interaction types between two or more requirements can be identified. Karlsson et al. (1997) were the first in proposing a list of interaction types. Later, Carlshamre et al. (2001) propose a set of interaction types as result of an in-depth study of interactions in distinct sets of requirements coming from different software development projects. Although interaction types are semantically different, in practice they can be grouped into:

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

These interactions must be considered as constraints in the requirements selection problem. So, two main goals are usually considered: maximize the customers satisfaction and minimize the required software development effort satisfying the given constraints. Therefore, optimization techniques can be used to find optimal or near optimal solutions in a reasonable amount of time. Our aim is to define the requirements selection problem as a science (Ruhe and Saliu 2005), formalizing the problem and applying computational algorithms to generate good solutions.

### 2.1 Previous and Related Works

The Search-Based Software Engineering (SBSE) area is the research field which proposes the application of search-based optimization algorithms to tackle problems in Software Engineering (Harman and Jones 2001; Harman 2007). In this section we provide a comprehensive review of different approaches that can be found in the literature to tackle with the requirements selection problem (an earlier review can be found in del Sagrado et al. 2010b).

As a problem in which is necessary to evaluate multiple conflicting objectives, its solution requires to find the best compromise between the different objectives. In order to achieve this, we can proceed in two ways. The first approach consists in transforming the multi-objective problem into a single objective problem. To do that, we need to combine the different objectives into a single one by means of an aggregation function (e.g. a weighted sum or product). This is the approach chosen by Bagnall et al. (2001), they formulate the problem of selecting a subset of requirements (i.e. Next Release Problem-NRP) having as goal meet the customer's needs, minimizing development effort and maximizing customers satisfaction and apply hill climbing, greedy algorithms and simulated annealing. Later, Baker et al. (2006) demonstrate that these metaheuristics techniques can be applied to a real-world NRP out-performing expert judgment. Greer and Ruhe (2004) study the generation of feasible assignments of requirements to increments taking into account different stakeholders' perspectives and resources constraints. The optimization method they used is iterative and essentially based on a genetic algorithm.

With respect to the application of ACO to tackle the single-objective version of NRP, the first approach can be found in the works of del Sagrado and del Águila (2009) and del Sagrado et al. (2010a) where an Ant Colony System (ACS) is proposed. Jiang et al. (2010) incorporate into ACO a local search for improving the quality of solution found. Nonetheless, all of these approaches do not take into account the existence of interactions between requirements. The problem of requirements selection, including interactions between requirements, is introduced in the works of del Sagrado et al. (2011) and de Souza et al. (2011) by adapting the ACS algorithm.

The second approach to multi-objective optimization builds a set with all the solutions found that are not dominated by any other (this is known as the set of efficient solutions or Pareto-optimal set). The decision maker will select a solution

from this set according to personal criteria. The works of Saliu and Ruhe (2007), Zhang et al. (2007), Finkelstein et al. (2008, 2009) and Durillo et al. (2009), study the NRP problem from the multi-objective point of view, either as an interplay between requirements and implementation constraints (Saliu and Ruhe 2007) or considering multiple objectives as cost-value (Zhang et al. 2007) or different measures of fairness (Finkelstein et al. 2008, 2009), or applying several algorithms based on genetic inspiration as NSGA-II, MOCell and PAES (Durillo et al. 2009, 2011).

We introduce a new multi-objective search-based approach based on ACO to the problem of selecting requirements to be included in the next software release, in the presence of several stakeholders (with their own importance perception of requirements) and different types of interactions between requirements. The proposed approach automates the search for optimal or near-optimal sets of requirements, within a given effort bound, that balance stakeholders' priorities while keeping requirements interactions. It is worth to note that, to date, no ACO approach has been applied to multi-objective NRP and that having more than one valid solution, as in the multi-objective approach, constitute a valuable aid for experts who must decide what is the set of requirements that has to be considered in the next software release. Requirements managers analyze these alternatives and their data (e.g. number of customer covered, additional information about risky requirements), before selecting a solution (i.e. the set of requirements to be developed) according to business strategies. Thus, it is considerably helpful, for any software developer, to have these techniques available either embedded in a CASE (Computer-Aided Software Engineering) tool (del Sagrado et al. 2012), or within a decision support tool for release planning (Carlshamre 2002; Momoh and Ruhe 2006).

### 2.2 NRP Formulation

Let $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$ be the set of requirements that are still to be implemented. These requirements represent enhancements to the current system that are suggested by a set of $m$ customers and are candidates to be included in the next software release. Customers are not equally important. So, each customer $i$ will have an associated weight $w_i$, which measures its importance. Let $\mathbf{W} = \{w_1, w_2, \ldots, w_m\}$ be the set of customers' weights.

Each requirement $r_j$ in $\mathbf{R}$ has an associated development cost $e_j$, which represents the effort needed for its development. Let $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$ 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 software release, is measured as a weighted sum of its importance values for all the customers and can be formalized as:

$$s_j = \sum_{i=1}^{m} w_i * v_{ij} \tag{1}$$

The set of requirements satisfaction computed in this way is denoted as $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$. Requirements interactions can be divided into two groups. The first consists of the functional interactions: *implication, combination* and *exclusion*. The second one includes those interactions that imply changes in the amount of resources needed or the benefit related to each requirement: *revenue-based* and *cost-based*. Functional interactions can be explicitly represented as a graph $G = (\mathbf{R}, \mathbf{I}, \mathbf{J}, \mathbf{X})$ where:

- $\mathbf{R}$ (the set of requirements) is the set of nodes
- $\mathbf{I} = \{(r_i, r_j) \mid r_i \Rightarrow r_j\}$ each pair $(r_i, r_j) \in \mathbf{I}$ is an implication interaction and will be represented as a directed link $r_i \rightarrow r_j$
- $\mathbf{J} = \{(r_i, r_j) \mid r_i \odot r_j\}$ each pair $(r_i, r_j) \in \mathbf{J}$ is a combination interaction and will be represented as a double directed link $r_i \leftrightarrow r_j$
- $\mathbf{X} = \{(r_i, r_j) \mid r_i \oplus r_j\}$ each pair $(r_i, r_j) \in \mathbf{X}$ is an exclusion interaction and will be represented as a crossed undirected link

For example, consider the set of requirements $\mathbf{R} = \{r_1, r_2, \ldots, r_{10}\}$ and the following functional dependencies $\mathbf{I} = \{(r_1, r_3), (r_1, r_6), (r_2, r_4), (r_2, r_5), (r_4, r_6), (r_5, r_7), (r_7, r_8), (r_7, r_9), (r_8, r_{10}), (r_9, r_{10})\}$, $\mathbf{J} = \{(r_3, r_4)\}$, $\mathbf{X} = \{(r_4, r_8)\}$. We can represent all of these sets as the graph showed in Fig. 1.

The second group of interactions represents changes in satisfaction and effort values of individual requirements. These two interaction types can be modelled as a pair of $n \times n$ symmetric matrices:

- $\Delta S$ where each element $\Delta s_{ij}$ of this matrix represents the increment or decrement of $s_j$ when $r_i$ and $r_j$ are implemented in the same release,
- $\Delta E$ where each element $\Delta e_{ij}$ of this matrix represents the increment or decrement of $e_j$ when $r_i$ and $r_j$ are implemented simultaneously,

in which the elements in the diagonal are equal to zero.

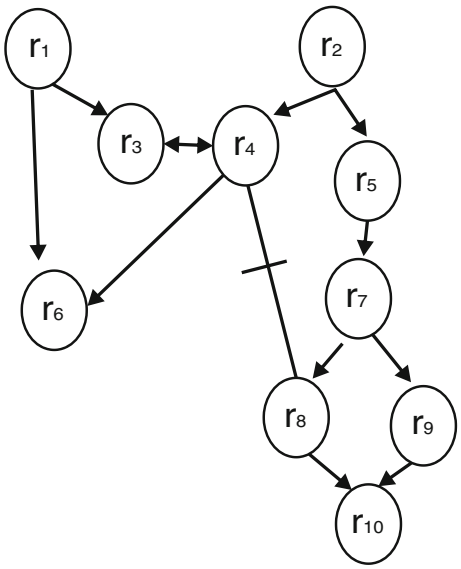



**Fig. 1** Functional interactions represented as a graph $G = (\mathbf{R}, \mathbf{I}, \mathbf{J}, \mathbf{X})$

In order to define the next software release, we have to select a subset of requirements $\hat{\mathbf{R}}$ included in $\mathbf{R}$, which maximize satisfaction and minimize development effort. The satisfaction and development effort of the next release can be obtained, respectively, as:

$$\text{sat}(\hat{\mathbf{R}}) = \sum_{j \in \hat{\mathbf{R}}} s_j \tag{2}$$

$$\text{eff}(\hat{\mathbf{R}}) = \sum_{j \in \hat{\mathbf{R}}} e_j \tag{3}$$

where $j$ is an abbreviation for requirement $r_j$. For a given release there is a cost bound $B$, that cannot be overrun. Under this set of circumstances the requirements selection problem for the next software release can be formulated as an optimization problem:

$$\begin{array}{l} \text{maximize sat}(\hat{\mathbf{R}}) \\ \text{minimize eff}(\hat{\mathbf{R}}) \end{array} \tag{4}$$

subject to the restriction $\sum_{j \in \hat{\mathbf{R}}} e_j \leq B$ due to the particular effort bound $B$ applied, where $\hat{\mathbf{R}}$ also fulfills functional requirements interactions. Thus, two conflicting objectives, such as maximizing customer satisfaction and minimizing software development effort, are optimized at the same time within a given software development effort bound. The solution to this problem consists of a set of solutions known as Pareto-optimal set (Coello et al. 2007; Deb 2001; Srinivas and Deb 1994).

### 2.3 Basic Instance of NRP

Bagnall et al. (2001) defines a *basic* NRP as an NRP where no requirement has any prerequisites. So, following Bagnall formulation, any NRP can be transformed into a *basic* NRP simply by grouping together each requirement and its ancestors in the graph of precedence interactions (i.e. requirements for which there is a path in the graph ending in the requirement that is being considered). However, there are also other types of interactions besides *implication* (i.e. *combination, exclusion, revenue-based* and *cost-based* interactions) that also have to be taken into account when solving an NRP. Therefore, we are going to extend the work of Bagnall et al. (2001), defining a process for transforming an NRP into a *basic* NRP, before applying metaheuristics optimization techniques.

The process to transform an NRP into a *basic* NRP is made on three steps:

1. Each pair $(r_i, r_j) \in \mathbf{J}$ is transformed into a new requirement $r_{i+j}$, with $s_{i+j} = s_i + s_j + \Delta s_{ij}$ and $e_{i+j} = e_i + e_j + \Delta e_{ij}$. The ancestors set of $r_{i+j}$ is defined as the union of the ancestors of $r_i$ and $r_j$, $anc(r_{i+j}) = anc(r_i) + anc(r_j)$. The same applies to descendant set (i.e. requirements for which there is a path in the graph starting in the requirement that is being considered) of $r_{i+j}$, that is, $des(r_{i+j}) = des(r_i) + des(r_j)$. In this way, the set $\mathbf{I}$ of implication interactions is modified. Finally, all occurences of $r_i$ and $r_j$ in the set of $\mathbf{X}$ of exclusion interactions are replaced by $r_{i+j}$. This produces a new functional interactions graph $G' = (\mathbf{R}, \mathbf{I}, \mathbf{J}, \mathbf{X})$ in which $\mathbf{J} = \emptyset$.

2. A pair $(r_i, r_j) \in \mathbf{X}$ requires the alternative deletion of each requirement together with its descendants from $G'$, resulting two new functional interactions graphs $G'_i$ and $G'_j$. This process is repeated from these new graphs until there are no exclusion interactions.
3. For each interactions graph $G'_i$ obtained in step 2, a *basic* NRP is built by grouping together implication interactions (i.e. each requirement and its ancestors in the graph), $r_j^+ = \{r_j\} \cup anc(r_j)$, which causes that $s_j^+ = \sum_{r_k \in r_j^+} s_k + \sum_{r_k, r_l \in r_j^+, k<l} \Delta s_{kl}$ and $e_j^+ = \sum_{r_k \in r_j^+} e_k + \sum_{r_k, r_l \in r_j^+, k<l} \Delta e_{kl}$.

Revenue-based and cost-based interactions are updated taking into account these graphs and the values of the original NRP:

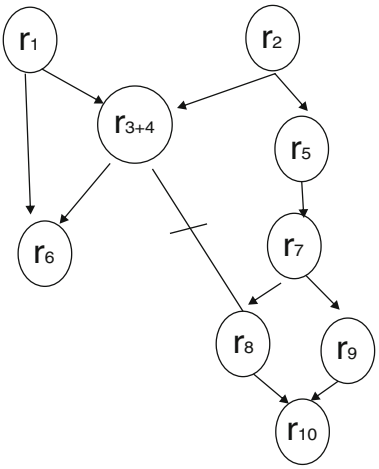
1. Each pair $(r_i, r_j) \in \mathbf{J}$ implies that, starting from $\Delta S' = \Delta S$, $\Delta s'_{ik} = \Delta s_{ik} + \Delta s_{jk}$ and $\Delta s'_{ki} = \Delta s'_{ik}$, for $i \neq k$, and $\Delta s'_{ii} = 0$. After that, the row and column associated to $r_j$ are deleted from $\Delta S'$. The same process applies to cost-based interactions. As result, we obtain new revenue-based, $\Delta S'$, and cost-based, $\Delta E'$, matrices.
2. A pair $(r_i, r_j) \in \mathbf{X}$ requires the alternative deletion of the rows and columns associated to each requirement and its descendants in $G'$ from $\Delta S$ and $\Delta E$. Thus, the deletion of the rows and columns associated to $\{r_i\} \cup des(r_i)$ produces $\Delta S'_i$ and $\Delta E'_i$, whereas that of $\{r_j\} \cup des(r_j)$ produces $\Delta S'_j$ and $\Delta E'_j$. This process is repeated from these new matrices and the graphs $G'_i$ and $G'_j$ until there are no exclusion interactions.
3. For each $r_j^+$ in the *basic* NRP built from an interactions graph $G'_i$ together with its pair of associated matrices $\Delta S'_i$ and $\Delta E'_i$, we define the elements of revenue-based, $\Delta S^+$, and cost-based, $\Delta E^+$ matrices as the sum of the values associated to unshared requirements,

$$\Delta s_{ij}^+ = \begin{cases} \sum_{\substack{r_k \in r_i^+ \\ r_l \in r_j^+}} \Delta s'_{kl}, & \text{if } r_k, r_l \notin r_i^+ \cap r_j^+, \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

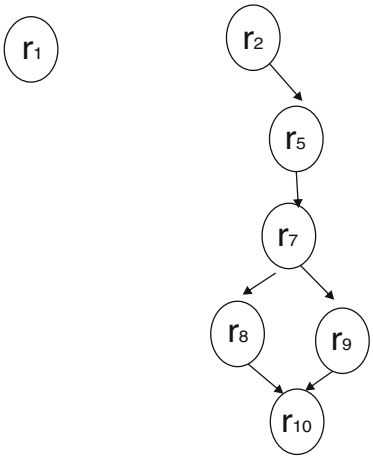
$$\Delta e_{ij}^+ = \begin{cases} \sum_{\substack{r_k \in r_i^+ \\ r_l \in r_j^+}} \Delta e'_{kl}, & \text{if } r_k, r_l \notin r_i^+ \cap r_j^+, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

For example, if we consider the NRP depicted in Fig. 1 as a functional interactions graph, then during the first stage of the process, combination interactions are erased and the graph shown in Fig. 2a is obtained. The second stage takes this functional interactions graph and proceeds to eliminate exclusion dependences. Thus, requirement $r_{3+4}$ and its descendants are deleted which produces the graph $G'_{3+4}$ (see Fig. 2b) and the elimination of requirement $r_8$ along with its descendants, produces the graph $G'_8$ (see Fig. 2c). Last two graphs contain only implication interactions and define the basic NRP.

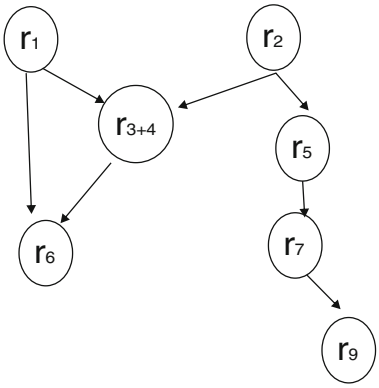
Once we get the set of functional interactions graphs, containing only implication interactions, the transformation ends simply by grouping together each requirement and its ancestors in each graph, obtaining several basic NRPs. For example, if we consider the graph $G'_{3+4}$ in Fig. 2b we get $r_1^+ = \{r_1\}$, $r_2^+ = \{r_2\}$, $r_5^+ = \{r_2, r_5\}$, $r_7^+ = \{r_2, r_5, r_7\}$, $r_8^+ = \{r_2, r_5, r_7, r_8\}$, $r_9^+ = \{r_2, r_5, r_7, r_9\}$ and $r_{10}^+ = \{r_2, r_5, r_7, r_8, r_9, r_{10}\}$.




(a) Functional interactions graph *G'* after eliminating combinations interactions




(b) Functional interactions graph $G_{3+4}$ after deleting $r_{3+4}$ and its descendants from *G'*




(c) Functional interactions graph $G_8$ after deleting $r_8$ and its descendants from *G'*

**Fig. 2** Transformation of an NRP into a *basic* NRP

After executing these processes of eliminating combination and exclusion interactions, we obtain a set of graphs, containing only implication interactions. Observe that the presence of exclusion interactions causes an alternative consideration of requirements and the appearance of a greater number of interactions graphs. This is due to the inner nature of exclusion. If there is an exclusion interaction between two requirements, $(r_i, r_j) \in \mathbf{X}$, by definition these requirements are incompatible software features that cannot be included in the same software product at the same time. Requirements Engineering faces this problem as a negotiation problem. Project developers must obtain an agreement by the elimination of one of them (i.e. discarding one of the requirements) or by generating two different software applications. For example, consider the exclusive requirements $r_i$: "all users should be able to search for data about both products and customers" and $r_j$: "only personnel with a high security level should be able to search for customers classified as military related". Each one of them leads to a different software product and we need to perform different search processes defining different alternatives for the next software release.

Note that a more restrictive special case arises when requirements are basic and independent: for all $r_i \in \mathbf{R}$, $r_i^+ = \{r_i\}$. Bagnall et al. (2001) has shown that NRP is an instance of 0/1 knapsack problem, which is NP-hard (Garey and Johnson 1990). This result means that large instances of the problem cannot be solved efficiently by exact optimization techniques. Nonetheless, in this situation the use of metaheuristics is suitable because they can find near-optimal solutions to NRP spending a reasonable amount of time.

## 3 Multi-Objective Ant Colony Optimization for the NRP

Ant Colony Optimization (ACO) has been applied to multi-objective optimization problems (Iredi et al. 2001; Doerner et al. 2004; Häckel et al. 2008) using a multi-colony strategy, extending the Ant Colony System (ACS) (Dorigo et al. 2006; Dorigo and Stützle 2004). Iredi et al. (2001) and Häckel et al. (2008) propose a multi-colony method to solve multi-objective optimization problems when the objectives cannot be ordered by importance, in which each colony searches for a solution in a different region of the Pareto-front. Whereas Doerner et al. (2004) propose the use of a single colony in which each ant searches in a different direction of the Pareto front, and apply it specifically to a portfolio optimization problem. This last approach enables us to extend in a natural way the ACS for NRP, described in del Sagrado et al. (2010a, 2011), to the multi-objective case by searching in different directions on the Pareto front.

In ACO algorithms, each ant builds its solution from an initial node (requirement) which is selected randomly. At each stage, an ant locates a set of neighbouring nodes to visit (these requirements must satisfy the restrictions of the problem). Among all of them it selects one in a probabilistic way, taking into account the pheromone level and heuristic information. The level of pheromone deposited in an arc from node $r_i$ to node $r_j$, $\tau_{ij}$, is stored in a matrix $\tau$, whereas the heuristic information about the problem is represented as the value $\eta_{ij}$ and have to be defined based on the problem itself.

Many authors Brooks (1995), Boehm (1981) and Albrecht and Gaffney (1983) have proposed several metrics for software projects and products, e.g. man-moths

(Brooks 1995), number of lines of code (Boehm 1981) and function points (Albrecht and Gaffney 1983), are only some of them. These metrics help in the assessment of the quality of a software product when its development has finished (i.e. quality metrics) and also in the estimation of the effort of a new software project, either using productivity metrics (e.g. time to delivered a feature or requirement to the final product, total number of units of production, or amount of units produced during a given period of time) or financial metrics (e.g. investment, operating expense that includes salaries, money amount of software sold minus the cost, return of investment, or cost of development effort per feature produced). Since we have to select requirements and the data available are related to requirements, we should consider a metric that measure the productivity in terms of the customers' benefit and development effort (sets $S$ and $E$). This concept has been applied in the selection and triage of the requirements (Davis 2003; Simmons 2004). In NRP we can define a productivity metric associated to a requirement $r_j$, as, $s_j/e_j$ which is the level of satisfaction obtained by the customers when including this requirement in an increment based on the effort applied in its development. Hence, we define:

$$\eta_{ij} = \xi \frac{s_j}{e_j} \tag{7}$$

where $\xi$ is a normalization constant.

Let $\mathbf{R}^k \subseteq \mathbf{R}$ be the partial solution to the problem built by ant $k$ and assume that the ant is located at node $r_i$, then

$$N_i^k = \{r_j | r_j \notin \mathbf{R}^k,\ eff(\mathbf{R}^k) + e_j \leq B, \\ \mathbf{R}^k \cup \{r_j\} \text{ fulfills functional interactions}\} \tag{8}$$

represents the set of non visited neighbours nodes that can be reached by ant $k$ from node $r_i$. That is, for ant $k$ a node $r_j$ is visible from node $r_i$ if and only if $r_j$ has not been previously visited, its inclusion in the partial solution $\mathbf{R}^k$ does not exceed the fixed development effort bound $B$ and does not break functional interactions.

In the multi-objective ACO, there will be a pheromone matrix $\tau^g$ for each one of the objectives, $g \in \mathbf{O}$ (for NRP the set of objectives is $\mathbf{O} = \{s, e\}$, where $s$ represents satisfaction and $e$ represents effort) and the solution constructed by one ant is based on a weighted combination of these pheromone matrices. The weights $\lambda_g^k$, that ant $k$ assigns to the different objectives, measure the relative importance of the optimization criteria and should be distributed uniformly over the different regions of the Pareto front. For the NRP case, if the colony has $z$ ants, then the weights for users' satisfaction and development effort used by the ant $k \in [0, z-1]$ are defined respectively as

$$\lambda_s^k = \frac{k}{(z-1)} \text{ and } \lambda_e^k = 1 - \lambda_s^k \tag{9}$$

Note that weights are kept fixed over the ant's lifetime (i.e. time expended by the ant to build its solution).

In the ACS algorithm, each ant builds, in a progressive way, a solution to the problem. During the construction process of the solution, ant $k$ selects from node

$r_i$ the next node $r_j$ to visit applying a pseudorandom proportional rule (Dorigo and Gambardella 1997) that takes into account the weights $\lambda_g^k$ (Doerner et al. 2004):

$$j = \begin{cases} argmax_{u \in N_i^k} \left\{ \left[ \sum_{g \in \mathbf{O}} \lambda_g^k \tau_{iu}^g \right]^\alpha [\eta_{iu}]^\beta \right\}, & \text{if } q \leq q_0, \\ u, & \text{otherwise} \end{cases} \tag{10}$$

where $q$ is a random number uniformly distributed in [0, 1], $q_0 \in [0, 1]$ is a parameter that determines a trade-off between exploitation ($q \leq q_0$) and exploration, and $u \in N_i^k$ is a node randomly selected. This selection is made by the ant $k$, which selects randomly from node $r_i$ the next node $u = r_j$ to visit with a probability $p_{ij}^k$ given by Doerner et al. (2004):

$$p_{ij}^k = \begin{cases} \dfrac{\left[ \sum_{g \in \mathbf{O}} \lambda_g^k \tau_{ij}^g \right]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in N_i^k} \left[ \sum_{g \in \mathbf{O}} \lambda_g^k \tau_{ih}^g \right]^\alpha [\eta_{ih}]^\beta}, & \text{if } j \in N_i^k, \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where the parameters $\alpha$ and $\beta$ reflect the relative influence of the pheromone with respect to the heuristic information. For example, if $\alpha = 0$ the nodes with higher heuristic information values will have a higher probability of being selected (the ACO algorithm will be close to a classical greedy algorithm). If $\beta = 0$ the nodes with higher pheromone value will be preferred in order to be selected. From these two examples, it is easy to deduce that is needed a balance between heuristic information and pheromone level.

While building its solution each ant $k$ in the colony *updates pheromone locally*. If it chooses the transition from node $r_i$ to $r_j$, then it has to update the pheromone level of the corresponding arc for each objective $g$ applying the following rule:

$$\tau_{ij}^g = (1 - \varphi) * \tau_{ij}^g + \varphi \tau_0 \tag{12}$$

where $\varphi \in [0, 1]$ is the pheromone decay coefficient and $\tau_0$ is the initial pheromone value, which is defined as $\tau_0 = 1/|\mathbf{R}|$. Each time an arc is visited, its pheromone level decreases making it less attractive for subsequent ants. Thus, this local update process encourages the exploration of other arcs avoiding premature convergence.

Once all ants in the colony have built a solution, only the two ants that have obtained the best solutions reinforce pheromone on the arcs that are part of the best solutions. They *update pheromone globally* for each objective $g \in \mathbf{O}$ and each arc, $(i, j)$, included in the best solutions applying the following rule:

$$\tau_{ij}^g = (1 - \rho) * \tau_{ij}^g + \rho \Delta \tau_{ij}^g \tag{13}$$

where $\rho \in [0, 1]$ is the pheromone evaporation rate and $\Delta \tau_{ij}^g$ represents the increase of pheromone with respect to objective $g$.

In the NRP case, we have two objectives, $\mathbf{O} = \{s, e\}$, satisfaction and effort, so, for a given best solution $\hat{\mathbf{R}}$, we define two pheromone increments ($\Delta \tau_{ij}^s$ for satisfaction and $\Delta \tau_{ij}^e$ for effort) as:

$$\Delta \tau_{ij}^s = \frac{1}{sat(\hat{\mathbf{R}})} \tag{14}$$

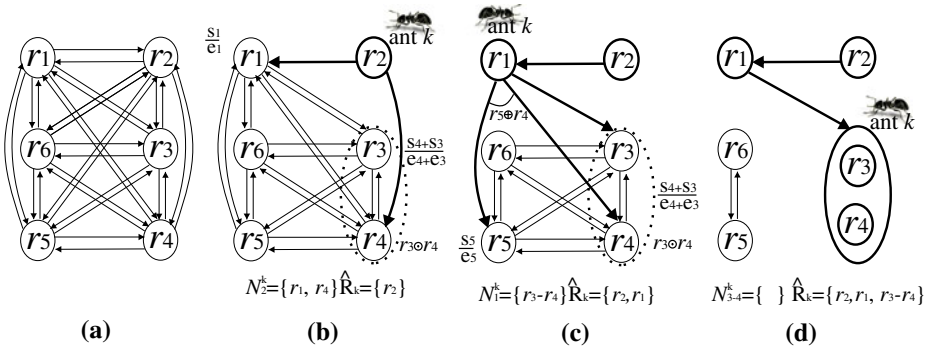



**Fig. 3** Search process in ACS

$$\Delta\tau_{ij}^{e} = \frac{1}{eff(\hat{\mathbf{R}})} \tag{15}$$

where $sat(\hat{\mathbf{R}})$ and $eff(\hat{\mathbf{R}})$ are the evaluations of the best solution (see (2) and (3)) with respect to each objective.

For example, Fig. 3 shows an example of the steps followed by the ant $k$ during an iteration. A thin arrow represents a candidate path for the ant and a bold arrow is a path yet crossed. Figure 3a shows the set of six requirements $\mathbf{R} = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ with associated development efforts and customers satisfaction sets, $\mathbf{E} = \{3, 4, 2, 1, 4, 1\}$, $\mathbf{S} = \{1, 2, 3, 2, 5, 4\}$, respectively. Also, consider the set of functional dependencies $\mathbf{F} = \{r_1 \Rightarrow r_3, r_1 \Rightarrow r_5, r_2 \Rightarrow r_4, r_2 \Rightarrow r_5, r_5 \Rightarrow r_6, r_3 \odot r_4, r_4 \oplus r_5\}$ and that the development effort bound $B$ is set to a value of 11. Figure 3b to d depict the steps followed by ant $k$. Initially (see Fig. 3b), the ant chooses randomly a requirement from the set $\{r_1, r_2\}$ that contains the visible requirements (i.e. those verifying requirements interactions and whose effort limit is lower than $B$). Suppose that it selects $r_2$ as the initial node and adds it to its solution, $\mathbf{R}^k = \{r_2\}$. Then the ant obtains the set of non visited neighbours nodes from $r_2$ is $N_2^k = \{r_1\}$ and arcs reaching to $r_2$ have been deleted because they could never be used. In this example, we are going to assume that the ant only uses heuristic information in order to build its solution $\mathbf{R}^k$, so at each step it will choose the vertex with the highest $\mu_{ij}$ value from the set of non visited neighbours nodes. Now the ant adds $r_1$ to its solution $\mathbf{R}^k = \{r_2, r_1\}$ and searches for a new requirement to add from this node as it is depicted in Fig. 3c. In this situation the ant's set of neighbours nodes is $N_1^k = \{r_3 - r_4, r_5\}$ and using only heuristic information the next node to travel to is $r_3 - r_4$ (note that this node is consequence of the combination interaction $r_3 \odot r_4$. Finally, Fig. 3d shows that once the ant has added $r_3 - r_4$ to its solution, $\mathbf{R}^k = \{r_2, r_1, r_3, r_4\}$, it has to stop because there are not any other visible vertex due to the exclusion relationship $r_4 \oplus r_5$.

## 4 Experimental Evaluation

In this section, we describe the aspects related with the design of the experiments for making the performance evaluation of the multi-objective ACO algorithm proposed.

**Table 1** Dataset 1: assignment of the priority level of each requirement, requirements development effort and interactions

| | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ | $r_{10}$ | $r_{11}$ | $r_{12}$ | $r_{13}$ | $r_{14}$ | $r_{15}$ | $r_{16}$ | $r_{17}$ | $r_{18}$ | $r_{19}$ | $r_{20}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 4 | 2 | 1 | 2 | 5 | 5 | 2 | 4 | 4 | 4 | 2 | 3 | 4 | 2 | 4 | 4 | 4 | 1 | 3 | 2 |
| $c_2$ | 4 | 4 | 2 | 2 | 4 | 5 | 1 | 4 | 4 | 5 | 2 | 3 | 2 | 4 | 4 | 2 | 3 | 2 | 3 | 1 |
| $c_3$ | 5 | 3 | 3 | 3 | 4 | 5 | 2 | 4 | 4 | 4 | 2 | 4 | 1 | 5 | 4 | 1 | 2 | 3 | 3 | 2 |
| $c_4$ | 4 | 5 | 2 | 3 | 3 | 4 | 2 | 4 | 2 | 3 | 5 | 2 | 3 | 2 | 4 | 3 | 5 | 4 | 3 | 2 |
| $c_5$ | 5 | 4 | 2 | 4 | 5 | 4 | 2 | 4 | 5 | 2 | 4 | 5 | 3 | 4 | 4 | 1 | 1 | 2 | 4 | 1 |
| Eff. | 1 | 4 | 2 | 3 | 4 | 7 | 10 | 2 | 1 | 3 | 2 | 5 | 8 | 2 | 1 | 4 | 10 | 4 | 8 | 4 |

$r_3 \odot r_{12}$ $r_{11} \odot r_{13}$

$r_4 \Rightarrow r_8$ $r_4 \Rightarrow r_{17}$ $r_8 \Rightarrow r_{17}$ $r_9 \Rightarrow r_3$ $r_9 \Rightarrow r_6$ $r_9 \Rightarrow r_{12}$ $r_9 \Rightarrow r_{19}$ $r_{11} \Rightarrow r_{19}$

First, we present the data used in the experiments. Then, we briefly describe other two metaheuristic algorithms used in the experiments. Finally, we define the set of quality measures applied and the comparison methodology we have followed.

### 4.1 Datasets

For testing the effectiveness of our proposed ACS we have used two datasets. The first dataset is taken from Greer and Ruhe (2004). It has 20 requirements and 5 customers. The development effort associated to each requirement and the level of priority or value assigned by each customer to each requirement are shown in Table 1. The customers' weights are given in the 1 to 5 range, following a uniform distribution (see Table 2). These values (and also those of the level of priority of each requirement) can be understood as linguistic labels such as: without importance (1), less important (2), important (3), very important (4), extremely important (5). Each requirement has an associate effort estimate in terms of score between 1 and 10. Also, we consider a set of implication and combination interactions between requirements. The reason for not including exclusion interactions is that its presence leads to different alternatives which can be considered as independent NRP instances (see Section 2.3). The main reason to use this dataset resides in its wide use in the evaluation of other studies of distinct instances of NRP (Finkelstein et al. 2009; Zhang et al. 2007; Durillo et al. 2009) and, as far as we know, the lack of other available real datasets due to the privacy policies followed by software development companies. At the time of performing the experiments we have set the development effort boundary as a percentage of the total development effort needed to include all the requirements in a software product. Then we have considered the 30, 50 and 70% of the total development effort, which respectively translates into an effort bound of 25, 43 and 60 effort units in our experiments. The change in the effort limit allows us to have, in a simple way, different instances of the problem on which to test the algorithms.

The second dataset has been generated randomly with 100 requirement, 5 customers and 44 requirements interactions, according to the NRP model (see Table 3).

**Table 2** Customers' relative importance

| Customers' weights | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ |
|---|---|---|---|---|---|
| For dataset 1 | 1 | 4 | 2 | 3 | 4 |
| For dataset 2 | 1 | 5 | 3 | 3 | 1 |

**Table 3** Dataset 2: assignment of the priority level of each requirement, requirements development effort and interactions

| | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ | $r_{10}$ | $r_{11}$ | $r_{12}$ | $r_{13}$ | $r_{14}$ | $r_{15}$ | $r_{16}$ | $r_{17}$ | $r_{18}$ | $r_{19}$ | $r_{20}$ | $r_{21}$ | $r_{22}$ | $r_{23}$ | $r_{24}$ | $r_{25}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 1 | 2 | 1 | 1 | 2 | 3 | 3 | 1 | 1 | 3 | 1 | 1 | 3 | 2 | 3 | 2 | 2 | 3 | 1 | 3 | 2 | 1 | 1 | 1 | 3 |
| $c_2$ | 3 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 3 | 3 | 2 | 1 | 3 | 2 | 3 | 3 | 1 | 3 | 3 | 3 | 2 | 3 |
| $c_3$ | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 3 | 2 | 2 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 2 | 3 | 3 | 2 | 1 | 2 | 3 | 2 |
| $c_4$ | 3 | 2 | 2 | 1 | 3 | 1 | 3 | 2 | 3 | 2 | 3 | 2 | 1 | 3 | 2 | 3 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 2 | 3 |
| $c_5$ | 1 | 2 | 3 | 1 | 3 | 1 | 2 | 3 | 1 | 1 | 2 | 2 | 3 | 1 | 2 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 3 | 3 | 3 |
| Eff | 16 | 19 | 16 | 7 | 19 | 15 | 8 | 10 | 6 | 18 | 15 | 12 | 16 | 20 | 9 | 4 | 16 | 2 | 9 | 3 | 2 | 10 | 4 | 2 | 7 |
| | $r_{26}$ | $r_{27}$ | $r_{28}$ | $r_{29}$ | $r_{30}$ | $r_{31}$ | $r_{32}$ | $r_{33}$ | $r_{34}$ | $r_{35}$ | $r_{36}$ | $r_{37}$ | $r_{38}$ | $r_{39}$ | $r_{40}$ | $r_{41}$ | $r_{42}$ | $r_{43}$ | $r_{44}$ | $r_{45}$ | $r_{46}$ | $r_{47}$ | $r_{48}$ | $r_{49}$ | $r_{50}$ |
| $c_1$ | 3 | 3 | 3 | 1 | 2 | 2 | 3 | 2 | 1 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 2 | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 3 |
| $c_2$ | 1 | 2 | 2 | 3 | 3 | 1 | 3 | 2 | 2 | 1 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 1 | 1 | 3 | 2 | 2 | 2 | 1 | 3 |
| $c_3$ | 3 | 3 | 1 | 3 | 3 | 3 | 2 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 3 | 3 | 3 | 3 | 1 | 3 | 2 |
| $c_4$ | 3 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 1 | 1 | 3 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 3 | 2 |
| $c_5$ | 2 | 2 | 3 | 2 | 3 | 1 | 1 | 3 | 3 | 2 | 2 | 1 | 1 | 2 | 1 | 3 | 1 | 1 | 2 | 1 | 2 | 3 | 3 | 2 | 2 |
| Eff | 15 | 8 | 20 | 9 | 11 | 5 | 1 | 17 | 6 | 2 | 16 | 8 | 12 | 18 | 5 | 6 | 14 | 15 | 20 | 14 | 9 | 16 | 6 | 6 | 6 |
| | $r_{51}$ | $r_{52}$ | $r_{53}$ | $r_{54}$ | $r_{55}$ | $r_{56}$ | $r_{57}$ | $r_{58}$ | $r_{59}$ | $r_{60}$ | $r_{61}$ | $r_{62}$ | $r_{63}$ | $r_{64}$ | $r_{65}$ | $r_{66}$ | $r_{67}$ | $r_{68}$ | $r_{69}$ | $r_{70}$ | $r_{71}$ | $r_{72}$ | $r_{73}$ | $r_{74}$ | $r_{75}$ |
| $c_1$ | 3 | 3 | 1 | 3 | 2 | 1 | 3 | 1 | 3 | 1 | 2 | 2 | 3 | 3 | 1 | 3 | 1 | 3 | 2 | 3 | 1 | 3 | 2 | 3 | 1 |
| $c_2$ | 3 | 3 | 1 | 2 | 2 | 3 | 3 | 2 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | 2 | 1 | 2 | 3 | 1 | 1 | 3 | 1 | 3 | 2 |
| $c_3$ | 3 | 1 | 2 | 3 | 2 | 3 | 2 | 1 | 2 | 3 | 1 | 1 | 2 | 3 | 3 | 1 | 3 | 3 | 3 | 1 | 3 | 1 | 3 | 1 | 1 |
| $c_4$ | 2 | 1 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 1 | 2 | 1 | 2 | 1 | 2 | 3 | 3 | 2 | 2 |
| $c_5$ | 1 | 3 | 3 | 2 | 3 | 1 | 2 | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 2 | 1 |
| Eff | 6 | 2 | 17 | 8 | 1 | 3 | 14 | 16 | 18 | 7 | 10 | 7 | 16 | 19 | 17 | 15 | 11 | 8 | 20 | 1 | 5 | 8 | 3 | 15 | 4 |

**Table 3** (continued)

| | $r_{76}$ | $r_{77}$ | $r_{78}$ | $r_{79}$ | $r_{80}$ | $r_{81}$ | $r_{82}$ | $r_{83}$ | $r_{84}$ | $r_{85}$ | $r_{86}$ | $r_{87}$ | $r_{88}$ | $r_{89}$ | $r_{90}$ | $r_{91}$ | $r_{92}$ | $r_{93}$ | $r_{94}$ | $r_{95}$ | $r_{96}$ | $r_{97}$ | $r_{98}$ | $r_{99}$ | $r_{100}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 1 | 2 | 3 | 3 | 1 | 2 | 1 | 3 | 1 | 2 | 2 | 2 | 1 | 3 | 2 | 2 | 3 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 |
| $c_2$ | 1 | 3 | 3 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 1 | 3 | 1 | 1 |
| $c_3$ | 2 | 3 | 3 | 1 | 2 | 1 | 2 | 3 | 2 | 3 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 3 | 2 | 3 |
| $c_4$ | 2 | 1 | 3 | 3 | 1 | 3 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 3 | 1 | 1 | 3 | 3 | 1 | 2 | 1 | 2 | 3 | 1 | 3 |
| $c_5$ | 3 | 2 | 3 | 1 | 3 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 3 | 3 | 3 | 1 | 1 | 3 | 1 | 3 | 3 | 3 | 3 | 3 |
| Eff | 20 | 10 | 20 | 3 | 20 | 10 | 16 | 19 | 3 | 12 | 16 | 15 | 1 | 6 | 7 | 15 | 18 | 4 | 7 | 2 | 7 | 8 | 7 | 7 | 3 |

$$r_{21} \odot r_{22}\ r_{32} \odot r_{33}\ r_{46} \odot r_{47}\ r_{65} \odot r_{66}$$

$$r_2 \Rightarrow r_{24}\ r_3 \Rightarrow r_{26}\ r_3 \Rightarrow r_{27}\ r_3 \Rightarrow r_{28}\ r_3 \Rightarrow r_{29}\ r_4 \Rightarrow r_5\ r_6 \Rightarrow r_7\ r_7 \Rightarrow r_{30}\ r_{10} \Rightarrow r_{32}$$

$$r_{10} \Rightarrow r_{33}\ r_{14} \Rightarrow r_{32}\ r_{14} \Rightarrow r_{34}\ r_{14} \Rightarrow r_{37}\ r_{14} \Rightarrow r_{38}\ r_{16} \Rightarrow r_{39}\ r_{16} \Rightarrow r_{40}\ r_{17} \Rightarrow r_{43}$$

$$r_{29} \Rightarrow r_{49}\ r_{29} \Rightarrow r_{50}\ r_{29} \Rightarrow r_{51}\ r_{30} \Rightarrow r_{52}\ r_{30} \Rightarrow r_{53}\ r_{31} \Rightarrow r_{55}\ r_{32} \Rightarrow r_{56}\ r_{32} \Rightarrow r_{57}$$

$$r_{33} \Rightarrow r_{58}\ r_{36} \Rightarrow r_{61}\ r_{39} \Rightarrow r_{63}\ r_{40} \Rightarrow r_{64}\ r_{43} \Rightarrow r_{65}\ r_{46} \Rightarrow r_{68}\ r_{47} \Rightarrow r_{70}\ r_{55} \Rightarrow r_{79}$$

$$r_{56} \Rightarrow r_{80}\ r_{57} \Rightarrow r_{80}\ r_{62} \Rightarrow r_{83}\ r_{62} \Rightarrow r_{84}\ r_{64} \Rightarrow r_{87}$$

The relative importance of the customers is given in the second row of Table 2. This dataset was defined because in real agile software projects development, in the initial timeboxes, we are faced with the problem of selecting requirements from a wider set. Therefore, the number of requirements has been incremented from 20 to 100. The development effort values of each requirement are given in the 1 to 20 range. We had fixed 20 units (4 weeks) as the maximum development effort for a requirement, considering the timebox limit defined in agile methods (e.g. Scrum proposes iteration in the range 2 to 4 weeks). The customers values of level of priority of requirements are in the range of 1 to 3, because when customers have to make an assignment of the benefit that will imply the inclusion of a given requirement, they prefer to use a coarse grained scale instead of one of finer granularity. Usually, they simply place requirements in one of three categories: inessential (1), desirable (2) or mandatory (3) (Wiegers 2003; Simmons 2004).

Following the same considerations made on dataset 1, we have set the development effort boundary using the same percentages (30, 50 and 70 %) of the total development effort needed to include all the requirements in a software product, which respectively translates into an effort bound of 312, 519 and 726 effort units in our experiments. So, we will test ACS using six instances of NRP.

### 4.2 Metaheuristic Techniques Applied in Experimentation

Many metaheuristic techniques have been applied to the requirement selection problem, a review of them can be found at del Sagrado et al. (2010b). We select two algorithms against with our multi-objective ant colony system will be evaluated for solving the NRP: Greedy Randomized Adaptive Search Procedure (GRASP) and Non-dominated Sorting Genetic Algorithm (NSGA-II).

The first one is a metaheuristic algorithm that generates a good approximation to the efficient set of solutions of a multi-objective combinatorial optimization problem (Vianna and Arroyo 2004). GRASP was first introduced by Feo and Resende (1989). Survey papers on GRASP include Feo and Resende (1995), Pitsoulis and Resende (2003), and Resende and Ribeiro (2003). This algorithm proceeds iteratively by first building a greedy randomized solution and then improving it through a local search. The greedy randomized solution is built from a list of elements ranked by a greedy function by adding elements to the solution set of the problem. The greedy function is in charge of measuring the profit of including an element in the solution with respect to the cost of its inclusion. In our approach, the greedy function used measures the quality of a requirement based on users satisfaction with respect to the effort (i.e. $s_i/e_i$) according to the following ratio:

$$\frac{\lambda_s * s_i + \frac{\lambda_e}{e_i}}{2 * e_i} \tag{16}$$

where $\lambda_s$ and $\lambda_e$ are the weights associated with each objective as defined in (9). This measure is a productivity metric that weighs the profit of including a requirement with respect to the resources involved in their development. The local search acts, in an iterative way, trying to replace the current solution by a better one located in its neighbourhood. GRASP terminates when no better solution can be found in the neighbourhood.

The fast Elitist Non-dominated Sorting Genetic algorithm (NSGA-II) was proposed by Deb et al. (2002). The word elitist refers to the fact that only the best individuals found so far are transferred to the next population. During each generation, a population is constructed by combining the parent and the child population. Individuals represent the possible solutions, that is, an individual is a set of requirements. Each individual is evaluated with respect to the two objective functions defined in (4), where the total effort is multiplied by $-1$ in order to transform the second objective into a maximization problem, so that the NSGA-II algorithm will try to maximize each one of them. Each generation represents the evolution of the population. The idea is that by means of crossover and mutation the new children and mutated individuals have even better fitness values than the original ones. Better individuals have a higher probability of stating at the Pareto front, whose cardinality is limited by the population defined in the execution. In the case of NRP, the crossover and mutation methods are more specific and difficult than in other problems because we need to take into account the resources bound in order to obtain new valid individuals (see del Sagrado et al. 2010a). The works of Durillo et al. (2011) and Zhang and Harman (2010) show that NSGA-II can solve NRP offering a set of comparable solutions with those obtained by other metaheuristics.

### 4.3 Performance Measures

We have calculated the optimal Pareto front of the problem defined by the first dataset, in order to compare it against those calculated by the metaheuristic optimization techniques. However, when the number of requirements grows, dataset 2, obtaining the optimal Pareto front becomes an intractable problem. So, our approach here is to compare the results using several quality indicators, giving an insight on the quality of the results achieved for all the executed algorithms. Therefore, we use the following indicators with the purpose of conducting a comparative measure of diversity and convergence of the solutions obtained:

- The number of non dominated solutions found (#Solutions). Pareto fronts with a higher number of non dominated solutions are preferred.
- The size of the space covered by the set on non-dominated solutions found (Hypervolume) (Zitzler and Thiele 1999). For a two dimensional problem, for each solution $i \in Q$, a vector $v_i$ is built with respect to a reference point $W$, and the solution $i$ is considered as the diagonal corner of an hypercube. Hypervolume is the volume occupied by the union of all of these hypercubes:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{17}$$

  Pareto fronts whit higher hypervolume values are preferred.
- The extent of spread achieved among the obtained solutions ($\Delta$-Spread) (Durillo et al. 2009).

$$\Delta(F) = \frac{d_f + d_l + \sum_{i=1}^{n-1} |d_i - \bar{d}|}{d_f + d_l + (n-1) * \bar{d}} \tag{18}$$

where the Euclidean distance between two consecutive solutions $\hat{\mathbf{R}}_i$ and $\hat{\mathbf{R}}_{i+1}$, $d_i$ is computed as

$$d_i = \sqrt{(sat(\hat{\mathbf{R}}_{i+1}) - sat(\hat{\mathbf{R}}_i))^2 + (eff(\hat{\mathbf{R}}_{i+1}) - eff(\hat{\mathbf{R}}_i))^2} \tag{19}$$

$\bar{d}$ is the average of these distances, $d_f, d_l$ are, respectively, the Euclidean distances of the first, $\hat{\mathbf{R}}_1$, and last, $\hat{\mathbf{R}}_n$, obtained solutions to the extreme solutions $\hat{\mathbf{R}}_f$ and $\hat{\mathbf{R}}_l$ of the optimal Pareto front, and $n$ is the number of solutions in the obtained Pareto front $F$. Pareto fronts with smaller spread are preferred.

– A measure that evaluates the uniformity of the distribution of non-dominated solutions found (Spacing) (Schott 1995). If the problem has $N$ objectives and its Pareto front has $n$ solutions, the spacing of $F$ is defined as:

$$Spacing(F) = \frac{\sum_{j=1}^{n} \left( \sum_{i=1}^{N} \left( 1 - \frac{|d_{ij}|}{\bar{d}_i} \right)^2 \right)^{1/2}}{n * N} \tag{20}$$

where, $\bar{d}_i$ is the mean value of the magnitude of the $i - th$ objective in the set $F$ and $d_{ij}$ is the value of the $i$-th objective for the $j$-th solution in $F$. Pareto fronts with higher spacing are preferred.

Pareto fronts with a higher number of non-dominated solutions, smaller spread, higher spacing and higher hypervolume are preferred. The average and standard deviation of all of these quality indicators are computed.

### 4.4 Comparison Methodology

In order explore the applicability of multi-objective ACO for solving NRP, we will compare its performance on the datasets with that of the multi-objective metaheuristics approaches selected. All algorithms are executed for a maximum of 10,000 fitness functions evaluations. Taking this fact into account, it is worth to note that, for the parameters setting we have used default values to explore different search behaviours in the algorithms. This choice, as Arcuri and Fraser (2013) points out, "is a reasonable and justified one, whereas parameter tuning is a long and expensive process that might or might not pay off in the end". Thus, the comparison of the results is done in four steps:

1. For each dataset and each development effort bound, we have performed 100 consecutive executions of each of the metaheuristic search techniques, using different parameters settings representative of the different behaviours that the algorithms can exhibit. We compute the average (as measure of central tendency) and standard deviation (as measure of statistical dispersion) of the number of solutions in the Pareto-front, of the quality indicators, and of the execution time.
2. With these values at hand, we have analyzed each technique separately. The scores of the five quality indicators are ranked, and we select the parameters setting that exhibits the best average ranking.
3. Once a parameter setting has been established for each one of the algorithms (GRASP, NSGA-II and ACS), the average values of their indicators are visually

compared, using Kiviat graphs. This step allows us evaluate the goodness of our ACS proposal, comparing it against the other two search methods used.

4. The Pareto fronts returned by each of the metaheuristic search techniques are compared using the non-parametric significance Mann-Whitney U test, which is a non-parametric statistical hypothesis test for assessing whether one of two samples of independent observations tends to have larger values than the other. The comparison is made based on the values computed for the productivity metric given by

$$Productivity(\hat{\mathbf{R}}) = \frac{sat(\hat{\mathbf{R}})}{eff(\hat{\mathbf{R}})} \tag{21}$$

for each solution $\hat{\mathbf{R}}$ in the Pareto front. This metric give us an insight of the profit returned by the solutions according to the software development process.

## 5 Analysis of the Experimental Results

Tables 8, 9 and 10 included in the Appendix collect the results of the experiments returned by GRASP, NSGA-II and ACS for the different datasets, requirements interactions and effort bounds used. We have highlighted with a star symbol the best values of the quality indicators and the parameters setting with the best average ranking. Next, we analyze these results following the comparison methodology previously exposed.

The GRASP algorithm receives as input parameters the number of iterations (each iteration consists of two phases: construction and local search) and a number in the 0 to 1 range that controls the amount of greediness and randomness (values close to zero favors a greedy behaviour, whereas values close to one favor a random behaviour). In most of the problems considered the value 0.9, that favor a random behaviour to explore wider areas, obtains the best results. The values obtained by GRASP for the quality indicators in the different datasets are shown in Table 8.

The execution time of NSGA-II depends on the number of generations developed. We look for a balance in the number of function evaluations performed by the algorithm, by adjusting the size of the populations and the number of iterations. So,

**Table 4** Dataset 1 best average results for each algorithm

| | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 30 % | | | | | |
| GRASP | 11.37±1.47 | 5,851.00±277.82 | 0.64±0.09 | ⋆0.36±0.03 | ⋆362.80±10.84 |
| NSGA-II | 9.69±2.09 | 6,842.92±849.03 | 0.76±0.09 | 0.29±0.11 | 1,891.55±196.10 |
| ACS | ⋆13.66±13.66 | ⋆7,805.39±49.87 | ⋆0.52±0.03 | 0.33±0.01 | 639.10±17.79 |
| Dataset 1 50 % | | | | | |
| GRASP | 17.65±2.22 | 14,508.20±265.88 | 0.73±0.07 | 0.35±0.03 | 1,208.25±29.15 |
| NSGA-II | 11.30±1.82 | 15,676.77±1,214.25 | 0.79±0.07 | 0.27±0.06 | 1,980.93±144.42 |
| ACS | ⋆17.75±0.61 | ⋆18,153.33±51.26 | ⋆0.52±0.01 | ⋆0.37±0.01 | ⋆787.62±33.29 |
| Dataset 1 70 % | | | | | |
| GRASP | 20.26±2.18 | 24,473.66±377.05 | 0.69±0.06 | 0.34±0.03 | 839.84±31.48 |
| NSGA-II | 11.70±1.90 | 24,408.67±1,746.18 | 0.80±0.07 | 0.26±0.05 | 2,034.25±120.81 |
| ACS | ⋆20.57±20.57 | ⋆29,196.12±53.72 | ⋆0.48±0.02 | ⋆0.40±0.01 | ⋆836.76±34.35 |

**Table 5** Dataset 2 best average results for each algorithm

| | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (s) |
|---|---|---|---|---|---|
| Dataset 2 30 % | | | | | |
| GRASP | ⋆57.99±3.66 | 112,418.88±235.53 | ⋆0.60±0.04 | ⋆0.41±0.02 | 28.92±0.37 |
| NSGA-II | 54.34±8.51 | 218,138.21±6,861.82 | 0.80±0.07 | 0.22±0.05 | ⋆28.13±1,28 |
| ACS | 47.12±5.44 | ⋆234,583.42±1,710.00 | 0.68±0.06 | ⋆0.41±0.04 | 616.87±21.98 |
| Dataset 2 50 % | | | | | |
| GRASP | ⋆75.81±5.81 | 425,642.47±1,894.97 | 0.74±0.04 | 0.32±0.03 | 118.34±2.00 |
| NSGA-II | 65.54±11.86 | 495,948.91±14,310.53 | 0.81±0.06 | 0.19±0.03 | ⋆35.05±0.84 |
| ACS | 57.68±5.69 | ⋆527,685.22±2,738.50 | ⋆0.66±0.06 | ⋆0.43±0.04 | 770.22±37.33 |
| Dataset 2 70 % | | | | | |
| GRASP | ⋆120.14±7.27 | 769,613.34±2,064.60 | 0.70±0.03 | 0.29±0.02 | 324.61±13.24 |
| NSGA-II | 83.32±10.52 | 873,383.61±24,556.01 | 0.77±0.05 | 0.19±0.03 | ⋆38.30±0.79 |
| ACS | 70.98±5.27 | ⋆902,769.29±3,141.76 | ⋆0.61±0.06 | ⋆0.45±0.03 | 881.95±54.19 |

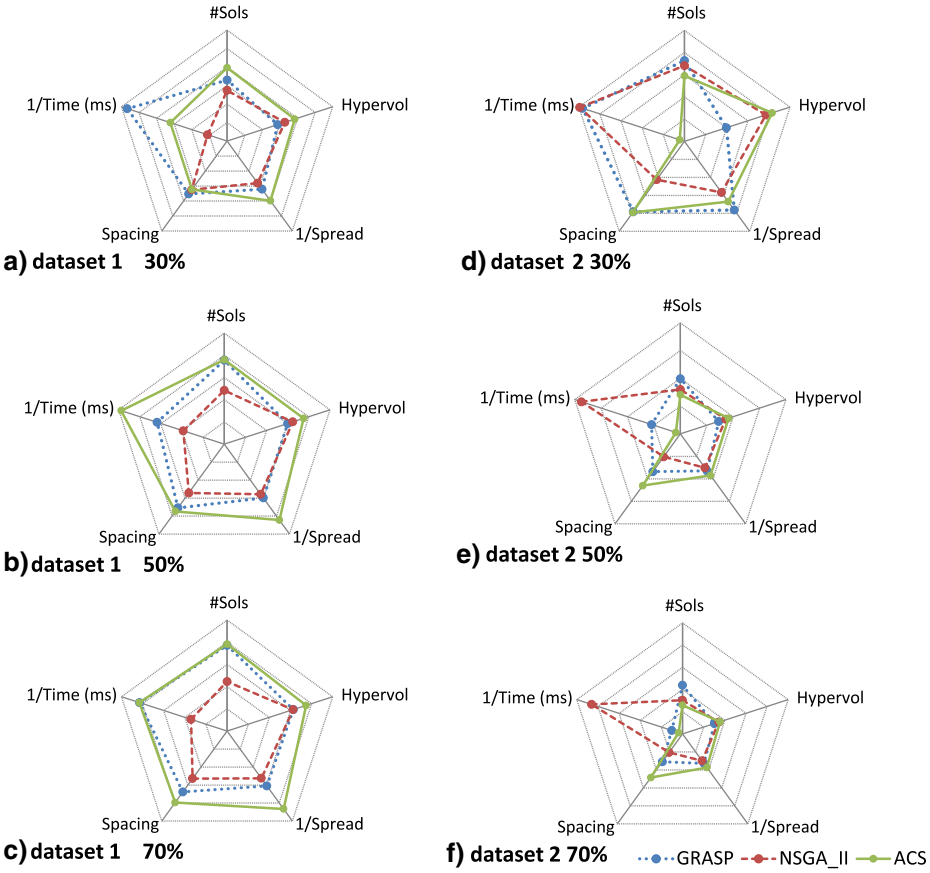



**Fig. 4** Kiviat graph comparing metaheuristic algorithms

in our tests, the size of the population have been fixed to 20 and 40 in dataset 1, and 100 and 125 in dataset 2, keeping the number of functions evaluation at 10,000. The mutation probability is kept fixed at a value equal to $1/n$, where $n$ is the number of requirements (i.e. 0.05 for dataset 1, and 0.01 for dataset 2) and we use a high crossover probability, assigning it a value in $\{0.8, 0.9\}$, to favor recombination of individuals. Best values have been obtained when using a higher population size with the lowest crossover probability (i.e. 40 individuals and 0.8) in the case of dataset 1 and the highest crossover probability (i.e. 125 individuals and 0.9) in dataset 2. Table 9 contains the quality indicators results obtained by NSGA-II in the different datasets.

The number of ants in the colony has a direct impact on execution time of ACS algorithm. We use as number of ants $n/2$, where $n$ is the number of requirements. We fix the pheromone evaporation rate $\rho = 0.01$ (we also set the pheromone decay coefficient $\varphi$ to 0.01) and the parameter $q_0 = 0.95$, favouring exploration. For the $(\alpha, \beta)$ pair of parameters, we have used different combinations (i.e. (0,1) (1,0) (1,1) (1, 2) (1,5)), reflecting the importance of using local information (i.e. pheromone) with respect to heuristic information in the search. We highlight that the best average

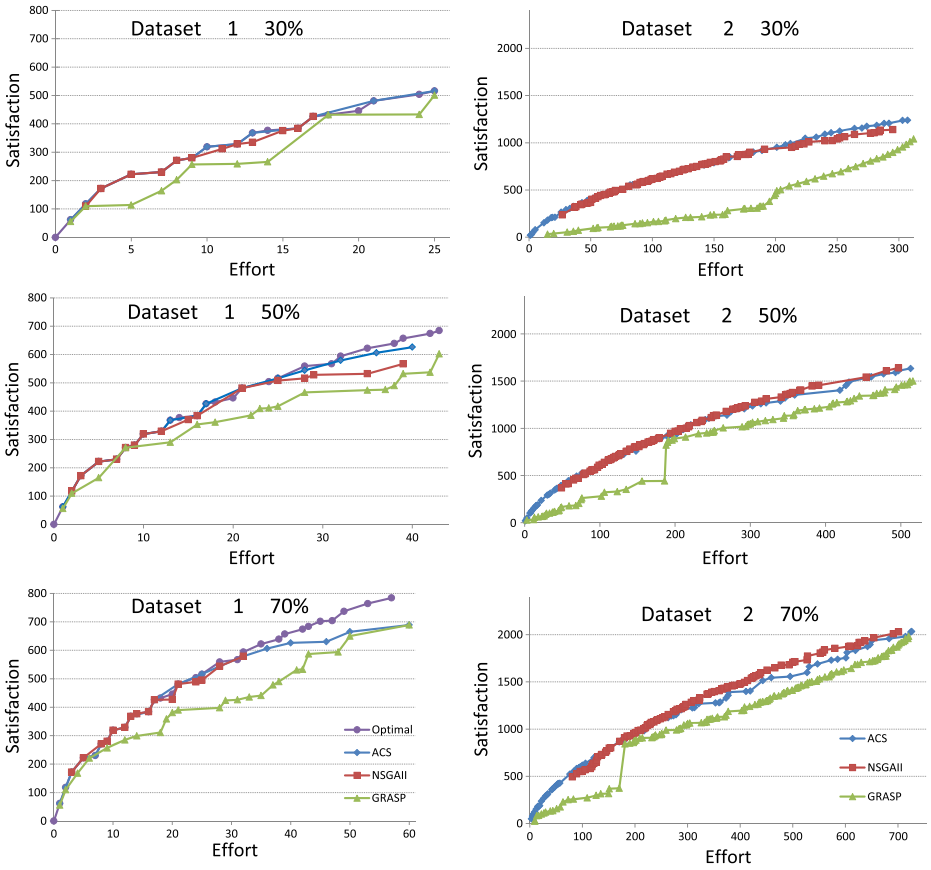



**Fig. 5** Pareto fronts for datasets 1 and 2

values for the ranking of quality indicators are obtained by ACS (see Table 10) when both local and heuristic information are used in the search, but given more importance to heuristic information than to pheromone, that is, $\alpha \geq 1, \alpha \leq \beta$.

Once, we have selected for each algorithm, a parameters setting taking into account the best average ranking of the quality indicators values obtained (see Tables 4 and 5), we proceed to make a visual comparison using the Kiviat graphs. Figure 4 shows Kiviat graphs depicting the quality indicators dataset and each development effort. With respect to dataset 1 (see Fig. 4a–c) ACS obtains the best values, except in spacing and execution time in one case (see, Fig. 4a). The differences between ACS and GRASP are mainly present in the hypervolume and spread, but both have similar execution times. NSGA-II and GRASP obtain Pareto fronts with similar hypervolume and spread values, but NSGA-II spends more execution time than the others. In the case of dataset 2 (see Fig. 4d–f), ACS obtains the best values in hypervolume, spacing and spread, although GRASP obtains a higher number of solutions but the non-dominated solutions found cover less space than the Pareto fronts built by ACS and NSGA-II which have similar hypervolume values. However, we detect a significantly worse execution time in ACS.

**Table 6** Mann-Whitney-Wilcoxon test for dataset 1

| | Optimal | | |
|---|---|---|---|
| | U=145.5 $n_1$=18 $n_2$=14 | | |
| ACS | $p_{value} = 4.6 * 10^{-1}$ | ACS | |
| | U=125.5 $n_1$=18 $n_2$=12 | U=86 $n_1$=14 $n_2$=12 | |
| NSGA-II | $p_{value} = 4.63 * 10^{-1}$ | $p_{value} 9.4 * 10^{-1}$ | NSGA-II |
| | U=137 $n_1$=18 $n_2$=11 | U=115.5 $n_1$=14 $n_2$= 11 | U=101 $n_1$=12 $n_2$=11 |
| GRASP | $p_{value} = 9.21 * 10^{-2}$ | $p_{value} = 3.32 * 10^{-2}$ | $p_{value} = 3.17 * 10^{-2}$ |

Results for the comparison of the Pareto-fronts obtained for dataset 1 with a development effort bound of the 30% of the total software development resources

| | Optimal | | |
|---|---|---|---|
| | U=272.5 $n_1$= 26 $n_2$=18 | | |
| ACS | $p_{value} = 3.62 * 10^{-1}$ | ACS | |
| | U=224.5 $n_1$=26 $n_2$=16 | U=156 $n_1$=18 $n_2$=16 | |
| NSGA-II | $p_{value} = 6.72 * 10^{-1}$ | $p_{value} = 6.96 * 10^{-1}$ | NSGA-II |
| | U=333.5 $n_1$=26 $n_2$=18 | U=245.5 $n_1$=18 $n_2$=18 | U=211.5 $n_1$=18 $n_2$=16 |
| GRASP | $p_{value} = 1.67 * 10^{-2}$ | $p_{value} = 7.07 * 10^{-3}$ | $p_{value} = 1.86 * 10^{-2}$ |

Results for the comparison of the Pareto-fronts obtained for dataset 1 with a development effort bound of the 50% of the total software development resources

| | Optimal | | |
|---|---|---|---|
| | U=362.5 $n_1$=31 $n_2$=21 | | |
| ACS | $p_{value} = 4.93 * 10^{-1}$ | ACS | |
| | U=311 $n_1$=31 $n_2$=16 | U=178 $n_1$=21 $n_2$=16 | |
| NSGA-II | $p_{value} = 1.62 * 10^{-1}$ | $p_{value} = 7.73 * 10^{-1}$ | NSGA-II |
| | U=531 $n_1$=31 $n_2$=24 | U=361.5 $n_1$=24 $n_2$=21 | U=301 $n_1$=24 $n_2$=16 |
| GRASP | $p_{value} = 6.43 * 10^{-3}$ | $p_{value} = 1.18 * 10^{-2}$ | $p_{value} = 2.17 * 10^{-3}$ |

Results for the comparison of the Pareto-fronts obtained for dataset 1 with a development effort bound of the 70% of the total software development resources

| How explain the test, When... | |
|---|---|
| $P < 0.001$ | The difference between the two samples is highly significant |
| $P < 0.01$ | Two samples are significantly different |
| $P < 0.05$ | The difference between the two samples is marginally significant |
| $P >= 0.05$ | Two samples are not significantly different |

Figure 5 represents a selected Pareto front according to the quality measures, for each algorithm and each problem. For the first dataset we have also obtained and depicted the Pareto optimal front. ACS and NSGA-II have very few differences compared to the optimum. However GRASP presents large differences but with fronts containing the greater number of solutions, this same trend is shown in the second dataset.

Our analysis concludes performing a Mann-Whitney U test on the values for the productivity metric (see (21)) of each solution in the Pareto fronts obtained. Its results are shown in Tables 6 and 7. In all cases the Pareto fronts obtained by GRASP show differences with those obtained by ACS and NSGA-II. For dataset 1 with effort bounds of 30 and 50 %, these differences are marginally significant (P< 0.05), but are significantly different (P<0.01) when considering the 70 % effort bound and highly significant (P<0.001) for dataset 2 in all cases. With respect to the optimal Pareto front in the dataset 1, GRASP exhibits not significant differences for 30 %, marginally significant differences (P< 0.05) for 50 % and significantly differences (P<0.01) for 70 % efforts bounds, respectively. Whereas fronts coming from the algorithms ACS and NSGA-II are not significantly different (P≥0.05), with the exception of dataset 2 with a 70 % effort bound which exhibits marginally significant differences (P< 0.05). There are no significant differences between the optimal Pareto front and those obtained by ACS and NSGA-II. These results confirm the graphical representation of the selected Pareto fronts (see Fig. 5).

**Table 7** Mann-Whitney-Wilcoxon test for dataset 2

| | ACS | |
|---|---|---|
| NSGA-II | $U = 2552, n_1 = 76, n_2 = 59$<br>$p_{value} = 1.16 * 10^{-1}$ | NSGA-II |
| GRASP | $U = 3776, n_1 = 64, n_2 = 59$<br>$p_{value} =< 2 * 10^{-6}$ | $U = 4864, n_1 = 76, n_2 = 64$<br>$p_{value} =< 2 * 10^{-6}$ |

Results for the Pareto-fronts obtained for dataset 2 with a development effort bound of 30%

| | ACS | |
|---|---|---|
| NSGA-II | $U = 3557, n_1 = 82, n_2 = 75$<br>$p_{value} = 3.6 * 10^{-3}$ | NSGA-II |
| GRASP | $U = 5889, n_1 = 84, n_2 = 75$<br>$p_{value} =< 2 * 10^{-6}$ | $U = 6555, n_1 = 84, n_2 = 82$<br>$p_{value} =< 2 * 10^{-6}$ |

Results for the Pareto-fronts obtained for dataset 2 with a development effort bound of 50%

| | ACS | |
|---|---|---|
| NSGA-II | $U = 4576, n_1 = 96, n_2 = 79$<br>$p_{value} = 3.48 * 10^{-4}$ | NSGA-II |
| GRASP | $U = 9220, n_1 = 134, n_2 = 79$<br>$p_{value} =< 2 * 10^{-6}$ | $U = 11360, n_1 = 134, n_2 = 96$<br>$p_{value} =< 2 * 10^{-6}$ |

Results for the Pareto-fronts obtained for dataset 2 with a development effort bound of 70%

| When... | |
|---|---|
| $P < 0.001$ | The difference between the two samples is highly significant |
| $P < 0.01$ | Two samples are significantly different |
| $P < 0.05$ | The difference between the two samples is marginally significant |
| $P >= 0.05$ | Two samples are not significantly different |

## 6 Threats to Validity

The results obtained in this research work are subject to the limitations which are inherent to any SBSE empirical study. It is worth to discuss the validity of the results, in order to provide a complete understanding of limitations and extent that the work has. In the literature (Wohlin et al. 2012; Ali et al. 2010), we can find different ways to classify aspects of validity and threats to validity. We are going to follow the types of threats proposed by Ali et al. (2010) that comprise four aspects of the validity: construct validity, internal validity, conclusion validity and external validity.

*Construct validity* reflects if the measures and scales used capture properly the concepts they need to represent. In this paper, the objects studied are set of requirements and the associated data are estimates provided by software engineers or customers (e.g cost and value added estimates). This fact leads to one possible validity threat due to subjectivity and accuracy (i.e. in real software development projects, the values associated to requirements are gathered as an answer from a closed set to a simple statement). Nonetheless, development teams use estimations in every day work both for exploring tradeoffs in release planning and decision-making. Related to the effectiveness measures we use quality measures related to the diversity and convergence of the sets of requirements selected. Specifically, the number of solutions, hypervolume, spread and spacing measures refer to the quality of the Pareto fronts obtained by the different search-based metaheuristic techniques applied, whilst the time expended refers to performance.

*Internal validity* regards to the analysis of the performance of the metaheuristic search techniques applied (i.e. parameter settings and biased selection of datasets that can favor a certain technique). In our experiments we have used default parameter settings with the idea of preserving the inner nature of the metaheuristic techniques and exploring different search behaviours in the algorithms. With respect to the quality of the problem instances we have used two datasets, probably a more significant number of datasets seems necessary, but, typically, real world datasets are considered confidential by the companies that own them. So, we have preferred to extend the available dataset by changing the effort limits, as others authors (Jiang et al. 2010) do, instead of generate and employ a greater number synthetic datasets. We think that, in this way, a biased selection in favor of a certain technique is more or less avoided.

*Conclusion validity* is concerned with the relationship between approach and outcome, ensuring that there is a right statistical relationship between them. In the case of our paper, the proposed multi-objective ACO approach automates the search for optimal or near-optimal sets of requirements, within a given effort bound, that balance stakeholders' priorities while keeping requirements interactions. As conclusion the evidence suggest that this approach is feasible. This conclusion is based on the statistical comparison of the experiments carried out on the datasets. In order to avoid the randomness all experiments were executed several times, so we have performed 100 consecutive executions on the same hardware platform. The variables measured are the quality indicators showing their average and dispersion values. As baseline for comparison, we have used two metaheuristic techniques (i.e. GRASP and NSGA-II) that have a very different nature and we believe that are sufficiently representative and appropriate. By other hand, the Pareto fronts returned are assessed in pairs using the non-parametric significance Mann-Whitney

U test on a productivity metric of the solutions from the point of view of the software project. The significance level chosen was 95 % looking for a compromise in the adoption of a technique between its performance and robustness. The results obtained suggest that our multi-objective ACO is a good enough search method that offers software developers, who must decide what is the best set of requirements (according to business strategies) that has to be considered in the next software release, several alternatives among the whole set of candidate requirements proposed by customers.

*External validity* deals with the generalization of observed results. For our study, this is a major threat to validity. First, as greater the complexity of a real software project, greater the ability for generalization. But real software projects are difficult to obtain due to the privacy policies followed by software development companies. Second, the generalization domain of our study is wide, as is the domain of human activities in which software is present. Thus, the set of stakeholders and requirements for a software project focused on the development of a computer game is significantly different from that of an accounting software system. As consequence, it is difficult to estimate, using only the results of this study, the extent in which our proposed ACS algorithm can help software development teams. However, we believe that for any software developer is considerably helpful to have these techniques available for determining which candidate requirements of a software product should be included in a certain release.

## 7 Conclusions and Further Works

In this paper, we study the applicability of multi-objective Ant Colony Optimization algorithms in the field of Software Engineering, specifically within the field of analysis and study of requirements for a software project. ACO has already been applied in other fields like testing, but not to requirements from the multi-objective perspective.

Our paper formally defines the Next Release Problem including interactions between requirements. As value added, we propose a method to reduce all the interactions of NRP in order to manage the problem as a basic instance of NRP, facilitating the execution of metaheuristics that do not take them into account. We develop our own multi-objective ant colony system (ACS) for finding a non-dominated solution set for NRP considering functional requirements interactions, and apply two other techniques (GRASP, NSGA-II) in order to evaluate our system.

GRASP has shown a bad performance in our experiments with NRP, but it will be useful to examine how it behaves when using other greedy functions (different from the one we have proposed) designed specifically for this problem. During the adaptation of NSGA-II to tackle NRP we have found several difficulties with crossover and mutation operations due to the restrictions imposed by NRP. These restrictions have force us to develop a repair operator, in order to ensure that the solutions built in an evolutionary way satisfy the constraints imposed by effort bound and requirements interactions. Our multi-objective ACS obtains non dominated solution sets slightly better than those found by NSGA-II, and GRASP. ACS can be applied efficiently to solve NRP, in order to obtain the Pareto front that allows

software developers to take decisions about the set of features that must to be included in the next software release.

The use of metaheuristic techniques constitute a valuable aid for experts who must decide what is the set of requirements that has to be considered in the next development stages when they face to contradictory goals. Our approach combine computational intelligence and the knowledge experience of the human experts with the idea of obtaining a better requirements selection than that produced by expert developer's judgment alone. The actual approaches in Requirements Engineering usually are assisted by requirements management tools. These tools assist the development team in the management of each iteration. Here is where we think that our proposal algorithm can be useful as an aid, by its inclusion as a new facility.

As future lines of work we plan to study the quality of the solution sets found and improve the model of requirements selection between increments in a software development project, that is to say, consider human and dynamic aspects. By other hand we are developing an experience, where novel software developers (students) solve NRP alone, and next they perform the same tasks with the aid and support of metaheuristic techniques, with the objective of evaluating human competitiveness.

**Acknowledgements** This research has been funded by the Spanish Ministry of Education, Culture and Sport under project TIN2010-20900-C04-02. We wish also thank to anonymous reviewers for their useful comments.

## Appendix: Experiments results

**Table 8** Results of Greedy Randomized Adaptive Search Procedure (GRASP)

| Parameters (iter, $\alpha$) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 30 % | | | | | |
| ⋆(1000,0.9) | 11.37±1.47 | ⋆5,851.00±277.82 | ⋆0.64±0.09 | 0.36±0.03 | 362.80±10.84 |
| (1000,0.8) | ⋆11.67±1.60 | 5,239.29±337.86 | 0.71±0.09 | ⋆0.40±0.03 | 356.41±11.28 |
| (1000,0.5) | 6.72±1.03 | 3,571.45±526.58 | 0.75±0.10 | 0.35±0.03 | 340.61±10.12 |
| (1000,0.3) | 6.00±0.00 | 1,487.00±0.00 | 0.65±0.00 | 0.31±0.00 | 339.06±11.66 |
| (500,0.9) | 10.26±1.60 | 5,386.75±318.31 | 0.65±0.09 | 0.38±0.03 | 183.77±10.46 |
| (500,0.8) | 11.12±1.48 | 4,764.96±407.39 | 0.72±0.09 | ⋆0.40±0.03 | 180.64±11.42 |
| (500,0.5) | 6.56±1.00 | 3,125.67±471.71 | 0.82±0.07 | 0.35±0.04 | ⋆169.54±9.28 |
| (500,0.3) | 6.00±0.00 | 1,487.00±0.00 | 0.65±0.00 | 0.31±0.00 | 169.99±9.75 |
| Dataset 1 50 % | | | | | |
| ⋆(1000,0.9) | ⋆17.65±2.22 | ⋆14,508.20±265.88 | ⋆0.73±0.07 | 0.35±0.03 | 1,208.25±29.15 |
| (1000,0.8) | 17.33±2.04 | 13,730.69±365.03 | 0.77±0.06 | ⋆0.40±0.03 | 1,202.66±24.38 |
| (1000,0.5) | 14.96±0.60 | 11,268.44±175.26 | 0.78±0.04 | 0.27±0.02 | 1,175.61±23.76 |
| (1000,0.3) | 14.00±0.00 | 9,847.00±0.00 | 0.80±0.00 | 0.23±0.00 | 1,162.34±23.35 |
| (500,0.9) | 16.99±2.10 | 13,932.90±368.64 | 0.74±0.07 | 0.35±0.03 | 605.58±22.47 |
| (500,0.8) | 16.66±2.01 | 13,275.39±477.28 | 0.77±0.07 | 0.40±0.03 | 599.04±19.03 |
| (500,0.5) | 14.64±0.63 | 11,122.17±184.16 | 0.79±0.03 | 0.26±0.02 | 584.37±16.30 |
| (500,0.3) | 14.00±0.00 | 9,847.00±0.00 | 0.80±0.00 | 0.23±0.00 | ⋆580.63±18.71 |

**Table 8** (continued)

| Parameters (iter, $\alpha$) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 70 % | | | | | |
| (1000,0.3) | ⋆20.97±2.16 | ⋆24,837.24±297.55 | ⋆0.68±0.06 | 0.34±0.03 | 1,678.75±41.20 |
| (1000,0.8) | 19.67±1.71 | 23,897.72±309.97 | 0.74±0.05 | ⋆0.38±0.03 | 1,531.68±38.51 |
| (1000,0.5) | 14.89±0.68 | 21,232.90±161.44 | 0.78±0.04 | 0.27±0.02 | 1,170.96±26.27 |
| (1000,0.3) | 14.00±0.00 | 19,826.00±0.00 | 0.80±0.00 | 0.23±0.00 | 1,160.77±24.36 |
| ⋆(500,0.9) | 20.26±2.18 | 24,473.66±377.05 | 0.69±0.06 | 0.34±0.03 | 839.84±31.48 |
| (500,0.8) | 19.48±1.59 | 23,413.63±387.34 | 0.75±0.05 | 0.37±0.03 | 764.94±29.40 |
| (500,0.5) | 14.70±0.63 | 21,074.26±162.06 | 0.80±0.04 | 0.26±0.02 | 588.96±18.68 |
| (500,0.3) | 14.00±0.00 | 19,826.00±0.00 | 0.80±0.00 | 0.23±0.00 | ⋆584.50±17.64 |
| Dataset 2 30 % | | | | | |
| (500,0.3) | 53.99±3.78 | 111,783.21±296.19 | ⋆0.55±0.05 | 0.40±0.02 | 15,715.18±1,619.54 |
| (500,0.5) | 56.21±4.79 | 117,227.38±871.86 | 0.65±0.05 | 0.38±0.03 | 15,291.71±1,240.43 |
| (500,0.8) | 48.99±4.88 | 126,148.96±1,824.04 | 0.65±0.05 | 0.34±0.04 | 15,436.13±1,112.42 |
| ⋆(500,0.9) | 46.32±3.87 | 128,336.00±1,770.28 | 0.64±0.05 | 0.33±0.03 | ⋆14,674.33±235.43 |
| (1000,0.3) | 57.99±3.66 | 112,418.88±235.53 | 0.60±0.04 | ⋆0.41±0.02 | 28,915.27±371.20 |
| (1000,0.5) | ⋆62.74±4.97 | 119,395.37±826.79 | 0.69±0.05 | 0.39±0.02 | 29,141.69±638.77 |
| (1000,0.8) | 53.66±5.14 | 129,680.62±1,501.14 | 0.67±0.06 | 0.35±0.03 | 29,346.86±384.38 |
| (1000,0.9) | 52.47±5.23 | ⋆132,182.64±1,924.53 | 0.66±0.05 | 0.34±0.03 | 29,526.05±363.79 |
| Dataset 2 50 % | | | | | |
| (500,0.3) | 71.05±4.20 | 402,416.59±1,039.90 | 0.77±0.03 | 0.41±0.02 | ⋆59,385.89±1,531.46 |
| (500,0.5) | 69.32±5.01 | 407,057.50±1,531.07 | 0.74±0.04 | 0.40±0.03 | 61,738.85±4,761.45 |
| (500,0.8) | 73.32±5.54 | 415,226.09±1,785.76 | 0.74±0.04 | 0.31±0.03 | 63,504.13±3,792.53 |
| (500,0.9) | 67.99±5.32 | 420,263.92±2,183.74 | 0.74±0.04 | 0.32±0.03 | 61,463.12±1,935.45 |
| (1000,0.3) | 75.80±4.69 | 404,784.23±790.84 | 0.79±0.03 | ⋆0.44±0.02 | 121,349.99±2,508.53 |
| (1000,0.5) | 74.82±4.50 | 410,221.09±1,464.63 | ⋆0.74±0.03 | 0.41±0.02 | 122,381.05±3,872.40 |
| (1000,0.8) | ⋆81.39±5.85 | 419,804.19±1,617.48 | 0.74±0.04 | 0.31±0.02 | 120,585.52±4,053.31 |
| ⋆(1000,0.9) | 75.81±5.81 | ⋆425,642.47±1,894.97 | 0.74±0.05 | 0.32±0.03 | 118,335.71±2,000.27 |
| Dataset 2 70 % | | | | | |
| (500,0.3) | 120.51±5.59 | 745,387.59±781.13 | 0.74±0.03 | 0.34±0.02 | ⋆164,035.28±8,251.63 |
| (500,0.5) | 117.43±5.67 | 750,964.34±1,601.37 | 0.71±0.03 | 0.32±0.02 | 165,807.50±7,222.89 |
| (500,0.8) | 118.35±7.19 | 758,626.65±2,045.66 | 0.71±0.03 | 0.28±0.02 | 173,550.47±14,891.71 |
| (500,0.9) | 109.35±6.57 | 763,586.43±2,015.47 | ⋆0.70±0.03 | 0.28±0.02 | 168,849.21±9,234.50 |
| (1000,0.3) | 129.59±5.76 | 747,684.46±773.36 | 0.75±0.02 | ⋆0.34±0.02 | 329,101.66±11,926.18 |
| (1000,0.5) | 128.50±6.49 | 754,667.97±1,647.21 | 0.72±0.02 | 0.32±0.02 | 337,569.40±21,378.16 |
| (1000,0.8) | ⋆130.88±5.90 | 762,924.13±1,822.50 | 0.71±0.03 | 0.28±0.02 | 336,762.35±13,095.81 |
| ⋆(1000,0.9) | 120.14±7.27 | ⋆769,613.34±2,064.60 | ⋆0.70±0.03 | 0.29±0.02 | 324,604.49±13,244.30 |

**Table 9** Results of Non-dominated Sorting Genetic Algorithm (NSGA-II)

| Parameters (Pop, $p_{\text{mut}}$, $p_{\text{cross}}$,Gen) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 30 % | | | | | |
| (20,.05,.9,500) | 7.83±1.70 | 6,429.18±1,078.58 | 0.76±0.11 | 0.27±0.12 | 1,105.33±124.41 |
| (20,.05,.9,500) | 8.32±1.80 | 6,623.22±802.87 | 0.76±0.10 | 0.27±0.11 | ⋆1,100.97±122.37 |
| (40,.05,.9,250) | ⋆9.69±2.09 | ⋆6,842.92±849.03 | 0.76±0.09 | 0.29±0.11 | 1,891.55±196.10 |
| ⋆(40,.05,.8,250) | 9.44±2.57 | 6,663.56±935.59 | ⋆0.73±0.10 | ⋆0.33±0.13 | 1,856.43±216.09 |

**Table 9** (continued)

| Parameters (Pop, $p_{mut}$, $p_{cross}$,Gen) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 50 % | | | | | |
| (20,.05,.9,500) | 8.86±2.14 | 14,832.53±1,957.58 | 0.80±0.08 | 0.26±0.10 | ⋆1,130.43±106.19 |
| (20,.05,.8,500) | 9.38±2.15 | 15,205.51±1,623.13 | ⋆0.79±0.07 | 0.26±0.10 | 1,137.50±99.64 |
| ⋆(40,.05,.9,250) | 11.30±1.82 | ⋆15,676.77±1,214.25 | ⋆0.79±0.07 | 0.27±0.07 | 1,980.93±144.42 |
| (40,.05,.8,250) | ⋆11.46±1.98 | 15,664.81±1,339.10 | 0.80±0.06 | ⋆0.27±0.06 | 1,999.39±138.59 |
| Dataset 1 70 % | | | | | |
| (20,.05,.9,500) | 9.11±1.61 | 23,494.48±2,303.49 | 0.80±0.09 | 0.24±0.06 | ⋆1,144.04±95.96 |
| (20,.05,.8,500) | 9.22±2.09 | 23,358.35±3,061.03 | ⋆0.78±0.08 | 0.26±0.10 | 1,156.72±100.40 |
| (40,.05,.9,250) | 11.33±1.93 | 23,967.40±1,974.73 | 0.81±0.06 | ⋆0.26±0.05 | 2,035.61±123.05 |
| ⋆(40,.05,.8,250) | ⋆11.70±1.90 | ⋆24,408.67±1,746.18 | 0.80±0.07 | ⋆0.26±0.05 | 2,034.25±120.81 |
| Dataset 2 30 % | | | | | |
| ⋆(125,.01,.9,80) | 54.34±8.51 | 218,138.21±6,861.82 | ⋆0.80±0.07 | 0.22±0.05 | 28,127.77±1,275.83 |
| (125,.01,.8,80) | ⋆56.86±8.91 | ⋆219,662.51±6,320.36 | 0.81±0.07 | ⋆0.22±0.04 | 28,428.73±1,385.88 |
| (100,.01,.9,100) | 46.73±8.59 | 213,287.75±9,791.80 | 0.83±0.07 | 0.20±0.06 | ⋆22,987.83±1,329.29 |
| (100,.05,.8,100) | 50.79±8.12 | 216,657.11±7,377.47 | 0.82±0.06 | 0.20±0.05 | 24,134.12±1,546.03 |
| Dataset 2 50 % | | | | | |
| ⋆(125,.01,.9,80) | ⋆85.19±8.54 | ⋆542,408.15±22,993.26 | ⋆0.65±0.06 | 0.14±0.02 | 40,683.34±518.12 |
| (125,.01,.9,80) | 65.54±11.86 | 495,948.91±14,310.53 | 0.81±0.06 | ⋆0.19±0.03 | 35,045.76±840.87 |
| (125,.01,0.8,80) | 67.00±11.24 | 497,866.51±16,045.67 | 0.81±0.06 | ⋆0.19±0.03 | 35,149.95±743.31 |
| (100,.01,.9,100) | 53.26±10.04 | 483,773.06±19,331.55 | 0.85±0.07 | 0.17±0.04 | ⋆28,453.78±708.03 |
| (100,.05,.8,100) | 57.30±9.89 | 488,446.02±19,717.84 | 0.83±0.06 | 0.17±0.04 | 28,620.55±843.44 |
| Dataset 2 70 % | | | | | |
| (125,.01,.9,80) | 80.43±10.04 | 868,208.57±24,183.01 | ⋆0.77±0.05 | 0.18±0.03 | 38,716.48±1,681.94 |
| ⋆(125,.01,.8,80) | ⋆83.32±10.53 | ⋆873,383.61±24,556.01 | 0.77±0.06 | ⋆0.19±0.03 | 38,295.95±784.52 |
| (100,.01,.9,100) | 68.06±6.99 | 859,481.95±25,202.17 | 0.80±0.06 | 0.18±0.03 | ⋆31,283.26±581.24 |
| (100,.05,.8,100) | 71.72±8.69 | 870,345.56±23,093.39 | 0.77±0.06 | ⋆0.19±0.03 | 31,352.58±717.18 |

**Table 10** Results of Ant Colony System (ACS)

| Parameters (Ants, $\alpha$, $\beta$) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 30 % | | | | | |
| (10,0,1) | ⋆13.80±0.51 | ⋆7,817.44±38.32 | ⋆0.52±0.02 | 0.33±0.01 | 637.69±21.42 |
| (10,1,0) | 12.56±1.00 | 6,326.28±375.84 | 0.52±0.08 | ⋆0.34±0.02 | ⋆533.57±23.97 |
| (10,1,1) | 13.66±0.59 | 7,801.76±54.74 | 0.52±0.03 | 0.33±0.01 | 641.09±21.85 |
| ⋆(10,1,2) | 13.66±0.61 | 7,805.39±49.87 | 0.52±0.03 | 0.33±0.01 | 639.10±17.79 |
| (10,1,5) | 13.69±0.61 | 7,806.74±50.69 | 0.52±0.03 | 0.33±0.01 | 744.67±23.54 |
| Dataset 1 50 % | | | | | |
| (10,0,1) | 17.65±0.56 | 18,144.53±55.00 | 0.52±0.02 | ⋆0.37±0.01 | 789.03±34.51 |
| (10,1,0) | 15.73±1.22 | 14,951.09±603.70 | 0.57±0.05 | 0.37±0.02 | ⋆761.86±34.02 |
| (10,1,1) | 17.60±0.60 | 18,143.78±56.05 | 0.52±0.02 | ⋆0.37±0.01 | 788.59±27.80 |
| ⋆(10,1,2) | ⋆17.75±0.61 | ⋆18,153.33±51.26 | ⋆0.52±0.01 | ⋆0.37±0.01 | 787.62±33.29 |
| (10,1,5) | 17.66±0.59 | 18,140.09±64.45 | 0.52±0.02 | ⋆0.37±0.01 | 914.80±33.36 |

**Table 10** (continued)

| Parameters (Ants, $\alpha$, $\beta$) | # Sols | Hypervol | ΔSpread | Spacing | Exec. time (ms) |
|---|---|---|---|---|---|
| Dataset 1 70 % | | | | | |
| (10,0,1) | 20.57±0.64 | 29,188.04±61.87 | 0.49±0.02 | ⋆0.40±0.01 | 842.63±36.16 |
| (10,1,0) | 19.77±1.35 | 25,882.17±686.76 | 0.53±0.05 | 0.38±0.02 | 855.28±38.66 |
| ⋆(10,1,1) | ⋆20.57±0.62 | ⋆29,196.12±53.72 | ⋆0.48±0.02 | ⋆0.40±0.01 | ⋆836.76±34.35 |
| (10,1,2) | 20.53±0.74 | 29,195.16±55.44 | ⋆0.48±0.02 | ⋆0.40±0.01 | 837.62±31.57 |
| (10,1,5) | 20.50±0.73 | 29,176.01±79.45 | 0.49±0.02 | ⋆0.40±0.01 | 969.39±44.37 |
| Dataset 2 30 % | | | | | |
| (50,0,1) | ⋆47.41±5.87 | ⋆234,609.66±1,689.93 | ⋆0.69±0.06 | 0.41±0.04 | 618,755.93±26,695.67 |
| (50,1,0) | 38.12±5.15 | 111,445.99±3,602.45 | 0.79±0.06 | 0.35±0.05 | ⋆490,487.56±24,849.13 |
| (50,1,1) | 45.74±5.65 | 234,212.10±2,000.59 | ⋆0.69±0.06 | ⋆0.42±0.04 | 630,320.35±31,083.56 |
| ⋆(50,1,2) | 47.12±5.44 | 234,583.42±1,710.00 | 0.68±0.06 | 0.41±0.04 | 616,873.55±21,983.20 |
| (50,1,5) | 45.30±4.72 | 234,354.84±1,881.55 | 0.68±0.07 | 0.41±0.04 | 646,635.17±30,675.35 |
| Dataset 2 50 % | | | | | |
| (50,0,1) | 57.72±5.16 | 527,407.75±3,107.23 | 0.67±0.07 | ⋆0.44±0.04 | 771,106.11±37,794.04 |
| (50,1,0) | ⋆63.17±5.89 | 334,719.66±5,059.43 | 0.74±0.06 | 0.34±0.03 | 919,818.36±45,425.92 |
| (50,1,1) | 57.76±5.37 | 527,577.63±3,002.42 | ⋆0.65±0.07 | 0.43±0.03 | ⋆768,485.74±34,399.04 |
| ⋆(50,1,2) | 57.68±5.69 | ⋆527,685.22±2,738.50 | 0.66±0.06 | 0.43±0.04 | 770,221.21±37,331.73 |
| (50,1,5) | 57.38±5.01 | 527,682.14±2,663.33 | 0.66±0.06 | 0.43±0.03 | 812,531.72±47,958.66 |
| Dataset 2 70 % | | | | | |
| (50,0,1) | 70.92±5.78 | 902,040.57±4,206.39 | 0.64±0.05 | 0.45±0.03 | 908,792.96±56,879.39 |
| (50,1,0) | 84.45±6.95 | 674,078.31±6,146.74 | 0.72±0.04 | 0.34±0.03 | 1,227,042.34±75,250.72 |
| (50,1,1) | 70.05±4.89 | 902,380.79±3,655.72 | 0.63±0.06 | 0.44±0.03 | 914,333.94±63,482.50 |
| ⋆(50,1,2) | ⋆70.98±5.27 | ⋆902,769.29±3,141.76 | ⋆0.61±0.06 | 0.45±0.03 | ⋆881,950.91±54,188.14 |
| (50,1,5) | 69.18±4.52 | 901,764.91±3,425.91 | 0.63±0.05 | 0.45±0.03 | 914,228.95±50,084.02 |

Iterations = 100, $\rho = 0.01$, $q_0 = 0.95$

## References


Albrecht AJ, Gaffney JE (1983) Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans Softw Eng 9(6):639–648

Ali S, Briand LC, Hemmati H, Panesar-Walawege RK (2010) A systematic review of the application and empirical investigation of search-based test case generation. IEEE Trans Softw Eng 36(6):742–762

Arcuri A, Fraser G (2013) Parameter tuning or default values? An empirical investigation in search-based software engineering. Empir Softw Eng 18(3):594–623

Bagnall AJ, Rayward-Smith VJ, Whittley I (2001) The next release problem. Inf Softw Technol 43(14):883–890

Baker P, Harman M, Steinhöfel K, Skaliotis A (2006) Search based approaches to component selection and prioritization for the next release problem. In: Proceedings of 22nd IEEE international conference on software maintenance (ICSM 2006). IEEE Computer Society, Philadelphia, pp 176–185

Boehm BW (1981) Software engineering economics. Prentice Hall, Englewood Cliffs

Brooks FP (1995) The mythical man-month (anniversary edn). Addison-Wesley Longman Publishing Co., Inc., Boston

Carlshamre P (2002) Release planning in market-driven software product development: provoking an understanding. Requir Eng 7(3):139–151

Carlshamre P, Sandahl K, Lindvall M, Regnell B, och Dag JN (2001) An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of 5th IEEE international symposium on requirements engineering (RE 2001). IEEE Computer Society, Toronto, pp 84–93

Cheng BHC, Atlee JM (2007) Research directions in requirements engineering. In: Proceedings of international conference on software engineering, ISCE 2007. Workshop on the future of software engineering (FOSE 2007). Minneapolis, pp 285–303

Coello CAC, Lamont GB, Veldhuizen DAV (2007) Evolutionary algorithms for solving multi-objective problems evolutionary algorithms for solving multi-objective problems. Springer, New York

Davis AM (2003) The art of requirements triage. IEEE Comput 36(3):42–49

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York

Deb K, Agrawal S, Pratap A, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

del Sagrado J, del Águila IM (2009) Ant colony optimization for requirement selection in incremental software development. In: Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09). IEEE, Cumberland Lodge, Windsor

del Sagrado J, del Águila I, Orellana F (2010a) Ant colony optimization for the next release problem: a comparative study. In: Proceeding of Second International Symposium on Search Based Software Engineering (SSBSE 2010), Benevento, Italy, pp 67–76

del Sagrado J, del Águila IM, Orellana FJ, Túnez S (2010b) Requirements selection: knowledge based optimization techniques for solving the next release problem. In: Proceedings of the 6th Workshop on Knowledge Engineering and Software Engineering (KESE6). Karlsruhe, Germany, CEUR-WS.org, CEUR Workshop Proceedings, vol 636

del Sagrado J, del Águila IM, Orellana FJ (2011) Requirements interaction in the next release problem. In: Proceedings of 13th annual Genetic and Evolutionary Computation Conference (GECCO 2011). Dublin, Ireland, pp 241–242

del Sagrado J, del Águila IM, Orellana FJ (2012) Metaheuristic aided software features assembly. In: Proceeding of 20th European Conference on Artificial Intelligence (ECAI 2012) Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track. Montpellier, France, pp 1009–1010

de Souza JT, Maia CLB, Ferreira T, do Carmo RAF, Brasil M (2011) An ant colony optimization approach to the software release planning with dependent requirements. In: Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE '11), vol 6956. Springer, Szeged, Hungary, pp 142–157

Doerner KF, Gutjahr WJ, Hartl RF, Strauss C, Stummer C (2004) Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection. Ann Oper Res 131(1–4): 79–99

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66

Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge, MA

Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39

Durillo J, Zhang Y, Alba E, Nebro A (2009) A study of the multi-objective next release problem. In: Proceeding of 1st international symposium on search based software engineering (SSBSE 2009). Cumberland Lodge, Windsor, pp 49–58

Durillo JJ, Zhang Y, Alba E, Harman M, Nebro AJ (2011) A study of the bi-objective next release problem. Empir Softw Eng 16(1):29–60

Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett 8(2):67–71

Feo T, Resende M (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133

Finkelstein A, Harman M, Mansouri SA, Ren J, Zhang Y (2008) "Fairness analysis" in requirements assignments. In: Proceeding of 16th IEEE international requirements engineering conference (RE 2008). Barcelona, pp 115–124

Finkelstein A, Harman M, Mansouri SA, Ren J, Zhang Y (2009) A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirement Eng 14(4):231–245

Garey MR, Johnson DS (1990) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253

Häckel S, Fischer M, Zechel D, Teich T (2008) A multi-objective ant colony approach for pareto-optimization using dynamic programming. In: Proceedings of genetic and evolutionary computation conference (GECCO 2008). ACM, Atlanta, pp 33–40

Harman M (2007) The current state and future of search based software engineering. In: Future of software engineering, FOSE '07, pp 342–357

Harman M, Jones BF (2001) Search-based software engineering. Inf Softw Technol 43(14):833–839

Iredi S, Merkle D, Middendorf M (2001) Bi-criterion optimization with multi colony ant algorithms. In: Proceedings of evolutionary multi-criterion optimization, first international conference (EMO 2001). Zurich, pp 359–372

Jiang H, Zhang J, Xuan J, Re Z, Hu Y (2010) A hybrid ACO algorithm for the next release problem. In: Proceedings of the 2nd international conference on software engineering and data mining (SEDM '10). IEEE, Chengdu, pp 166–171

Johnson J (2003) CHAOS chronicles v3.0. Tech. rep.

Karlsson J, Olsson S, Ryan K (1997) Improving practical support for large-scale requirement prioritising. Requirement Eng 2(1):51–60

Kotonya G, Sommerville I (1998) Requirements engineering: processes and techniques. Wiley, New York

Momoh J, Ruhe G (2006) Release planning process improvement an industrial case study. Software Process Improvement and Practice 11(3):295–307

Pitsoulis L, Resende M (2003) Greedy randomized adaptive search procedures. Oxford University Press, pp 168–183

Resende M, Ribeiro C (2003) Greedy randomized adaptive search procedures. Kluwer Academic Publishers, Dordrecht, pp 219–249

Ruhe G, Saliu MO (2005) The art and science of software release planning. IEEE Softw 22(6): 47–53

Saliu MO, Ruhe G (2007) Bi-objective release planning for evolving software systems. In: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT international symposium on foundations of software engineering. Dubrovnik, Croatia, pp 105–114

Schott J (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization. PhD thesis, Massachusetts Institute of Technology, M.S., USA

Simmons E (2004) Requirements triage: what can we learn from a "medical" approach? IEEE Softw 21(4):86–88

Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evoltionary Computation 2:221–248

Vianna DS, Arroyo JEC (2004) A GRASP algorithm for the multi-objective knapsack problem. In: Proceedings of 24th International Conference of the Chilean Computer Science Society (SCCC 2004), pp 69–75

Wiegers KE (2003) Software requirements. Microsoft Press, Redmon, WA, USA

Wohlin C, Runeson P, Höst M, Ohlsson M, Regnell B, Wesslén A (2012) Experimentation in software engineering: an introduction. Springer, Berlin

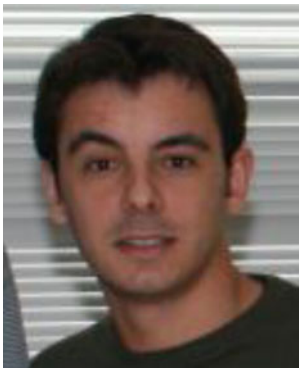
Zhang Y, Harman M (2010) Search based optimization of requirements interaction management. In: Proceedings of Second International Symposium on Search Based Software Engineering (SSBSE 2010). Benevento, Italy, pp 47–56

Zhang Y, Harman M, Mansouri SA (2007) The multi-objective next release problem. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007). London, England, UK, pp 1129–1137

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

**José del Sagrado** is currently a Professor of Computer Science in the Department of Informatics at the University of Almería. He obtained his PhD in Computer Science from the University of Granada in 2000. He is a member of the Spanish Association for Artificial Intelligence (AEPIA). His current research interests are mainly focused on probabilistic graphical models, specially combination of models, and probabilistic decision graphs, and also include the application of metaheuristic optimization in software engineering problems.

**Isabel María del Águila** is currently a Professor of Computer Science at Almería University. She received her Ph.D. degree in Computer Science from the University of Almería in 2010. She is specialized in Software Engineering, UML, information systems, and search based Software Engineering. Her research interests include Software engineering and Knowledge engineering methods, particularly the unification of these engineering approaches.



**Francisco J. Orellana** is currently a PhD student in the Department of Informatics at the University of Almería. He is a member of the Data, Knowledge and Software Engineering Group of the University of Almería and his research interests include search based software engineering, probabilistic graphical models and their applications.