Applied Mathematics and Computation 252 (2015) 1–13

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Differential evolution with Pareto tournament for the multi-objective next release problem

José M. Chaves-González [*], Miguel A. Pérez-Toledano

Computer Science Department, University of Extremadura, Cáceres, Spain

A R T I C L E   I N F O

Keywords:
Software requirements selection
Multi-objective evolutionary algorithm
Differential evolution metaheuristic
Search Based Software Engineering
Next release problem

A B S T R A C T

Software requirements selection is the engineering process in which the set of new requirements which will be included in the next release of a software product are chosen. This NP-hard problem is an important issue involving several contradictory objectives that have to be tackled by software companies when developing new releases of software packages. Software projects have to stick to a budget, but they also have to cover the highest number of customer requirements. Furthermore, in real instances of the problem, the requirements tackled suffer interactions and other restrictions which complicate the problem. In this paper, we use an adapted multi-objective version of the differential evolution (DE) evolutionary algorithm which has been successfully applied to several real instances of the problem. For doing this, the software requirements selection problem has been formulated as a multiobjective optimization problem with two objectives: the total software development cost and the overall customer's satisfaction, and with three interaction constraints. On the other hand, the original DE algorithm has been adapted to solve real instances of the problem generated from data provided by experts. Numerical experiments with case studies on software requirements selection have been carried out to demonstrate the effectiveness of the multiobjective proposal and the obtained results show that the developed algorithm performs better than other relevant algorithms previously published in the literature under a set of public datasets.

1. Introduction

The complexity and extension of modern software systems have been increased in the last decade. In addition, software products have to be usually developed in limited periods of time and with severe cost restrictions. Thus, software development companies have to efficiently satisfy large sets of requirements by minimizing the production efforts (in time and cost). In fact, in most cases it is not possible to develop all the new features suggested by the clients. Software requirements optimization is an important task in Software Engineering, and especially relevant within the incremental approaches of software development, such as in the agile methodologies [1]. In these methodologies, the software product is developed by generating releases which have to be produced in short iterative cycles. A new set of requirements, tailored to fit the needs of the clients and the development costs is proposed for each iteration. In this context, the challenge of Software Engineering is to define which requirements should be developed taking in consideration several complex factors (priority for different clients with different importance, development effort, cost restrictions, interactions between different requirements, etc). There is not a simple solution to this complex problem, which is also called in the related literature next release problem (NRP) [2].

* Corresponding author.
E-mail address: jm@unex.es (J.M. Chaves-González).

http://dx.doi.org/10.1016/j.amc.2014.11.093
0096-3003/© 2014 Elsevier Inc. All rights reserved.

The NRP is an NP-hard problem [3] which simultaneously manages two independent and conflicting objectives which have to be simultaneously optimized: the development effort (cost), and the clients' satisfaction. Thus, the problem cannot be managed by traditional exact optimization methods. In this case, multi-objective evolutionary algorithms (MOEAs) are the most appropriate strategies [4,5], because MOEAs tackle simultaneously several conflicting objectives without the artificial adjustments included in classical single-objective optimization methods. However, most of related works in the bibliography are simplified by using an aggregation function and they manage the problem as a single objective version of the problem. Furthermore, there are others works that do not tackle the interactions produced between the requirements in real NRP instances of the problem.

In this paper, a novel technique within the Search-Based Software Engineering (SBSE) [6] research field has been proposed to deal with a real multiobjective version of the NRP (MONRP). Specifically, in this paper we propose a multi-objective approach based on the differential evolution (DE) metaheuristics, which is a population based evolutionary algorithm [7]. We have modified the original scheme of DE to deal with the multiobjective NRP (MONRP). Thus, the MOEA proposed, which was named Differential Evolution with Pareto Tournaments (DEPT) incorporates some multi-objective features, such as the concept of Pareto tournament or the non-dominated sorting [8]. In order to test the accuracy of the proposed algorithm, we have compared it with the multi-objective standard NSGA-II (Fast Non-dominated Sorting Genetic Algorithm) [8], and other approaches proposed in other works published in the literature. As will be shown in this paper, DEPT algorithm provides high quality results, surpassing the results obtained by other authors, for several instances of the software requirements selection problem.

The rest of the paper has been organized as follows: Section 2 discusses related work. Section 3 describes the basic background on the problem and the multiobjective formulation which has been proposed. In Section 4, we explain our proposal and we provide some implementation details about the algorithm developed to deal with the software requirements selection problem. The experiments performed and the results obtained are presented and analyzed in Section 5. Finally, Section 6 summarizes the conclusions of the paper.

2. Related work

Requirements optimization is an NP-hard problem [3] which consists of selecting a set of requirements that will be developed for the next release of a software product, such that the requirements selected minimize the development cost and maximize the clients' satisfaction. The problem evaluates two conflicting objectives, and both objectives have to be equally considered by the solutions found. In the literature, Karlsson proposed in [9] two kinds of methods for selecting and prioritizing software requirements: Analytical Hierarchy Process (AHP) and Quality Function Deployment (QFD). In QFD, requirements are prioritized in an ordinal scale, and in AHP the requirements are classified by a pair cost-value. However, both kinds of methods do not support requirements interdependencies, which are real needs nowadays, and they have to perform a huge number of comparisons when the project scale is increased.

The requirements selection problem was firstly formulated as a single-objective problem in the Search-Based Software Engineering (SBSE) field by Bagnall et al. in [2]. SBSE is the research field in which search-based optimization algorithms are proposed to tackle problems in Software Engineering [6]. The original problem proposed in [2] has been solved with different metaheuristics along the last years. However, most of the approaches published are single-objective evolutionary algorithms which combine the objectives by using an aggregation function [10,11]. In all cases, those works did not consider the interactions produced among the requirements. Moreover, single objective formulation has the inconvenient of making a biased search of the solution space, because the objectives have to be artificially aggregated in some way, for example with a weighted sum of the objectives.

The NRP has been recently formulated as a multi-objective optimization problem (MOOP). Zhang et al. proposed the first multi-objective formulation for the original NRP [12] (MONRP). This formulation tackles each objective separately, without any combination function, and without considering any problem constraint, such as the interactions produced among the requirements or the cost limitations. In [13,14], the authors also use multi-objective optimization for the analysis of trade-offs among multiple clients with potentially conflicting requirements priorities, but the interactions produced among the requirements are not again considered. The same occurs in the works [15–17], in which different multi-objective evolutionary algorithms are proposed for solving NRP, but dependencies among requirements are not considered. In [15], the authors propose a quantum inspired evolutionary algorithm. On the other hand, in [16], the authors propose the well-known algorithms: PAES (Pareto Archived Evolution Strategy) [18], NSGA-II (Fast Non-dominated Sorting Genetic Algorithm) [8], and MOCell (MultiObjective Cellular Genetic Algorithm) [19] for solving the requirements selection problem. Finally, in [17], the authors solve the problem by using an Ant Colony Optimization (ACO) algorithm [20].

As far as we know, the only two studies in which the MONRP is tackled by considering the interactions between the software requirements are the works by Sagrado et al. [21,22], and by Souza et al. [23]. All those works propose the use of different ACO strategies to solve the problem, but only in [22], the datasets used in the study are public, so our results will be compared with that study in Section 5. To this respect, it is worth mentioning that many of the related studies do not make public the information about the datasets used (probably for commercial reasons), so it is not possible to perform numerical comparisons with them.

In this paper, we present a multi-objective search-based approach based on the differential evolution evolutionary algorithm [7]. We have adapted the algorithm to work with a MONRP formulation in which different types of requirements interactions and effort constraints are considered. Our proposal searches for high quality sets of solutions within a given development effort bound that balance the clients' priorities while keeping the requirements interactions. Very recent publications show the applicability of this approach in different domains. For example, in [30], DEPT is applied to the generation of DNA sequences for reliable DNA computing, and in [31] the DE algorithm is used for synonym recognition in the biomedical domain. Moreover, in [32] a complete survey about the DE algorithm is presented. In this study it can be seen that DE has been successfully applied to several engineering applications. Moreover, in those studies, the results obtained by the differential evolution algorithm have been compared with the results obtained by other approaches. In this paper, we apply DEPT to other research field, but, as shown in Section 5, the results obtained by our proposal are better than the results obtained by other multiobjective approaches published in the literature.

3. The multi-objective software requirements selection problem

This section explains the MONRP for the selection of software requirements. As was mentioned, the NRP problem was originally formulated in [12], but in this paper the formulation has been updated in order to manage real instances of the problem, in which different types of interactions are produced among the requirements tackled in the problem, and a new constraint to manage the cost restrictions has been included to the formulation. But before the explanation of the problem and its formulation, we introduce some multi-objective concepts which are necessary for a better understanding of this work.

3.1. Multi-objective concepts

In multi-objective optimization (MOO), the Pareto dominance and the Pareto front are two of the main ideas. In this paradigm, a problem does not have a unique optimal solution, but a Pareto front of solutions [4], which is a vector of decision variables which optimize the objective functions being considered and satisfy the problem constraints. Thus, the Pareto front contains a set of Pareto solutions which are dominated by no other solution. In this context, a solution $x = [x_1, x_2, \ldots, x_n]$, where n is the number of problem objectives, is said to dominate a solution $y = [y_1, y_2, \ldots, y_n]$, if and only if y is not better than x for any objective $i = 1, 2, \ldots, n$, and there exist at least one objective, x_i, in x which is better than the corresponding y_i objective in y. On the contrary, two solutions are said to be non-dominated whenever none of them dominates the other. Fig. 1 shows some examples of dominated and non-dominated solutions. If the objective functions represented in Fig. 1, *f1 and f2, are to be minimized*, the solution A dominates to the solution D because $f1\ (A) < f1\ (D)$ and $f2\ (A) < f2\ (D)$. On the other hand, A, B and C are non-dominated solutions because they are better than the other solutions in some of the objective functions in every case. Thus, A is better than B and C for the objective 1 ($f1$), C is better than A and B for the objective 2 ($f2$), and B is better than A for the objective 2 and better than C for the objective 1.

The solution of a given MOO problem is not a single high-quality solution, but a set of high-quality solutions, referred as Pareto optimal set, which satisfies two properties. First, every two solutions into the set are non-dominated solutions, and second, any other solution found, is dominated by at least one solution in the Pareto optimal set. The representation of this set in the objective space is known as Pareto front (see Fig. 1). Theoretically, a Pareto front could contain a large number of non-dominated solutions, but in practice, this set only contains a limited number of them. For this reason, the number of non-dominated solutions found for a specific problem is important, and also they are uniformly spread in the Pareto front (as shown in Fig. 1).

3.2. The next release problem

The next release problem consists of selecting a set of requirements which involves simultaneously a set of conflicting criteria that have to be simultaneously optimized. These criteria are equally important and they are usually related to the minimal cost of producing the new release of a software package, and to the maximal clients' satisfaction obtained with the release of that new version of the package software. The NRP is particularly relevant in agile software development approaches [1], which promote the iterative development of releases of a software product in short development cycles in order to improve productivity and to introduce checkpoints with the clients.

Moreover, the clients involved in the NRP have different categories depending on their level of importance or the category they occupy in the company. This importance can also be related to market factors which could make the clients modify their weight (e.g. the company may be interested on satisfying the newest clients' needs). In addition, each requirement will have different consideration for each client, which means that the final level of importance for a particular software requirement

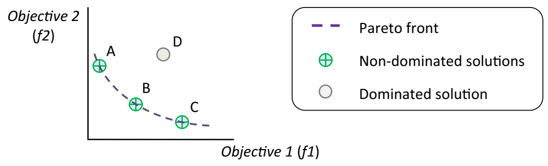

Fig. 1. Representation of dominated solution, non-dominated solutions and Pareto front.

will depend on the *priority* of that requirement for each client and on the *importance* level of each client. On the other hand, each requirement means an effort in terms of development *costs*, and the company resources are limited, so requirements which imply high costs are not preferred initially.

Finally, the requirements also present different problem interactions. This means that, for example, some requirements must be developed only after some others have been developed, or that some requirements cannot be developed if some others have been already included in the next release of the software package. The requirements interactions will be managed as problem constraints in the multi-objective model proposed in this paper. In [24], the authors proposed a classification for the interactions depending on different software development projects. This classification can be grouped into four main categories [22]: (a) *Implication* or *precedence*: $r_i \Rightarrow r_j$, which means that a requirement r_i cannot be chosen if the requirement r_j has not been previously chosen. (b) *Combination* or *coupling*: $r_i \oplus r_j$, which means that a requirement r_i has to be compulsorily included with the requirement r_j. (c) *Exclusion*: $r_i \otimes r_j$, which means that a requirement r_i cannot be included together with the requirement r_j. (d) *Modification*: The development of the requirement r_i implies that some other requirements modify their implementation cost or the satisfaction they provide to the clients.

Therefore, the requirements selection problem can be formulated as a MOOP in which the two objectives are: the *clients' satisfaction*, which has to be *maximized*, and the *software development cost*, which has to be *minimized*. Furthermore, these objectives are subject to the problem *interactions* produced among the requirements, which have to be managed as constraints for the MOOP. Considering the requirements selection problem as a MOOP has the advantage of providing to the decision maker with a set of non-dominated solutions instead of a single best solution. Thus, the expert can eventually choose the best solution depending on different situations and considering complex technical, sociological, economic and political factors which are beyond any mathematical formulation. The search based proposal described in this paper can complement this rich human domain knowledge, by providing the best choices available according to a cost-benefit analysis. The MO proposal reduces the enormous number of potential solutions guided by cost-benefit data supplied by the human expert, and it is the human who considers the range of solutions provided in the Pareto front generated by the multi-objective algorithm.

3.3. Multi-objective formulation

This section provides a formal description for the multi-objective NRP (MONRP), which is an extension of the original NRP defined in [2]. Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of n requirements which have to be developed for the next release of a software package. These requirements repreent improvements to the current software system which are suggested by a set of m clients, $C = \{c_1, c_2, \ldots, c_m\}$. Each client has a degree of importance for the company that is reflected by a weight factor. The set of clients' weights is denoted by $W = \{w_1, w_2, \ldots, w_m\}$. Moreover, each requirement r_j in R has an associated cost, e_j, which represents the effort needed for the development of that requirement. Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of costs which are associated to the n requirements.

It is assumed that each client in C gives a different importance to each requirement in R. The importance that a requirement r_j has for a particular client c_i is given by a value $v_{ij} > 0$. A zero value for v_{ij} means that the client c_i has not suggested the requirement r_j. A priority matrix of $m \times n$ holds all the importance values v_{ij}. The global satisfaction s_j of a given requirement r_j is calculated as the weighted sum of its importance values for all the clients considered, and can be expressed as indicated in Eq. (1). The set of the global satisfactions calculated in that way is denoted by $S = \{s_1, s_2, \ldots, s_n\}$.

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

The MONRP consists of finding a decision vector, X, which determines the requirements that will be developed for the next software release. X is a subset of R, and contains the requirements that maximize the clients' satisfaction, and minimize the development cost. These are the two problem objectives, but in addition, all requirements in X must satisfy the constraints of the problem. These constraints are mainly related to the restrictions generated for the requirement interactions. According to the datasets used in this work, three types of interactions are managed: implication interactions: $r_i \Rightarrow r_j$ (if r_i belongs to X, r_j must be also in X), combination interactions: $r_i \oplus r_j$ (if r_i belongs to X, r_j must be also in X, and vice versa), and exclusion interaction: $r_i \otimes r_j$ (if r_i belongs to X, r_j cannot be in X, and vice versa). Furthermore, every solution proposed has to satisfy a cost threshold constraint L_C, as is expressed in Eq. (2).

$$\sum_{j \in X} e_j \leqslant L_C \tag{2}$$

Thus, MONRP can be formulated as expressed in Eq. (3):

$$\begin{aligned} &\mathit{Maximize}\ S(X) = \sum_{j \in X} s_j \\ &\mathit{Minimize}\ E(X) = \sum_{j \in X} e_j \\ &\text{Subject to} \begin{cases} \mathit{cost\ threshold\ constraint} \\ \mathit{interactions\ constraints} \end{cases} \end{aligned} \tag{3}$$

The solution to the problem will be a set of nd non-dominated solutions $\{X_1, X_2, \ldots, X_{nd}\}$ which satisfy the conditions indicated in Eq. (3).

4. Multi-objective differential evolution for the MONRP

After the introduction of the details about the MONRP, in this section we detail the main features of the multi-objective evolutionary algorithm developed, but previously to this explanation, we present the codification of the solutions managed by the proposal.

4.1. Solution codification

In evolutionary computation, the individual used in the algorithm is a solution for the problem being tackled. This solution should be improved generation after generation of the algorithm, so it is important a good design that provides a fast processing when the genetic operators work with them. For the problem tackled, the solution is encoded with the purpose of giving support to all the information necessary to represent and evaluate valid sets of requirements for the requirements selection problem. Fig. 2 shows the representation of the individual used by the multiobjective metaheuristic proposed in this paper. The solution has been encoded as a requirement vector of Booleans (X) with n positions. Each position of X indicates whether the requirement j is selected for the next release of the software package. In this case, that position is equal to 1. Otherwise, the value for that position is 0. The total number of requirements in X, υ, is also saved in the individual data structure ($X_{ReqsNumber}$). Finally, the global quality for the solution is evaluated through the values of the two objectives managed (cost and overall satisfaction). These values are calculated as explained in Section 3.3, and they are saved also in the individual data structure (as can be observed in Fig. 2).

4.2. Multi-objective DE algorithm with Pareto tournament

In this paper, a multi-objective population-based evolutionary algorithm is proposed to solve the multi-objective requirements selection problem. Differential evolution (DE) was originally proposed in [7]. As was explained in the previous section, the problem has been formulated as a MOOP, so we have developed an adapted multiobjective version of the algorithm, the Differential Evolution with Pareto Tournaments (DEPT). This algorithm adds to the original design of DE some multiobjective (MO) features. Thus, from NSGA-II [8], we took the ideas of *dominance* and *non-dominated sorting*. These concepts are used to qualify and sort the solutions within the population in a multiobjective environment. On the other hand, from PAES [18], we took the idea of *non-dominated solution archive* (*NDS_archive*). This archive maintains updated the best solutions found by the algorithm, according to an acceptance function. A pseudocode of our proposal is shown in Algorithm 1.

DEPT includes three important parameters: the crossover factor, Cr, the mutation factor, F, and the *selection schemes* (shown in Table 1). Each scheme uses different expressions to generate trial requirements vectors. However, it is very likely that the resultant trial individuals do not satisfy the problem constraints. Therefore, each new individual generated, x_{trial}, has to be corrected in case that any constraint is not satisfied. This feature makes our proposal faster than other approaches published, as will be discussed in Section 5.2.

Fig. 3 shows an example for the process of individual verification and correction. As can be observed in the example, the x_{trial} individual must satisfy two constraints. There is a combination constraint between r_2 and r_3 ($r_2 \oplus r_3$), which means that if the requirement r_2 is chosen, the requirement r_3 has to be also chosen, and an exclusion constraint between r_2 and r_n ($r_2 \otimes r_n$), which means that if r_2 is chosen, r_n cannot be chosen. In the example, the interactions are checked and the requirements with problems are fixed. The result is given in *auxInd* (the changes are marked with a discontinuous line). But after that, it is necessary that every solution satisfies the cost threshold constraint L_C, which in case of the example is established to 30% of the total development effort. Thus, in Fig. 3 can be observed that requirements r_2 and r_3 are unselected to fulfill this constraint, and the final solution is represented as indicated in the x'_{trial} final solution. It is worth mentioning here that in case that the individual is changed to fulfill any requirement constraint, it is checked again to verify that no new restrictions have been violated due to the changes performed. The verification process stops when the final individual, x'_{trial}, is a valid solution which satisfies all the problem constraints. The procedure for the constraint verification allows that every artificially generated solution can be efficiently checked and, if necessary, repaired, so every solution managed by the algorithm can be considered a valid solution.

Our proposal uses the P_X parameter to determine which requirements in the vector have to be crossed. It uses the P_{Mut} parameter to obtain the mutation probability applied to the crossing process. The requirements of the individuals which

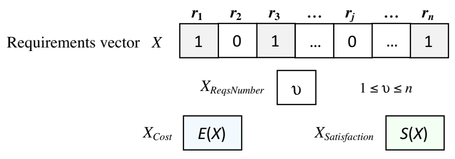

Fig. 2. Data structure for the individual.

Table 1
Selection schemes used in the differential evolution algorithm.

DE scheme	x_{trial} vector generation
Best/1/exponential	$x_{trial} = x_{best} + F(x_{r1} - x_{r2})$
Rand/1/exponential	$x_{trial} = x_{r3} + F(x_{r1} - x_{r2})$
RandToBest/1/exponential	$x_{trial} = x_{r3} + F(x_{best} - x_{r3}) + F(x_{r1} - x_{r2})$
Best/2/exponential	$x_{trial} = x_{best} + F(x_{r1} + x_{r2} - x_{r3} - x_{r4})$
Rand/2/exponential	$x_{trial} = x_{r5} + F(x_{r1} + x_{r2} - x_{r3} - x_{r4})$
Best/1/binomial	$\mathrm{x}_{\mathrm{trial}} = \mathrm{x}_{\mathrm{best}} + F(\mathrm{x}_{\mathrm{r1}} - \mathrm{x}_{\mathrm{r2}})$
Rand/1/binomial	$x_{trial} = x_{r3} + F(x_{r1} - x_{r2})$
RandToBest/1/binomial	$x_{trial} = x_{r3} + F(x_{best} - x_{r3}) + F(x_{r1} - x_{r2})$
Best/2/binomial	$x_{trial} = x_{best} + F(x_{r1} + x_{r2} - x_{r3} - x_{r4})$
Rand/2/binomial	$x_{trial} = x_{r5} + F(x_{r1} + x_{r2} - x_{r3} - x_{r4})$

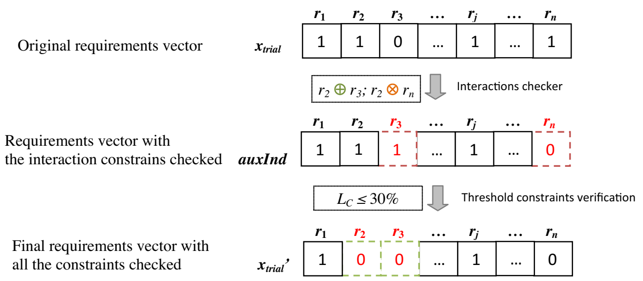
Original requirements vector
x_{trial}
r_1 r_2 r_3 … r_j … r_n
1 1 0 ... 1 ... 1
$r_2 \oplus r_3;\ r_2 \otimes r_n$
Interactions checker
Requirements vector with the interaction constrains checked
auxInd
1 1 1 ... 1 ... 0
$L_C \leq 30\%$
Threshold constraints verification
Final requirements vector with all the constraints checked
x_{trial}'
1 0 0 ... 1 ... 0

Fig. 3. Procedure for the constraints verification and correction of the individual.

are used in the schemes expressed in Table 1 are obtained either randomly ($x_{r1} - x_{r5}$) or from one of the best individuals in the population (x_{best}). We distinguish two types of selection schemes: binomial and exponential. These schemes correspond to the binomial and exponential crossovers [25]. In the exponential schemes, only one requirement is mutated, so a requirement in the whole individual is randomly chosen and mutated, and the rest of requirements remain unchanged. On the other hand, the binomial schemes mutate all requirements that are not selected by the P_X parameter.

Algorithm 1. Pseudocode for our multi-objective differential evolution algorithm

```
1: $P \Leftarrow$ generateRandomInitialPopulation ($P\_Size$)
2: $P \Leftarrow$ evaluatePopulation ($P\_Size$)
3: $NDS\_archive \Leftarrow \emptyset$
4: while not stop condition satisfied do
5:    for $i = 1$ to $P\_Size$ do
6:       $x_{target} \Leftarrow P_i$
7:       /* Select three random individuals from population, P
           $x_{r1} \neq x_{r2} \neq x_{r3}$ */
8:        $x_{r1}, x_{r2}, x_{r3} \Leftarrow$ selectRandomIndividuals ($P$)
9:        /* Generate a new individual, $x_{trial}$, by using the selection
           scheme Rand/1/Bin */
10:         for each requirement $r$ in $x_{trial}$ do
11:            if $P_X$ indicates that the requirement $r$ has to be changed then
12:               $x_{trial}[r] \Leftarrow x_{r3}[r] + P_{Mut}(x_{r1}[r] - x_{r2}[r])$
13:            else
14:               $x_{trial}[r] \Leftarrow x_{target}[r]$
15:            end if
16:         end for
17:         $x_{trial} \Leftarrow$ evaluateIndividual ($x_{trial}$)
18:         $P_i \Leftarrow$ paretoTournamet ($x_{target}, x_{trial}$)
19:      end for
20:      /* Update Pareto solutions archive */
21:      $NDS\_archive \Leftarrow$ updateNDS_archive ($P$)
22:   end while
```

The algorithm starts with the random generation of the initial population P (line 1). After that, the value of each objective is calculated for each individual according to the mathematical description given in Section 3.3. Then, the individuals in P are classified into different Pareto fronts depending on their quality (line 2). Related to this, a particular solution is of higher quality than another if it is dominated by a fewer number of solutions, and in case of the MONRP, a particular solution dominates another if it is better, at least, in one of the two objectives tackled and it is not worse in the other. After the solutions are ranked by dominance, the crowding distance [8] of each solution is also calculated. By considering both measures, it is said that a particular individual is better than another if it has fewer dominance ranking, or, in case that it has the same dominance ranking, if it presents higher crowding distance, because in this case, solutions which are located in lesser crowded regions are preferred [8].

After the solution ranking, but previous to the main loop of the algorithm, the set of non-dominated solutions (*NDS_archive*) is also initialized (line 3). Afterwards, the core of the algorithm starts, and it begins with the selection of one individual of the population as a target, x_{target}, to generate the corresponding trial individual (line 6). The first target is the first population member, the second is the second population member, and so on (line 5). The process for generating the trial individual will be different depending on the selection scheme configured (Table 1). In Algorithm 1, the rand/1/binomial selection scheme is used, because this is the scheme which provides better results for our problem. This scheme requires the selection of three random individuals (line 8) for the trial individual generation process. This trial individual is generated by following the expressions in lines 10–16.

After the trial individual is generated, it is evaluated by following the equations described in Section 3.3 (line 17), and after that, the algorithm selects the best solution between the target and the trial individuals. For doing this, we have developed the *paretoTournament* function (line 18), which is detailed in Algorithm 2. The winner of the Pareto tournament will be added to the population for the next generation of the algorithm (and the other individual will be discarded). This process is repeated until all the population members are processed as target solutions. Finally, the algorithm updates the *NDS* (non dominated solutions) archive with the best ranked solutions in the population (line 21). When the algorithm finishes, this record will be processed to get the best ranked solutions in the global process.

Algorithm 2 paretoTournament algorithm

```
1: $tournamentWinner \Leftarrow x_{target}$
2: if $x_{target} \neq x_{trial}$ then
3:    $targetRank \Leftarrow$ calculateDominance ($x_{target}$)
4:    trialRank $\Leftarrow$ calculateDominance ($x_{trial}$)
5:    if $trialRank < targetRank$ then
6:       $tournamentWinner \Leftarrow x_{trial}$
7:    else if $trialRank == targetRank$ then
8:       $CWPopulation \Leftarrow$ calculateCrowdingPopulation ($x_{target}, x_{trial}$)
9:       $targetCW \Leftarrow$ calculateCrowdingDistance (CWPopulation, $x_{target}$)
10:      $trialCW \Leftarrow$ calculateCrowdingDistance (CWPopulation, $x_{trial}$)
11:      if $trialCW >$ targetCW then
12:         $tournamentWinner \Leftarrow x_{trial}$
13:      end if
14:   end if
15: end if
16: return tournamentWinner
```

The *paretoTournament* function used in the DEPT algorithm (Algorithm 2) chooses the best individual between two given solutions (which are the input parameters). If both solutions are different, their rank is calculated (lines 3 and 4 in Algorithm 2) by considering the concept of dominance. As was mentioned previously, an individual is better than another if it is dominated by fewer individuals, (it is better, at least, in one of the objectives and it is not worse in any of the others). If the rank of the trial individual is lower than the rank calculated by the target individual, the winner of the Pareto tournament is the trial individual. In case of a tie (line 7), the crowding distance of both individuals is calculated. Then, a population with the solutions that belong to the trial and target Pareto fronts is generated (line 8), and after that, the crowding distance for the trial and target individuals is calculated as indicated in lines 9 and 10 of Algorithm 2 [8]. If the crowding distance of the trial individual is higher than that obtained by the target individual, the trial individual is the winner of the Pareto tournament. Otherwise, the winner is the target individual, and no replacements are performed.

5. Experiments and results

In this section, we first describe the methodology followed in the experiments performed and the data sets used to evaluate the performance of the algorithm proposed. Then, we present the obtained results by using different quality indicators and we compare them with the results obtained by other approaches published in the literature.

5.1. Experimental methodology

All the experiments performed in this study have been run under the same environment. On the hardware side, we have used an Intel Xeon 2.33 GHz processor with 4 GB RAM. On the software side, we have used the *gcc* 4.4.5 compiler on a Linux kernel 3.13 64 bits OS. Since we are dealing with a stochastic algorithm, we have carried out 100 independent runs for each experiment. Results provided in the following subsections are average results of these independent executions. It is important to point here that the use of the arithmetic mean is a valid statistical measurement because the results follow a normal distribution, as is confirmed by an obtained *p-value* greater than 0.05 for the Shapiro–Wilk test [26]. Moreover, all the obtained results present a very low dispersion (as will be shown in the result tables presented in the following subsection), so the generated data can be considered statistically reliable.

The effectiveness of the algorithm proposed has been tested by using two different real data sets. In addition, each dataset has been restricted by using four different development effort boundaries (30%, 50%, 70% and 100% of the total development effort), so it can be said that our proposal has been tested by using four instances of each dataset (a total number of eight instances of the next release problem).

The first dataset was taken from Greer and Ruhe [11], and it includes 20 requirements and 5 clients. Table 2 shows the development effort associated to each requirement, the level of priority assigned to each requirement for each client, and the interactions constraints. The priority level for each requirement takes values from 1 to 5 depending on the level of importance. These values can be understood as follows: (1) not important requirement, (2) minor requirement, (3) important requirement, (4) very important requirement, and (5) extremely important requirement. Moreover, each requirement has an associated development cost (effort) which is estimated in terms of a score between 1 and 10. Finally, a set of implication and combination interactions between requirements are also considered.

On the other hand, each client has a relative importance in the decision making of the company. The global priority that a particular client gives to a specific requirement will depend on the priority level assigned (expressed in Tables 2 and 3) and the weight assigned to that client (see Eq. (1)). The clients' weights for the two datasets used in this paper are shown in Table 4. The values in this table are between 1 (the less important client) and 5 (the most important client).

The dataset presented in Table 2 is not very large, but to this respect, we want to clarify that due to the privacy policies followed by software development companies, as far as we know, the only two available real datasets are the ones which are included in this study. In addition, these datasets have been previously used in other relevant recent studies [22], so we use these datasets to compare the results obtained in this paper with other studies published in the literature (Section 5.2).

The second dataset was proposed by Sagrado et al. [22], and it includes 100 requirements, 5 clients and 44 requirements interactions. Table 3 shows the development effort associated to each requirement, the level of priority assigned to each requirement for each client, and the interactions constraints. The relative importance for each client is shown in the second row of Table 4. As observed, this second dataset is more complex than the previous one. In fact, both, the number of requirements (100) and the development costs (which are, in this case, in a range between 1 and 20) are taken from real agile software projects developments. Thus, the maximum development effort for a requirement is established in 20 effort units, which can be translated into 4 weeks. This temporal limit is usually defined as a time box in agile Software Engineering methods (e.g. Scrum proposes iterations between 2 and 4 weeks, [1]). The values for the priority level are in this case between 1 and 3, because when clients have to make an assignment of the benefit that will imply the inclusion of a new requirement, they prefer to use a coarse grained scale. Thus, clients simply place the requirements into one of these three categories [27,28]: (1) inessential, (2) desirable, and (3) mandatory.

As we work in a multiobjective environment, the parameter configuration of our proposal has been established according to the quality of the Pareto front produced in each test. In this work, we have used three quality indicators with the purpose of conducting a comparative study with other relevant works published. First, the *number of non-dominated solutions* found (*NDS*). Pareto fronts with a higher number of non-dominated solutions are preferred. Second, the *hypervolume* (*HV*) indicator [29]. This measure calculates the volume (in the objective space) covered by members of a non-dominated set of solutions Q (see Eq. (4)). *HV* measures convergence and diversity of the obtained Pareto fronts, so a Pareto front with a higher *HV* than another one could be due to two factors: some solutions in the better front dominate solutions in the other, or, solutions in the better front are more widely distributed than in the other. Since both properties are considered to be good, algorithms with larger values of *HV* are considered to be desirable. In order to calculate this metrics, two reference points are required. Since the problem we manage includes two objectives, these points are: $r_{min}\,(obj1_{min}, obj2_{min})$ and $r_{max}\,(obj1_{max}, obj2_{max})$,

Table 2
Requirements development cost (effort), requirements priority level for each client and interactions for dataset 1.

Effort		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}
		1	4	2	3	4	7	10	2	1	3	2	5	8	2	1	4	10	4	8	4
Priority level	cl_1	4	2	1	2	5	5	2	4	4	4	2	3	4	2	4	4	4	1	3	2
	cl_2	4	4	2	2	4	5	1	4	4	5	2	3	2	4	4	2	3	2	3	1
	cl_3	5	3	3	3	4	5	2	4	4	4	2	4	1	5	4	1	2	3	3	2
	cl_4	4	5	2	3	3	4	2	4	2	3	5	2	3	2	4	3	5	4	3	2
	cl_5	5	4	2	4	5	4	2	4	5	2	4	5	3	4	4	1	1	2	4	1
Interactions	$r_4 \Rightarrow r_8$ $r_4 \Rightarrow r_{17}$ $r_8 \Rightarrow r_{17}$ $r_9 \Rightarrow r_3$ $r_9 \Rightarrow r_6$ $r_9 \Rightarrow r_{12}$ $r_9 \Rightarrow r_{19}$ $r_{11} \Rightarrow r_{19}$ $r_3 \oplus r_{12}$ $r_{11} \oplus r_{13}$																				

Table 3
Requirements development cost (effort), requirements priority level for each client and interactions for dataset 2.

Effort		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}
		16	19	16	7	19	15	8	10	6	18	15	12	16	20	9	4	16	2	9	3
Priority level	cl_1	1	2	1	1	2	3	3	1	1	3	1	1	3	2	3	2	2	3	1	3
	cl_2	3	2	1	2	1	2	1	2	2	1	2	3	3	2	1	3	2	3	3	1
	cl_3	1	1	1	2	1	1	1	3	2	2	3	3	3	1	3	1	2	2	3	3
	cl_4	3	2	2	1	3	1	3	2	3	2	3	2	1	3	2	3	2	1	3	3
	cl_5	1	2	3	1	3	1	2	3	1	1	2	2	3	1	2	1	1	1	1	3

		r_{21}	r_{22}	r_{23}	r_{24}	r_{25}	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}
Effort		2	10	4	2	7	15	8	20	9	11	5	1	17	6	2	16	8	12	18	5
Priority level	cl_1	2	1	1	1	3	3	3	3	1	2	2	3	2	1	2	2	1	3	3	2
	cl_2	3	3	3	2	3	1	2	2	3	3	1	3	2	2	1	2	3	2	3	3
	cl_3	2	1	2	3	2	3	3	1	3	3	3	2	1	2	2	1	1	3	1	2
	cl_4	1	1	1	2	3	3	2	1	1	1	1	2	2	2	3	2	2	3	1	1
	cl_5	1	1	3	3	3	2	2	3	2	3	1	1	3	3	2	2	1	1	2	1

		r_{41}	r_{42}	r_{43}	r_{44}	r_{45}	r_{46}	r_{47}	r_{48}	r_{49}	r_{50}	r_{51}	r_{52}	r_{53}	r_{54}	r_{55}	r_{56}	r_{57}	r_{58}	r_{59}	r_{60}
Effort		6	14	15	20	14	9	16	6	6	6	6	2	17	8	1	3	14	16	18	7
Priority level	cl_1	2	2	3	1	1	1	2	2	3	3	3	3	1	3	2	1	3	1	3	1
	cl_2	3	3	1	1	3	2	2	2	1	3	3	3	1	2	2	3	3	2	1	1
	cl_3	1	3	1	3	3	3	3	1	3	2	3	1	2	3	2	3	2	1	2	3
	cl_4	3	1	1	3	1	2	1	1	3	2	2	1	3	2	1	3	3	1	2	3
	cl_5	3	1	1	2	1	2	3	3	2	2	1	3	3	2	3	1	2	1	3	2

		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}	r_{76}	r_{77}	r_{78}	r_{79}	r_{80}
Effort		10	7	16	19	17	15	11	8	20	1	5	8	3	15	4	20	10	20	3	20
Priority level	cl_1	2	2	3	3	1	3	1	3	2	3	1	3	2	3	1	1	2	3	3	1
	cl_2	1	3	2	3	1	2	1	2	3	1	1	3	1	3	2	1	3	3	1	2
	cl_3	1	1	2	3	3	1	3	3	3	1	3	1	3	1	1	2	3	3	1	2
	cl_4	2	2	3	3	3	1	2	1	2	1	2	3	3	2	2	2	1	3	3	1
	cl_5	2	2	1	2	1	3	2	1	2	1	2	2	3	2	1	3	2	3	1	3

		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}
Effort		10	16	19	3	12	16	15	1	6	7	15	18	4	7	2	7	8	7	7	3
Priority level	cl_1	2	1	3	1	2	2	2	1	3	2	2	3	1	1	1	2	1	3	1	1
	cl_2	1	2	1	2	2	1	3	2	2	2	3	2	2	3	2	2	1	3	1	1
	cl_3	1	2	3	2	3	1	2	2	3	3	3	3	2	1	1	2	3	3	2	3
	cl_4	3	1	2	2	2	1	1	1	3	1	1	3	3	1	2	1	2	3	1	3
	cl_5	3	2	1	2	2	2	2	1	3	3	3	1	1	3	1	3	3	3	3	3

Interactions
$r_2 \Rightarrow r_{24}$ $r_3 \Rightarrow r_{26}$ $r_3 \Rightarrow r_{27}$ $r_3 \Rightarrow r_{28}$ $r_3 \Rightarrow r_{29}$ $r_4 \Rightarrow r_5$ $r_6 \Rightarrow r_7$ $r_7 \Rightarrow r_{30}$ $r_{10} \Rightarrow r_{32}$ $r_{10} \Rightarrow r_{33}$ $r_{14} \Rightarrow r_{32}$
$r_{14} \Rightarrow r_{34}$ $r_{14} \Rightarrow r_{37}$ $r_{14} \Rightarrow r_{38}$ $r_{16} \Rightarrow\ \Rightarrow r_{39}$ $r_{16} \Rightarrow r_{40}$ $r_{17} \Rightarrow r_{43}$ $r_{29} \Rightarrow r_{49}$ $r_{29} \Rightarrow r_{50}$ $r_{29} \Rightarrow r_{51}$ $r_{30} \Rightarrow r_{52}$
$r_{30} \Rightarrow r_{53}$ $r_{31} \Rightarrow r_{55}$ $r_{32} \Rightarrow r_{56}$ $r_{32} \Rightarrow r_{57}$ $r_{33} \Rightarrow r_{58}$ $r_{36} \Rightarrow r_{61}$ $r_{39} \Rightarrow r_{63}$ $r_{40} \Rightarrow r_{64}$ $r_{43} \Rightarrow r_{65}$ $r_{46} \Rightarrow r_{68}$
$r_{47} \Rightarrow r_{70}$ $r_{55} \Rightarrow r_{79}$ $r_{56} \Rightarrow r_{80}$ $r_{57} \Rightarrow r_{80}$ $r_{62} \Rightarrow r_{83}$ $r_{62} \Rightarrow r_{84}$ $r_{64} \Rightarrow r_{87}$
$r_{21} \oplus r_{22}$ $r_{32} \oplus r_{33}$ $r_{46} \oplus r_{47}$ $r_{65} \oplus r_{66}$

Table 4
Clients' relative importance.

Clients' weights	cl_1	cl_2	cl_3	cl_4	cl_5
Dataset 1	1	4	2	3	4
Dataset 2	1	5	3	3	1

where those points contain the minimum and maximum values for the two objectives (development cost and overall clients' satisfaction). Note that *HV* is not free from arbitrary scaling of objectives, so the value of this metrics may be distorted if the range of each objective function is different. Thus, before calculating hypervolume, all the objective function values have to be normalized. The normalization points used for each dataset are shown in Table 5.

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

The third quality indicator is referred to the extent of spread achieved by the set of obtained solutions (Δ-*Spread*). This indicator measures the diversity of the solutions by using the Euclidean distances between consecutive solutions in the Pareto front. Pareto fronts with a smaller spread value are preferred. Δ is defined as expressed in Eq. (5).

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1}|d_i - \bar{d}|}{d_f + d_l + (n-1)d} \tag{5}$$

where d_i is the Euclidean distance between two consecutive solutions, $\bar{d}$ is the mean of every distance between each pair of solutions, N is the total number of solutions in the Pareto front, and d_f and d_l are, respectively, the Euclidean distance from the first and the last solution in the Pareto front to the extreme solutions of the optimal Pareto front in the objective space.

Finally, about the algorithm configuration, in order to perform fair comparisons with other works, we have established the same stop condition for our proposal: 10,000 fitness function evaluations [22]. The other algorithm parameters were tuned separately to obtain the best results for the problem tackled. The value of each parameter has been obtained after testing, at least, five equidistant values within the range of that parameter, and all the selection schemes shown in Table 1 have been tested. Table 6 summarizes the parameter configuration for our proposal.

5.2. Result discussion and comparison with other authors

In this subsection we present the results obtained with our proposal for different problem instances and we compare them with the results obtained with other proposals published in the literature. We start subsection by analyzing the values of *HV* and Δ quality indicators. After that, we perform a comparative study on the number of non-dominated solutions (*NDS*) obtained by different approaches.

5.2.1. Hypervolume (HV) results

Tables 7 and 8 summarize the comparative results in terms of the hypervolume indicator. We compare the results obtained with our proposal, DEPT, with other algorithms published very recently in the literature (ACO, NSGA-II and GRASP) [22]. Note that the *HV* results shown in [22] are not given in percentage. These values can be easily obtained by multiplying the values in Tables 7 by 75,905 (85 × 893), and then, by dividing the obtained result by 100, in case of dataset 1. For dataset 2, the results given in Table 8 have to be multiplied by 2,754,272 (1037 × 2656), and then, divided the obtained result by 100.

Results show average hypervolume (and standard deviation) of 100 independent runs for the 2 datasets under study, with the four development effort boundaries managed (30%, 50%, 70% and 100% – no effort limit). Therefore, a total number of 8 instances of the problem have been studied. For *HV* indicator, the highest the value, the better the quality of the obtained results. Therefore, we can observe that the DEPT algorithm is the proposal which obtains the best results regarding to this indicator for every problem instance. This means that DEPT is able to explore better the space search of solutions, because it can generate more diverse non-dominated solutions, which are the best solutions in MOO. This could be explained because

Table 5
Datasets main properties and *HV* reference points.

Dataset 1	20 requirements, 5 clients, 10 interactions constraints
r_{min} (*cost*, *satisfaction*) = (0, 0) r_{max} (*cost*, *satisfaction*) = (85, 893)	
Dataset 2	100 requirements, 5 clients, 44 interactions constraints
r_{min} (*cost*, *satisfaction*) = (0, 0) r_{max} (*cost*, *satisfaction*) = (1037, 2656)	

Table 6
Parameter setting for our proposal.

DEPT configuration for MONRP	Population size (*P_Size*)	40
	Crossover probability (P_X)	0.5
	Mutation factor (P_{Mut})	0.02
	Selection scheme	Rand/1/Bin

Table 7
Average *HV* and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	38.881% ± 1.27e−5	10.283% ± 6.57e−2	9.015% ± 1.12	7.708% ± 3.66e−1
50%	50.112% ± 1.62e−4	23.912% ± 6.75e−2	20.652% ± 1.60	19.114% ± 3.50e−1
70%	58.954% ± 2.24e−4	38.464% ± 7.08e−2	32.157% ± 2.30	32.242% ± 4.96e−1
Without effort limit	60.776% ± 1.03e−3	–	–	–

Table 8
Average *HV* and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	36.508% ± 6.01e−3	8.517% ± 6.21e−2	7.920% ± 2.49e−1	4.088% ± 8.55e−3
50%	46.650% ± 7.36e−3	19.159% ± 9.94e−2	18.006% ± 5.20e−1	15.454% ± 6.88e−2
70%	52.753% ± 4.25e−3	32.777% ± 1.14e−1	31.710% ± 8.92e−1	27.943% ± 7.50e−2
Without effort limit	58.026% ± 4.81e−3	–	–	–

the algorithm combines the information taken from three randomly chosen individuals (line 8 in Algorithm 1) with an elitism strategy that is explained in Algorithm 2 (the Pareto tournament function). Once a new trial solution is generated, the Pareto Tournament function checks whether this trial solution improves the quality of the original target solution in terms of Pareto dominance and crowding distance. If this is the case, the new solution will replace the target solution, so the population is improved with a new solution which is covering a new area of the search space in which a better solution is found. Thus, the Pareto tournament function makes our proposal to update the population with the best solutions in each generation. But in addition, the *NDS_archive* maintained by DEPT contains every non-dominated solution generated in every iteration of the algorithm (line 21 in Algorithm 1), so our proposal does not discard any high quality solution generated during the algorithm execution.

Moreover, results obtained with DEPT present very low dispersions for every effort boundary tested, so it can be said that the overall improvement achieved by our proposal is quite significant in terms of this multiobjective measure. This means that our proposal is able to explore the search space better than the other approaches published, and consequently, the solutions provided for the requirements selection problem will be of better quality. On the other hand, we can observe that GRASP is the metaheuristic which obtains the poorest results. This could be explained because GRASP is a trajectory-based metaheuristic and it does not work with a population of individuals, which is the case of the other approaches, so the exploration of the space search is more limited.

5.2.2. Spread (Δ) results

In this subsection, we focus on analyzing the Δ quality indicator for the 8 instances of the problem under study. For the spread indicator, lower values denote better results. Tables 9 and 10 summarize the results obtained attending to this indicator and compare them with other approaches recently published in the literature [22].

The tables show that our proposal is the evolutionary algorithm obtaining the best results in all the cases, with reduced standard deviations, so it can be said that the DEPT algorithm is the proposal computing the fronts with the best distribution of solutions in all the cases. This is probably due to the *NDS_archive* maintained by our proposal. This file contains every

Table 9
Average Δ and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	0.52 ± 0.02	0.52 ± 0.03	0.76 ± 0.09	0.64 ± 0.09
50%	0.48 ± 0.01	0.52 ± 0.01	0.79 ± 0.07	0.73 ± 0.07
70%	0.42 ± 0.03	0.48 ± 0.02	0.80 ± 0.07	0.69 ± 0.06
Without effort limit	0.40 ± 0.04	–	–	–

Table 10
Average Δ and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	0.56 ± 0.04	0.68 ± 0.06	0.80 ± 0.07	0.60 ± 0.04
50%	0.51 ± 0.03	0.66 ± 0.06	0.81 ± 0.06	0.74 ± 0.04
70%	0.47 ± 0.03	0.61 ± 0.06	0.77 ± 0.05	0.70 ± 0.03
Without effort limit	0.44 ± 0.04	–	–	–

Table 11
Average number of *NDS* and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	15 ± 0.00	13.66 ± 13.66	9.69 ± 2.09	11.37 ± 1.47
50%	19.76 ± 0.38	17.75 ± 0.61	11.30 ± 1.82	17.65 ± 2.22
70%	26.22 ± 2.17	20.57 ± 20.57	11.70 ± 1.90	20.26 ± 2.18
Without effort limit	30.51 ± 2.62	–	–	–

Table 12
Average number of *NDS* and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2 Effort boundary	DEPT Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	110.108 ± 5.45	47.12 ± 5.44	54.34 ± 8.51	57.99 ± 3.66
50%	123.64 ± 5.20	57.68 ± 5.69	65.54 ± 11.86	75.81 ± 5.81
70%	139.73 ± 8.32	70. 98 ± 5.27	83.32 ± 10.52	120.14 ± 7.27
Without effort limit	144.50 ± 7.16	–	–	–

non-dominated solution generated (line 21 in Algorithm 1), so no high quality solution generated by DEPT is lost, even if this solution is surpassed by other non dominated solution generated some iterations later. This is translated into a set of optimal solutions which present more variety than the results obtained by the other approaches published. We cannot compare the Pareto fronts graphically to observe this difference, because we do not have the information required of the approaches published, but the numerical results of both, the *HV* and the spread, indicate that our approach is able to compute results of more quality.

5.2.3. Number of non dominated solutions (NDS)

In multi-objective optimization, the more optimal solutions found, the better for the human expert when the selection of the final solution has to be performed. However, the optimal solutions for the problem tackled in this paper are, *a priori*, unknown. For this reason, the set of all the final high-quality solutions found by the MOEAs are not named optimal solutions, but non-dominated solutions. Tables 11 and 12 contain the average number of non dominated solutions obtained by our approach (DEPT) and by the other algorithms published in the literature for the different problem instances.

The results in Tables 11 and 12 show that DEPT obtains a higher number of non-dominated solutions in every case. DEPT algorithm uses the information of three different individuals selected randomly (line 8 in Algorithm 1) in order to generate the new trial individual. Then, the new individual is evaluated (line 17) according to the evaluation function explained in Section 3.3, and if the new generated individual is of better quality than the target individual (line 18), the new individual is included in the population. By selecting three different random individuals for each target individual being processed in each generation, it is ensured a more extensive exploration of the space search of solutions. Thus, DEPT is able to find a higher number of non dominated solutions.

The number of *NDS* found is bigger for dataset 2 in every case, because dataset 2 is more complex. However, we can also observe that differences between our proposal and the other approaches published are more significant for the more complex dataset. If fact, the number of *NDS* obtained by DEPT doubles the number of *NDS* generated by ACO for dataset 2 (Table 12).

6. Conclusions and future work

In this paper, we have studied the next release problem and we have proposed the use of an adapted Differential Evolution with Pareto Tournaments (DEPT) to tackle real instances of the problem. Specifically, we have formally defined a constrained multiobjective version of the requirement selection problem in which different types of interactions between requirements and several development effort boundaries have been considered. We have evaluated our proposal in terms of several quality indicators, by comparing the obtained results with several approaches recently published in the literature (ACO, NSGA-II, GRASP), and results show that the proposed algorithm is able to obtain the best sets of requirements for the MONRP. In other words, the DEPT algorithm obtains sets of non-dominated solutions with more solutions in the Pareto fronts, with a higher hypervolume and with a lower spread between the solutions. Thus, we have shown that our proposal can generate high quality solutions that allow software engineers to take decisions about the set of features that have to be included in the next release of a software package. Eight different problem instances taken from two real-world datasets have been used to ensure the effectiveness of our evolutionary algorithm. These datasets include different number of requirements, different clients' priorities and different requirement interactions, and all of them were previously solved by other published algorithms that we compare with in this study.

Due to the good results obtained with our proposal, as future work, it could be interesting to study the use of this search technique to larger real-world problems, or even to other Software Engineering problems. Moreover, due to the privacy policy of the software companies, it would be necessary the development of a MONRP instances generator which allows the systematic creation of realistic instances. In addition, other formulations of the problem by considering other objectives or more complex constraints and the application of other multiobjective metaheuristics are also interesting lines of future study.

References

[1] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, 2001.
[2] A.J. Bagnall, V.J. Rayward-Smith, I. Whittley, The next release problem, Inf. Softw. Technol. 43 (14) (2001) 883–890.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP Completeness, Freeman, New York, 1990.
[4] C.A.C. Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary Algorithms for Solving Multiobjective Problems, Springer, New York, 2007.
[5] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, New Jersey, 2001.
[6] M. Harman, A. Mansouri, Y. Zhang, Search based software engineering: trends, techniques and applications, ACM Comput. Surv. 45 (1) (2012) 11.
[7] K. Price, R. Storn, Differential evolution – a simple evolution strategy for fast optimization, Dr. Dobb's J. 22 (4) (1997) 18–24.
[8] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast elitist multi-objective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.
[9] J. Karlsson, Software requirements prioritizing, in: Proceedings of the Second International Conference on Requirements Engineering (RE '96), Colorado Springs, 1996, pp. 110–116.
[10] P. Baker, M. Harman, K. Steinhofel, A. Skaliotis, Search based approaches to component selection and prioritization for the next release problem, in: Proceedings of the 22nd IEEE International Conference on Software Maintenance, 2006, pp. 176–185.
[11] D. Greer, G. Ruhe, Software release planning: an evolutionary and iterative approach, Inf. Softw. Technol. 46 (4) (2004) 243–253.
[12] Y. Zhang, M. Harman, S.A. Mansouri, The multi-objective next release problem, in: Proc. of the 9th conference on Genetic and evolutionary computation (GECCO '07), New York, 2007, pp. 1129–1137.
[13] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, Y. Zhang, A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making, Requirem. Eng. J. (RE '08 Special Issue) 14 (2009) 232–245.
[14] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, Y. Zhang, Fairness analysis in requirements assignments, in: Proc. 16th IEEE Int. Requirements Engineering Conf., Washington, DC, 2008, pp. 115–124.
[15] A. Charan Kumari, K. Srinivas, M.P. Gupta, Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution, in: O. Schüze et al. (Eds.), EVOLVE – A Bridge between Probability, Springer, New York, 2013, pp. 107–120.
[16] J. Durillo, Y. Zhang, E. Alba, A.J. Nebro, A study of the bi-objective next release problem, Empirical Softw. Eng. 16 (2009) 29–60.
[17] H. Jiang, J. Zhang, J. Xuan, Z. Re, Y. Hu, A hybrid ACO algorithm for the next release problem, in: Proc. of the 2nd Intern. Conf. on Software Engineering and Data Mining, Chengdu, 2010, pp. 166–171.
[18] J. Knowles, D. Corne, The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization, in: Proc. Congr. Evol. Comput. (CEC), 1999, pp. 98–105.
[19] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, E. Alba, Mocell: a cellular genetic algorithm for multiobjective optimization, Int. J. Intell. Syst. 24 (7) (2009) 726–746.
[20] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, 2004.
[21] J. del Sagrado, I.M. del Águila, F.J. Orellana, Requirements interaction in the next release problem, in: Proc. of Genetic and Evolutionary Computation Conference (GECCO 2011), 2011, pp. 187–188.
[22] J. del Sagrado, I.M. del Águila, F.J. Orellana, Multi-objective ant colony optimization for requirements selection, J. Empirical Softw. Eng. (2014), http://dx.doi.org/10.1007/s10664-013-9287-3.
[23] J.T. Souza, C.L. Brito Maia, T.N. Ferreira, R.A. Ferreira do Carmo, M.M. Albuquerque Brasil, An ant colony optimization approach to the software release planning with dependent requirements, in: Proc. of the 3th Int. Symposium on Search Based Software Engineering (SBSE '11), 2011, pp. 142–157.
[24] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, An industrial survey of requirements interdependencies in software product release planning, in: Proceedings of 5th IEEE international symposium on requirements engineering (RE 2001), 2001, pp. 84–93.
[25] D. Zaharie, A comparative analysis of crossover variants in differential evolution, in: Proc. of Intern. Multiconference on Computer Science and Information Technology (IMCSIT), 2007, pp. 171–181.
[26] J. Demsar, Statistical comparison of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[27] E. Simmons, Requirements triage: what can we learn from a "medical" approach?, IEEE Softw 21 (4) (2004) 86–88.
[28] K.E. Wiegers, Software Requirements, Microsoft Press, Redmon, WA, 2003.
[29] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
[30] J.M. Chaves-González, M.A. Vega-Rodríguez, DNA strand generation for DNA computing by using a multi-objective differential evolution algorithm, BioSystems 116 (2014) 49–64.
[31] J.M. Chaves-González, J. Martínez-Gil, Evolutionary algorithm based on different semantic similarity functions for synonym recognition in the biomedical domain, Knowl.-Based Syst. 37 (1) (2013) 62–69.
[32] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.