Engineering Applications of Artificial Intelligence 43 (2015) 89–101

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection

José M. Chaves-González*, Miguel A. Pérez-Toledano, Amparo Navasa

Computer Science Department, University of Extremadura, Cáceres, Spain

ARTICLE INFO

Article history:
Received 2 October 2014
Received in revised form
1 April 2015
Accepted 7 April 2015
Available online 29 April 2015

Keywords:
Software requirements selection
Multi-objective evolutionary algorithm
Teaching learning based optimization
Search Based Software Engineering
Next Release Problem
Swarm intelligence

ABSTRACT

Software requirements selection is a problem which consists of choosing the set of new requirements which will be included in the next release of a software package. This NP-hard problem is an important issue involving several contradictory objectives which 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 satisfy the highest number of customer requirements. Furthermore, when managing real instances of the problem, the requirements tackled suffer interactions and other restrictions which make the problem even harder. In this paper, a novel multi-objective teaching learning based optimization (TLBO) algorithm has been successfully applied to several 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. In addition, three interaction constraints have been also managed. In this context, the original TLBO 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 in order to prove the effectiveness of the multiobjective proposal. In fact, the obtained results show that the developed algorithm performs better than other relevant algorithms previously published in the literature.

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 satisfy in an efficient way large sets of requirements by minimizing the production efforts (in time and cost). In fact, in most of cases it is not possible to develop all the new features suggested by the clients when the new release of a software product has to be produced. Software requirements optimization is an important task in Software Engineering, and especially relevant within the incremental approaches of software development, e.g. agile methodologies. In these kinds of methodologies, the software product is developed by generating releases which have to be produced in short iterative cycles and a new set of requirements, tailored to fit the needs of the clients and the development costs, is proposed in each iteration. In this context, the challenge of Software Engineering consists of defining which requirements should be developed by considering several complex factors (different clients' priorities with different importance, development efforts, 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 the Next Release Problem, NRP (Bagnall et al., 2001).

The NRP is an NP-hard problem (Garey and Johnson, 1990) 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 (Coello et al., 2007; Deb, 2001) 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) research field (Harman et al., 2012) has been proposed to deal with a real multiobjective version of the NRP (MONRP). Specifically, in this paper we introduce an adapted

* Corresponding author.
E-mail addresses: jm@unex.es (J.M. Chaves-González), toledano@unex.es (M.A. Pérez-Toledano), amparonm@unex.es (A. Navasa).

http://dx.doi.org/10.1016/j.engappai.2015.04.002
0952-1976/

Acronyms

ACO Ant Colony Optimization
∅ Empty set
AHP Analytical Hierarchy Process
$C=\{c_1, c_2,\ldots, c_m\}$ Set of m clients or customers
cl_i Client i
CWPopulation Crowding distance of the individuals in the population
Δ Spread
DE Differential evolution
d_f, d_l Euclidean distance from the first and the last solution in the Pareto front, respectively
d_i Euclidean distance between two consecutive solutions
$E=\{e_1, e_2,\ldots, e_n\}$ Set of n costs associated to the n requirements
$E(X)$, X_{Cost} Overall effort (cost) for X
GRASP Greedy Randomize Adaptive Search Procedure
HV Hypervolume
L_C Cost threshold constraint
meanInd Mean individual
$Mean_{Req}$ Number of requirements in the mean individual
MOCell Multi-objective Cellular genetic algorithm
MOEA Multi-objective evolutionary algorithm
MONRP Multi-objective NRP
MOO Multi-objective optimization
MOOP Multi-objective optimization problem
MO-TLBO Multi-objective TLBO
N Total number of solutions in the Pareto
nd Non-dominated
NDS Non-dominated solutions
NDS_archive Non-dominated solution archive
NRP Next Release Problem
NSGA-II Fast Non-dominated Sorting Genetic Algorithm
$obj1_{max}$, $obj2_{max}$ Highest values for the objectives 1 and 2
$obj1_{min}$, $obj2_{min}$ Minimal values for the objectives 1 and 2
P Population
P_Size Size of the population
PAES Pareto Archived Evolution Strategy
QFD Quality Function Deployment
$R=\{r_1, r_2,\ldots, r_n\}$ Set of n requirements
r[0,1] Random number between 0 and 1
$r_i \otimes r_j$ Exclusion interaction
$r_i \oplus r_j$ Combination interaction
$r_i \Rightarrow r_j$ Implication interaction
r_i, r_j Requirements i and j
$S(X)$, $X_{Satisfaction}$ Overall satisfaction for X
$S=\{s_1, s_2,\ldots, s_n\}$ Set of global satisfaction
SBSE Search-Based Software Engineering
SPEA-2 Strength Pareto Evolutionary Algorithm 2
Std. dev. Standard deviation
$Sum_{Req}(r)$ Addition of the requirements in r
T_{factor} Teaching factor
TLBO Teaching learning based optimization
υ, $X_{ReqsNumber}$ Total number of requirements in X
v_{ij} Importance that a requirement r_j has for a particular client c_i
$W=\{w_1, w_2,\ldots, w_m\}$ Set of clients' weights
X1CW, X2CW Crowding distance values for $X1$ and $X2$
X1Rank, X2Rank Dominance values for the individuals $X1$ and $X2$
X_{new}, X_a, X_b, $X_{new''}$, P_i, *auxInd* Individuals of the population
xTeacher High-quality individual of the population

version of the teaching learning-based optimization (TLBO) method, a very recent swarm intelligence evolutionary algorithm (Rao et al., 2012), which was adapted to obtain high-quality results of the multiobjective NRP (MONRP). We will name our proposal MO-TLBO, because some multi-objective features of well known MOEAs were wisely included to the original version of the algorithm. In addition, in order to test the accuracy of MO-TLBO, we have compared it with the multi-objective standard NSGA-II (Fast Non-dominated Sorting Genetic Algorithm) proposed by Deb et al. (2002), and other approaches proposed in other works published in the literature. As will be shown in this paper, our proposal provides high quality results, surpassing the results previously published in the literature for several instances of the problem.

The rest of the paper has been organized as follows: Section 2 discusses related work. Section 3 summarizes the basic background on the problem and the multiobjective formulation which has been proposed. Next section presents our proposal: a multi-objective teaching learning based optimization (MO-TLBO) algorithm for 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 (Garey and Johnson, 1990) which consists of selecting a set of requirements that will be developed for the next release of a software product. The problem evaluates two conflicting objectives, and both objectives have to be equally considered. In the literature, Karlsson (1996) proposed 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 need to perform huge numbers 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. (2001). SBSE is the research field in which search-based optimization algorithms are proposed to tackle problems in Software Engineering (Harman et al., 2012). The original problem proposed by Bagnall et al. (2001) 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 (Baker et al., 2006; Greer and Ruhe, 2004). 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. (2007) proposed the first multi-objective formulation for the original NRP (MONRP). This formulation tackles each objective separately, without any combination function. This feature allows the algorithm to explore non-dominated solutions (the solutions of more quality) for the problem. The works by Finkelstein et al. (2008, 2009) also include the use of 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 by (Charan Kumari et al., 2013; Durillo et al., 2009; Jiang et al., 2010), in which different multi-objective evolutionary algorithms are proposed for solving NRP, but dependencies among requirements are not considered. Thus, Durillo et al. (2009) proposed the well-known algorithms: PAES (Pareto Archived Evolution Strategy) (Knowles and Corne, 1999), NSGA-II (Fast Non-dominated Sorting Genetic Algorithm) (Deb et al., 2002), and MOCell (Multiobjective Cellular genetic algorithm) (Nebro et al., 2009) for solving the requirements selection problem. On the other hand, Jiang et al. (2010) solve the problem by using an Ant Colony Optimization (ACO) algorithm (Dorigo and Stützle, 2004). Finally, in the work by Charan Kumari et al. (2013), the authors propose a hybrid differential evolution (DE) evolutionary algorithm (Price and Storn, 1997).

However, 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. (2011, 2014), and by Souza et al. (2011). All those works propose the use of different ACO strategies to solve the problem, but only in the work by Sagrado et al. (2014), 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 any numerical comparisons with them.

In this paper, we present a multi-objective search-based approach based on the recent teaching-learning-based optimization (TLBO) evolutionary algorithm (Rao et al., 2012). TLBO has become highly popular in a very short period of time, because even though the algorithm is very recent, there are several works in which TLBO has been applied to the resolution of several optimization and engineering problems (Azizipanah-Abarghooee et al., 2012, 2014b; Niknam et al., 2012a, 2012b, 2012c, 2013a; Rao and Kalyankar, 2013; Rao and Patel, 2013a, 2013b). However, this is the first time that the algorithm is applied to SBSE. Thus, in spite of TLBO is a very recent swarm intelligence metaheuristics, its suitability has been widely tested in the literature. In addition, other swarm intelligence algorithms have been successfully applied to different engineering problems, so this fact shows the suitability of these kinds of approaches. For example, in Azizipanah-Abarghooee and Niknam (2012), an adaptation of the bat algorithm was applied for fuzzy interactive multi-objective economic–emission dispatch. The same problem was also solved with other classical swarm intelligence approach (PSO) in Azizipanah-Abarghooee and Aghaei (2011), and with a new multi-objective self-adaptive learning bat algorithm in Niknam et al. (2013b). In Azizipanah-Abarghooee et al. (2014a) a shuffled frog leaping algorithm was used for multi-objective optimal power flow. Finally, other engineering problems are solved by using swarm intelligence based the cuckoo optimization algorithm (Azizipanah-Abarghooee et al., 2014c, 2015).

3. The multi-objective software requirements selection problem

This section explains the MONRP for the selection of software requirements, 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 (Coello et al., 2007), 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).

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. In addition, the clients involved in the NRP have different categories depending on their level of importance or the category they occupy in the company. Furthermore, each requirement will also have different considerations for each client, which means that the final level of importance for a particular software requirement 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. Moreover, 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. 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

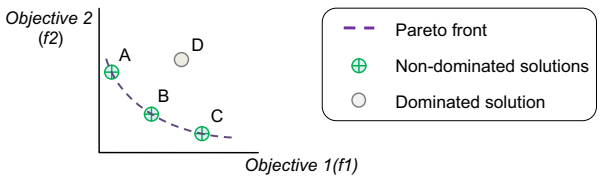

Fig. 1. Representation of dominated solution, non-dominated solutions and Pareto front.

objectives are subject to the problem *interactions* produced among the requirements, which are managed as constraints.

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 Bagnall et al. (2001). In order to perform comparisons with other works published in the literature, we have followed the same formulation proposed by Sagrado et al. (2014) and used in a previous work by Chaves-González and Pérez-Toledano (2015).

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 represent 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 the following equation :

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

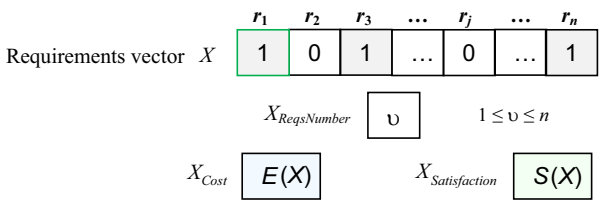

Fig. 2. Data structure for the individual.

Thus, MONRP can be formulated as expressed in the following equation :

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

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 TLBO 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 multi-objective 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 TLBO algorithm with Pareto tournament

In this paper, a multi-objective swarm intelligence evolutionary algorithm which is based on the behavior of the different participants in a classroom is proposed to solve the requirements selection problem. Teaching-learning based optimization (TLBO) is a novel swarm intelligence metaheuristics proposed by Rao et al. (2012). The algorithm population is considered as a class of students in which the individuals evolve together in two stages. During the first stage, called *teacher phase*, the population learns from the best individuals in the population (they play the role of teacher). Then, in the *learner phase*, all the individuals learn from their partners. We have adapted the algorithm to the features of the problem tackled. Thus, as each individual contains a vector of n requirements (see Fig. 2), these n requirements will represent the n subjects that each individual has to study.

As was explained in the previous section, the problem has been formulated as a MOOP, so the original TLBO scheme was modified to work with the MOOP tackled in this study. Thus, in this paper, we propose the use of a multiobjective version of TLBO (MO-TLBO) to solve the software requirement selection problem for different

real instances. One of the best features of the variant developed is that the only two parameters to be configured are the population size and the stop condition for the algorithm. Moreover, as we propose an improved multiobjective formulation for the problem, we have included to the original design of the algorithm some multiobjective features from well-known MOEAs. Thus, from NSGA-II (Deb et al., 2002), 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 (Knowles and Corne, 1999), 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 the MO-TLBO algorithm developed is shown in Algorithm 1.

Algorithm 1. Pseudocode of MO-TLBO

1: $P \Leftarrow$ randomGenerationOfInitialPopulation (P_Size)
2: $P \Leftarrow$ evaluate Initial Population (P)
3: $NDS\_archive \Leftarrow \varnothing$
4: **while** not stop condition satisfied **do**
5: $meanInd \Leftarrow$ calculateMeanIndividual (P)
6: $meanInd' \Leftarrow$ correctSolution ($meanInd$)
7: **for** $i=1$ to P_Size do
8: $xTeacher \Leftarrow$ selectTeacher (P)
9: $X_{new} \Leftarrow$ teacherPhase (P_i, $xTeacher$, $meanInd$ ')
10: $X_a, X_b \Leftarrow$ selectRamdomIndividuals (P)
11: $X_{new}' \Leftarrow$ learnerPhase (X_{new}, X_a, X_b)
12: $X_{new}'' \Leftarrow$ correctSolution (X_{new}')
13: $X_{new}'' \Leftarrow$ evaluateIndividual (X_{new}'')
14: $P_i \Leftarrow$ paretoTournamet (X_{new}'', P_i)
15: **end for**
16: $NDS\_archive \Leftarrow$ updateNDS_archive(P)
17: **end while**

The algorithm starts with the random generation of P_Size solutions (line 1). Each solution includes σ requirements (see Fig. 2). These solutions are valid individuals which satisfy the problem constraints (see Section 3.2). After that, each individual is evaluated according to the formulation explained in Section 3.3 (line 2). Thus, the P_Size solutions are classified according to different Pareto fronts (see Section 3.1) by ranking and afterwards, each solution is sorted according to non-dominated ranks (line 3). To this respect, it is said that *a particular solution is of more quality than another if it is dominated by fewer solutions*, and in multiobjective optimization, it is known that *a particular solution dominates another if the solution is better, at least, in one of the objectives and it is not worse in any of the others*. In case that two solutions present equal ranks, they are classified according to the crowding distance (Deb et al., 2002). This multiobjective measure is also calculated for each solution at the beginning of the algorithm (line 2). Thus, *a particular solution is better than another if it has the same or a fewer value of dominance ranking and it presents higher crowding distance*, because solutions which are located in lesser crowded regions are preferred (Deb et al., 2002). Finally, previous to the main loop of the algorithm, the set of non-dominated solutions (*NDS_archive*), which will contain the best solutions found along the algorithm execution, is initialized (line 3).

After that, in each generation (until the stop condition is satisfied), the algorithm creates a mean individual (*meanInd*). This individual represents the average behavior of the classroom, or in other words, the subjects studied by this student are the most popular subjects among the students in the population. This is translated into an individual with the requirements which are most widely included in the solutions of the population. An illustrative example of the generation of the mean individual is shown in Fig. 3.

However, it is very likely that the resultant mean individual does not satisfy the problem constraints. Therefore, *meanInd* have to be corrected (line 6) in case that any constraint is not satisfied. This feature makes our proposal to be very fast, in comparison with other approaches published, as will be discussed in Section 5.2. Fig. 4 shows an example for the process of individual verification and correction. As can be observed, the *meanInd* individual must satisfy two constraints (Fig. 4). 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. 4 it 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 *meanInd*' 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, *meanInd*', 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.

The valid *meanInd'* individual will be used in the teacher phase along with one individual selected randomly from the non-dominated solutions front (line 8). This non-dominated solution (*xTeacher*) plays the role of teacher in the teacher phase of the algorithm (line 9), which generates a new individual X_{new} by using the expression defined in (4). Since the individual X_{new} is

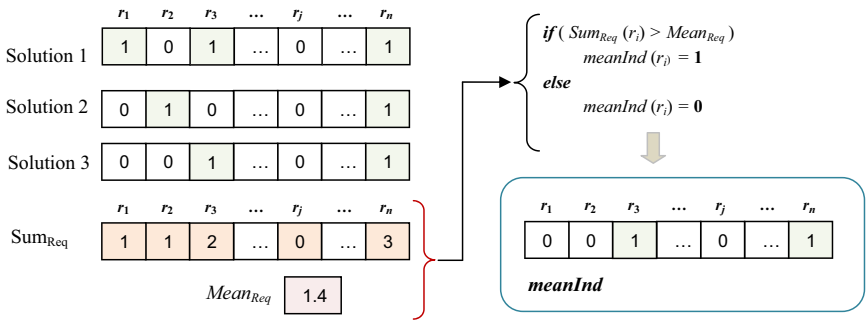

Fig. 3. Illustration of the generation of the *meanInd* solution with a classroom of 3 individuals.

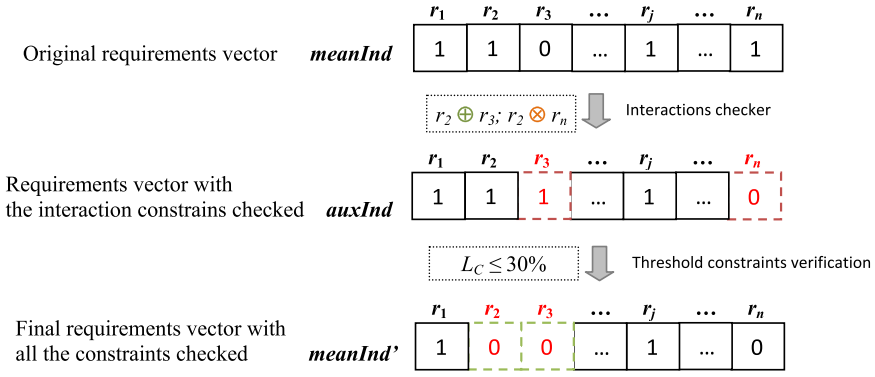

Fig. 4. Procedure for the constraints verification and correction of the individual.

composed of n requirements (considered as subjects), the expression in (4) is repeated for each requirement within the individual. Thus, for each subject, a particular student P_i, can learn more from its teacher (*xTeacher*), or can learn more from the average individual (*meanInd′*), or even can learn nothing.

$$X_{new} \Leftarrow P_i + r[0,1] \cdot (xTeacher - T_{factor}[0,1] \cdot meanInd') \quad (4)$$

where P_i is the current individual, $r[0,1]$ is a value randomly generated that can be either 0 or 1. If r is equal to 0, the requirement of X_{new} being considered is directly taken from P_i, which means that the student P_i does not learn anything of that particular subject. On the other hand, if r is equal to 1, the requirement of X_{new} will be formed by using *xTeacher* and *meanInd′*. Finally, $T_{factor}[0,1]$ is the teaching factor used for the algorithm to determine the influence of *meanInd′* in the learning process. T_{factor} is also randomly generated and only takes two values (0 or 1). If T_{factor} is equal to 0, X_{new} will take the requirement being studied directly from *xTeacher*. On the contrary, if T_{factor} is equal to 1, X_{new} will take the information from *meanInd′*.

After the teacher phase, two individuals X_a, X_b, are randomly chosen from the population (line 10) and the learner phase of the algorithm starts (line 11). In this stage, each requirement of X_{new} can be modified according to the individuals X_a and X_b according to the expression in (5). Similarly to the teacher phase, the learning process is performed for each requirement of the individual.

$$X_{new}' \Leftarrow X_{new} + r[0,1] \cdot \text{Crossover}(X_a, X_b) \quad (5)$$

X_{new} is the individual generated after the teacher phase. $r[0,1]$ is a value randomly generated that can be either 0 or 1. If r is equal to 0, the requirement of X_{new}' under construction is directly taken from X_{new}. On the other hand, if r is equal to 1, the requirement X_{new}' under construction will be generated by using a uniform crossover operation between X_a and X_b.

After the learning phase, the individual X_{new}' is verified and corrected (line 12) with the procedure previously explained (Fig. 4). The new improved individual X_{new}'' is evaluated according to the multiobjective formulation explained in Section 3.3 (line 13). Then, X_{new}'' is compared with the original individual P_i (line 14) and the algorithm selects the best one to be part of the population for the next generation. Related to this part of the algorithm, since we are in a multiobjective environment, the algorithm determines the best individual by using the Pareto tournament function explained in Algorithm 2. The winner of the Pareto tournament will remain in the next generation of the algorithm.

Finally, the *NDS_archive* (non-dominated-solutions archive) is updated by saving the best ranked solutions in the population (line 16). When the algorithm finishes, this record will be processed to obtain the best ranked solutions generated in the global process.

Algorithm 2. Pseudocode for the Pareto tournament function

```
1:  Input: Individuals X1 and X2
2:  tournamentWinner ⇐ X2
3:  if X1 ≠ X2 then
4:    X1Rank ⇐ calculateIndividualDominance (X1)
5:    X2Rank ⇐ calculateIndividualDominance (X2)
6:    if X1Rank < X2Rank then
7:      tourmamentWinner ⇐ X1
8:    else if X1Rank = X2Rank then
9:      CWPopulation ⇐ calculateCrowdingPopulation (X1,
X2)
10:       X1CW ⇐ calculateCrowdingDistance (CWPopulation,
X1)
11:       X2CW ⇐ calculateCrowdingDistance (CWPopulation,
X2)
12:       if X1CW > X2CW then
13:         tourmamentWinner ⇐ X1
14:       end if
15:     end if
16:   end if
17:   return tournamentWinner
```

Algorithm 2 summarizes the Pareto tournament function used by our proposal. This function selects the best individual between two given, X_1 and X_2. If both individuals are different, their rank is calculated (lines 4 and 5 in Algorithm 2) by considering their dominance. In this context, a given individual is better than another if it is dominated by fewer individuals. As was mentioned, a particular individual dominates another if it is better, at least, in one of the objectives and it is not worse in any of the others. If the rank of X_1 is lower than the obtained by X_2, the winner of the Pareto tournament is X_1. In case of a tie (line 8), the algorithm uses the crowding distance. Then, a population with the solutions that belong to the X_1 and X_2 Pareto fronts is generated (line 9), and after that, the crowding distance for both individuals is calculated (lines 10, 11) by using this new population (*CWPopulation*), as is explained in Deb et al. (2002). If the crowding distance of X_1 is higher than that obtained by X_2, X_1 is the winner of the Pareto tournament. Otherwise, the winner of the Pareto tournament is X_2.

Algorithm 3. Pseudocode of SPEA2

```
1:  P ⇐ generateInitialPopulation (P_Size)
2:  A ⇐ ∅ //Archive of A_Size individuals
3:  while not stop condition satisfied do
4:    FitnessAssignment (P, A)
5:    EnvironmentalSelection (A, P) //Truncate A if necessary
6:    for i=1 to P_Size do
```

7: $X_a, X_b \Leftarrow$ binaryTournament (A) // $X_a \neq X_b$
8: $P_i \Leftarrow$ recombination (X_a, X_b, Cr)
9: $P'_i \Leftarrow$ randomMutation (P_i, f)
10: **end for**
11: **end while**

4.3. Strength Pareto Evolutionary Algorithm 2 (SPEA2)

In this paper, we also include an adapted version of SPEA2 (Strength Pareto Evolutionary Algorithm 2) to provide a wider comparison of the results obtained with our proposal. SPEA2 is a population-based algorithm originally created by Zitzler et al. (2002). The pseudocode of our version of this MOEA is shown in Algorithm 3. SPEA2 uses a regular population *P*, of *P_Size* individuals, and an archive *A*, which is an external set of *A_Size* individuals. A fitness value which is the sum of its strength raw fitness plus a density estimation is assigned to each individual in *P* and in *A* (line 4 in Algorithm 3). After that, the non-dominated individuals (the best ones) of both, the population and the archive, are copied into a new population, truncating it for avoiding to exceed the size of the population (line 5). Then, the algorithm also applies selection, crossover and mutation operators (lines 7–9) to generate the new population. This population is also checked (as in the MO-TLBO algorithm) in order to verify that every individual satisfies the problem constraints (see Fig. 4).

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. After that, we compare the results generated by MO-TLBO with the results provided by the MO reference SPEA2, and finally, we compare the results obtained in this study with a results obtained in a previous study in which the problem was solved with a MO version of the differential evolution algorithm.

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 (Demsar, 2006). 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 (2004), and it includes 20 requirements and 5 clients. Table 1 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 1 and 2) 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 3. The values in this table are between 1 (the less important client) and 5 (the most important client).

The dataset presented in Table 1 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 studies (Sagrado et al., 2014), 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. (2014), and it includes 100 requirements, 5 clients and 44 requirements interactions. 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 relative importance for each client is shown in the second row of Table 3. 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 (Schwaber and Beedle, 2001). 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

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

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

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

Table 3
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

scale. Thus, clients simply place the requirements into one of these three categories (Simmons, 2004; Wiegers, 2003): (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 four 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 *running time* of the algorithm. Execution times are included in order to provide a measure of efficiency for the algorithm. Third, the hypervolume (*HV*) indicator (Zitzler and Thiele, 1999). This measure calculates the volume covered by members of a non-dominated set of solutions Q (see Eq. 6). *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, and due to the problem tackled includes two objectives, these points are: r_{min} ($obj1_{min}$, $obj2_{min}$) and r_{max} ($obj1_{max}$, $obj2_{max}$). 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 4.

$$HV = volume(\cup_{i=1}^{|Q|} v_i) \tag{6}$$

The fourth 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 following equation :

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{d_f + d_l + (n-1)d} \quad (7)$$

where d_i is the Euclidean distance between two consecutive solutions, $\overline{d}$ 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. The last quality indicator considered was chosen in order to provide a measure in terms of the efficiency obtained by the proposal, so the running time of the algorithm is also included.

In order to perform fair comparisons, we have established the same stop condition for our proposal than in previously published works: 10,000 evaluations of the fitness function (Sagrado et al., 2014). Finally, as was explained in Section 4.2, the only parameter of our proposal is the population size (*P_Size*), which was empirically set to 40 individuals. This feature is an advantage of our proposal, because the algorithm does not require a complex configuration procedure for obtaining high-quality results with different problem instances.

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. Then, we perform a comparative study on the number of non-dominated solutions (*NDS*) obtained by different approaches, and finally, we compare the execution time of different algorithms.

Table 4
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)

5.2.1. Hypervolume (HV) results

Tables 5 and 6 summarize the comparative results in terms of the *HV* indicator. We compare the results generated by our proposal, MO-TLBO, with other algorithms published in the literature (ACO, NSGA-II and GRASP). 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 MO-TLBO is the algorithm which obtains the best results regarding to this indicator for every problem instance.

Moreover, results obtained with MO-TLBO 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 MO-TLBO 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 7 and 8 summarize the results obtained attending to this indicator and compare them with other approaches published in the literature.

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 MO-TLBO is the algorithm computing the fronts with the best distribution of solutions in all the cases. 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.

Table 5
Average *HV* and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1 Effort boundary	MO-TLBO Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	42.981% ± 1.10e−5	10.283% ± 6.57e−2	9.015% ± 1.12	7.708% ± 3.66e−1
50%	56.219% ± 2.61e−4	23.912% ± 6.75e−2	20.652% ± 1.60	19.114% ± 3.50e−1
70%	62.837% ± 3.24e−4	38.464% ± 7.08e−2	32.157% ± 2.30	32.242% ± 4.96e−1
Without effort limit	66.562% ± 1.14e−3	–	–	–

Table 6
Average *HV* and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2 Effort boundary	MO-TLBO Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	43.182% ± 1.01e−2	8.517% ± 6.21e−2	7.920% ± 2.49e−1	4.088% ± 8.55e−3
50%	53.122% ± 1.26e−2	19.159% ± 9.94e−2	18.006% ± 5.20e−1	15.454% ± 6.88e−2
70%	59.992% ± 4.56e−3	32.777% ± 1.14e−1	31.710% ± 8.92e−1	27.943% ± 7.50e−2
Without effort limit	64.126% ± 4.94e−3	–	–	–

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 9 and 10 contain the average number of non-dominated solutions obtained by our approach (MO-TLBO) and by the other algorithms published in the literature for the different problem instances.

The results in the tables show that MO-TLBO obtains a higher number of non-dominated solutions in every case. 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 MO-TLBO doubles the number of *NDS* generated by ACO for dataset 2 (Table 10).

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

Dataset 1 Effort boundary	MO-TLBO Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	0.52 ± 0.01	0.52 ± 0.03	0.76 ± 0.09	0.64 ± 0.09
50%	0.46 ± 0.01	0.52 ± 0.01	0.79 ± 0.07	0.73 ± 0.07
70%	0.40 ± 0.02	0.48 ± 0.02	0.80 ± 0.07	0.69 ± 0.06
Without effort limit	0.38 ± 0.04	–	–	–

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

Dataset 2 Effort boundary	MO-TLBO Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	0.45 ± 0.02	0.68 ± 0.06	0.80 ± 0.07	0.60 ± 0.04
50%	0.41 ± 0.02	0.66 ± 0.06	0.81 ± 0.06	0.74 ± 0.04
70%	0.36 ± 0.02	0.61 ± 0.06	0.77 ± 0.05	0.70 ± 0.03
Without effort limit	0.34 ± 0.03	–	–	–

Table 9
Average number of *NDS* and standard deviation of the results for the 4 instances of dataset 1.

Dataset 1 Effort boundary	MO-TLBO 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%	24.16 ± 0.18	17.75 ± 0.61	11.30 ± 1.82	17.65 ± 2.22
70%	32.95 ± 1.12	20.57 ± 20.57	11.70 ± 1.90	20.26 ± 2.18
Without effort limit	41.85 ± 1.36	–	–	–

5.2.4. Execution time

The hardware and software conditions for the algorithms published in the literature are not provided, so we cannot directly compare the execution times for our proposal and the other approaches published. However, in this subsection we show the running time of our proposal to provide the reader with a general idea of the execution times of the different proposals which tackle the problem. In our case, as was mentioned previously, MO-TLBO has been executed on an Intel Xeon 2.33 GHz processor with 4 GB RAM, and the running times for dataset 1 were smaller than 260 ms in every case. On the other hand, processing dataset 2 took less than 1800 ms in all cases.

In case of the algorithms proposed in the literature (Sagrado et al., 2014), the fastest algorithm for dataset 1 is the ACO, with execution times between 600 and 800 ms approximately. Therefore, it could be said that our proposal is at least twice faster than the fastest algorithm referred for dataset 1. On the other hand, the fastest algorithm proposed for dataset 2 is NSGA-II, which is not the one obtaining the best results (as observed in Tables 5–10), with execution times between 28 and 38 s, approximately. Moreover, the ACO algorithm, which is the most competitive published proposal, takes longer than 10 min to obtain competitive results. Thus, our proposal is at least 10 times faster than the fastest algorithm, and greatly faster (in other order of magnitude) than the best algorithm proposed in the literature for the problem tackled. Anyway, as was said before, this cannot be considered a rigorous comparative study, because we do not know the hardware and software conditions in which the aforementioned algorithms were executed, but given the conditions of our study, which can be reached by any user or company nowadays, we can conclude that our proposal has very competitive execution times.

5.3. Multi-objective comparison between MOEAs

This subsection presents a direct comparison of our proposal (MO-TLBO) with a multiobjective reference algorithm: SPEA2. The main goal of this comparison is to show the strength of our proposal for solving the MONRP compared with a well known standard algorithm in the multiobjective domain. We will see how MO-TLBO achieves better results than SPEA2. In addition, as mentioned in Section 5.1, the only parameter of MO-TLBO is the population size (*P_Size*), and SPEA2 include several parameters which have to be tuned for the specific problem tackled (see Table 11). This is one of the main advantages of our approach.

With the aim of making a fair comparison, we have tuned the parameters of each algorithm separately, as well as performing a statistical study to ensure that the results obtained in each experiment were statistically relevant. Table 11 summarizes the best parameter configuration found for each algorithm.

Table 10
Average number of *NDS* and standard deviation of the results for the 4 instances of dataset 2.

Dataset 2 Effort boundary	MO-TLBO Mean ± Std. dev.	ACO Mean ± Std. dev.	NSGA-II Mean ± Std. dev.	GRASP Mean ± Std. dev.
30%	129.15 ± 7.57	47.12 ± 5.44	54.34 ± 8.51	57.99 ± 3.66
50%	136.13 ± 7.60	57.68 ± 5.69	65.54 ± 11.86	75.81 ± 5.81
70%	144.85 ± 7.93	70. 98 ± 5.27	83.32 ± 10.52	120.14 ± 7.27
Without effort limit	152.55 ± 7.90	–	–	–

In the first place we compare the MOEAs by using the *HV* indicator. In Table 12, we present a numerical view of the average hypervolume of 31 independent executions obtained by the algorithms MO-TLBO and SPEA2 for the two datasets managed in this study. If we focus on Table 12, we notice that the MO-TLBO approach obtains results of more quality in terms of the *HV* metrics than SPEA2 for all data sets and effort boundaries. We can also observe that differences are quite significant in all cases. In addition, we notice that standard deviations for all experiments are minimal, so, we conclude that our proposal explores the search space of solutions more efficiently than the multiobjective standard SPEA2.

On the other hand, we also performed a comparison among the developed MOEAs by using the *Set Coverage* indicator. This multiobjective metrics allows the comparison of two multiobjective algorithms in terms of Pareto dominance. Table 13 shows a coverage relation by pairs between our proposal (MO-TLBO) and the standard SPEA2. As can be observed, the non-dominated solutions (*NDS*) provided by MO-TLBO cover more *NDS* provided by SPEA2 than in the contrary for the two datasets under study. In fact, the *ND* solutions obtained by MO-TLBO cover all the *ND* solutions of SPEA2 for the two datasets. MO-TLBO is able to cover 100% of solutions obtained by SPEA2. In contrast, the SPEA2 algorithm only covers, in the best case, 4.3% of the solutions generated by the proposed MO-TLBO. Therefore, the set coverage results indicate that the Pareto fronts obtained with MO-TLBO are of better quality than Pareto fronts obtained with SPEA2.

To summarize, after analyzing the results we can say that MO-TLBO results surpass the results provided by the standard SPEA2 algorithm. Therefore, in a global view, MO-TLBO algorithm seems to be a very promising approach to solve the NRP.

5.4. Comparison with a previous work

In this subsection, we want to show a direct comparison of the results obtained with our proposal and the results published in a previous stage of our work (Chaves-González and Pérez-Toledano, 2015). That work is a first attempt to the resolution of the problem with a classical MOEA: the differential evolution (DE) algorithm. Both, the analysis of the results and the state-of-the-art study introduced in this paper are more complete. In addition, the results obtained with our swarm intelligence proposal (MO-TLBO) are substantially better than the results provided by the differential evolution. We show the comparison between both MOEAs by using the hypervolume indicator in Table 14. As in previous subsections, we present a numerical view of the average hypervolume of 31 independent executions. The first column corresponds to the results obtained by our proposal (MO-TLBO), and in the second column we show the results published in (Chaves-González and Pérez-Toledano, 2015). Every row in the table corresponds to an instance of the two datasets managed in this study.

If we focus on Table 14, we notice that the MO-TLBO approach obtains results of more quality in terms of the *HV* metrics than DE. In fact, results provided by DE are closer to the results generated by SPEA2 (see Table 12) than to the results obtained with our proposal for all data sets and effort boundaries. We can also observe that differences are quite significant in all cases. In addition, we notice that standard deviations for all experiments are minimal, so, we conclude that MO-TLBO obtains significantly better results for the Next Release Problem than the previously published DE algorithm.

Table 11
MOEA configurations for the multi-objective Next Release Problem.

Parameters for MO-TLBO	
Population size (*P_Size*)	40
Parameters for SPEA2	
Population size (*P_Size*)	40
Archive size (*A_size*)	*P_Size*
Crossover probability (*Cr*)	0.80
Mutation probability (*f*)	0.30
Parent choice scheme	Binary tournament

Table 12
Comparison between MO-TLBO and H-SPEA2 by using average *HV* of 31 independent runs.

Datasets and effort boundaries	MO-TLBO, Mean ± Std. dev	SPEA2, Mean ± Std. dev.
Dataset 1 (30%)	42.981% ± 1.10e−5	32.888% ± 1.56e−2
Dataset 1 (50%)	56.219% ± 2.61e−4	47.112% ± 2.45e−2
Dataset 1 (70%)	62.837% ± 3.24e−4	53.464% ± 3.98e−2
Dataset 1 (no limit)	66.562% ± 1.14e−3	58.559% ± 1.43e−3
Dataset 2 (30%)	43.182% ± 1.01e−2	33.113% ± 2.01e−2
Dataset 2 (50%)	53.122% ± 1.26e−2	43.951% ± 2.23e-2
Dataset 2 (70%)	59.992% ± 4.56e−3	50.538% ± 3.98e−3
Dataset 2 (no limit)	64.126% ± 4.94e−3	55.966% ± 3.81e−3

Table 13
Coverage relation between MO-TLBO and SPEA2.

Dataset	*SC* (MO-TLBO, SPEA2)	*SC* (SPEA2, MO-TLBO)
Dataset 1 (no effort limit)	**100%**	4.3%
Dataset 2 (no effort limit)	**100%**	6.5%

Table 14
Comparison between MO-TLBO and DE by using average *HV* of 31 independent runs.

Datasets and effort boundaries	MO-TLBO, Mean ± Std. dev.	DE (Chaves-Gonzalez and Pérez-Toledano, 2015), Mean ± Std. dev.
Dataset 1 (30%)	42.981% ± 1.10e−5	38.881% ± 1.27e−5
Dataset 1 (50%)	56.219% ± 2.61e−4	50.112% ± 1.62e−4
Dataset 1 (70%)	62.837% ± 3.24e−4	58.954% ± 2.24e−4
Dataset 1 (no limit)	66.562% ± 1.14e−3	60.776% ± 1.03e−3
Dataset 2 (30%)	43.182% ± 1.01e−2	36.508% ± 6.01e−3
Dataset 2 (50%)	53.122% ± 1.26e−2	46.650% ± 7.36e−3
Dataset 2 (70%)	59.992% ± 4.56e−3	52.753% ± 4.25e−3
Dataset 2 (no limit)	64.126% ± 4.94e−3	58.026% ± 4.81e−3

6. Conclusions and future work

In this paper, we have proposed the use of a novel multiobjective swarm intelligence approach (MO-TLBO) with Pareto tournament to tackle real instances of the Next Release Problem. The paper includes a constrained multiobjective formulation of the 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 results generated by our proposal with several approaches published in the literature

(ACO, NSGA-II, GRASP, SPEA2, DE). After analyzing the results, we can conclude that MO-TLBO is able to obtain the best sets of requirements, because MO-TLBO generates sets of non-dominated solutions with more solutions in the Pareto fronts (Tables 9 and 10), with a higher hypervolume (Tables 5, 6, 12 and 14) and with a lower spread between the solutions (Tables 7 and 8) than the rest of approaches. Thus, the results given in the paper show that our proposal can efficiently 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 were used to check the effectiveness of our evolutionary algorithm. These datasets included different numbers of requirements, client priorities, and requirement interactions, and both of them had previously been employed by other published work so that we were able to make comparisons with the results of the present study.

Due to the good results obtained with MO-TLBO, in future work, it might be interesting to work with other multiobjective approaches based on swarm intelligence that can be applied to the problem. An example might be a hybrid version of our proposal with some other multiobjective approach. It could also be interesting to study the use of this search technique applied to larger real-world datasets. To this end, it would be necessary to develop a MONRP dataset generator for the systematic generation of instances. This improvement should be addressed with the help of software design experts, because the question of the interactions among the requirements in a complex software package is in no way trivial. Other formulations of the problem, considering other objectives or more complex constraints, would also be interesting lines for future work.

Acknowledgments

This research was partially funded by the Spanish Ministry of Science and Innovation under the contract TIN2012-34945, Department of Employment, Enterprise and Innovation of the Government of Extremadura and European Regional Development Fund (GR10129).

References

Azizipanah-Abarghooee, R., Aghaei, J., 2011. Stochastic dynamic economic emission dispatch considering wind power. In: Proceedings of the Power Engineering and Automation Conference, Wuhan, China, pp. 158–161.

Azizipanah-Abarghooee, R., Narimani, M.R., Bahmani-Firouzi, B., Niknam, T., 2014a. Modified shuffled frog leaping algorithm for multi-objective optimal power flow with FACTS devices. J. Intell. Fuzzy Syst. 26, 681–692.

Azizipanah-Abarghooee, R., Niknam, T., 2012. A new improved bat algorithm for fuzzy interactive multi-objective economic/emission dispatch with load and wind power uncertainty. In: Proceedings of the 10th International FLINS Conference, Istanbul, Turkey, pp. 388.

Azizipanah-Abarghooee, R., Niknam, T., Bina, M.A., Zare, M., 2015. Coordination of combined heat and power-thermal-wind-photovoltaic units in economic load dispatch using chance-constrained and jointly distributed random variables methods. Energy 79, 50–67.

Azizipanah-Abarghooee, R., Niknam, T., Bavafa, F., Zare, M., 2014b. Short-term scheduling of thermal power systems using hybrid gradient based modified teaching-learning optimizer with black hole algorithm. Electr. Power Syst. Res. 108, 16–34.

Azizipanah-Abarghooee, R., Niknam, Roosta, A., Malekpour, A.R., Zare, M., 2012. Probabilistic multiobjective wind-thermal economic emission dispatch based on point estimated method. Energy 37, 322–335.

Azizipanah-Abarghooee, R., Niknam, T., Zare, M., Gharibzadeh, M., 2014c. Multi-objective short-term scheduling of thermoelectric power systems using a novel multi-objective improved cuckoo optimisation algorithm. IET Gener., Transm. Distrib. 8, 873–894.

Bagnall, A.J., Rayward-Smith, V.J., Whittley, I., 2001. The next release problem. Inf. Softw. Technol. 43 (14), 883–890.

Baker, P., Harman, M., Steinhofel, K., Skaliotis, A., 2006. Search based approaches to component selection and prioritization for the next release problem. In: Proceedings of the 22nd IEEE International Conference on Software Maintenance, Washington DC, pp. 176–185.

Charan Kumari, A., Srinivas, K., Gupta, M.P., 2013. Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution. In: Schüze, O., et al. (Eds.), EVOLVE – A Bridge between Probability, Set Oriented Numerics,and Evolutionary Computation II AISC, 175. Springer, New York, pp. 107–120.

Chaves-González, J.M., Pérez-Toledano, M.A., 2015. Differential evolution with Pareto tournament for the multi-objective next release problem. Appl. Math. Comput. 252, 1–13.

Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V., 2007. Evolutionary Algorithms for Solving Multiobjective Problems. Springer, New York.

Deb, K., 2001. Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, New Jersey.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197.

Demsar, J., 2006. Statistical comparison of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge.

Durillo, J., Zhang, Y., Alba, E., Nebro, A.J., 2009. A study of the bi-objective next release problem. Empir. Softw. Eng. 16, 29–60.

Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y., 2009. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir. Eng. J. (RE '08 Special Issue) 14, 232–245.

Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y., 2008. Fairness analysis in requirements assignments. In: Proceedings of the 16th IEEE International Requirements Engineering Conference, Washington DC, pp. 115–124.

Garey, M.R., Johnson, D.S., 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.

Harman, M., Mansouri, A., Zhang, Y., 2012. Search based software engineering: trends, techniques and applications. ACM Comput. Surv. 45 (1), 11.

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, Chengdu, pp. 166–171.

Karlsson, J., 1996. Software requirements prioritizing. In: Proceedings of the Second International Conference on Requirements Engineering (RE '96), Colorado Springs, pp. 110–116.

Knowles, J., Corne, D., 1999. The pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC), pp. 98–105.

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

Niknam, T., Azizipanah-Abarghooee, R., Aghaei, J., 2013a. A new modified teaching-learning algorithm for reserve constrained dynamic economic dispatch. IEEE Trans. Power Syst. 28, 749–763.

Niknam, T., Azizipanah-Abarghooee, R., Narimani, M.R., 2012a. A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems. Eng. Appl. Artif. Intell. 25, 1577–1588.

Niknam, T., Azizipanah-Abarghooee, R., Narimani, M.R., 2012b. An efficient scenario-based stochastic programming framework for multi-objective optimal micro-grid operation. Appl. Energy 99, 455–470.

Niknam, T., Azizipanah-Abarghooee, R., Zare, M., Bahmani-Firouzi, B., 2013b. Reserve constrained dynamic environmental/economic dispatch: a new multi-objective self-adaptive learning bat algorithm. IEEE Syst. J. 7, 763–776.

Niknam, T., Golestaneh, F., Sadeghi, M.S., 2012c. θ – multiobjective teaching-learning-based optimization for dynamic economic emission dispatch. IEEE Syst. J. 6 (2), 341–352.

Price, K., Storn, R., 1997. Differential evolution – a simple evolution strategy for fast optimization. Dr. Dobb's J. 22 (4), 18–24.

Rao, R.V., Savsani, V., Vakharia, D., 2012. Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf. Sci. 183 (1), 1–15.

Rao, R.V., Kalyankar, V.D., 2013. Parameter optimization of modern machining processes using teaching–learning-based optimization algorithm. Eng. Appl. Artif. Intell. 26, 524–531.

Rao, R.V., Patel, V., 2013a. Comparative performance of an elitist teaching-learning-based optimization algorithm for solving unconstrained optimization problems. Int. J. Ind. Eng. Comput. 4 (1), 29–50.

Rao, R.V., Patel, V., 2013b. Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based-optimization algorithm. Eng. Appl. Artif. Intell. 26, 430–445.

Sagrado, J., del Águila, I.M., Orellana, F.J., 2011. Requirements interaction in the next release problem. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 187–188.

Sagrado, J., del Águila, I.M., Orellana, F.J., 2014. Multi-objective ant colony optimization for requirements selection. J. Empir. Softw. Eng. 1, 1–34. http://dx.doi.org/10.1007/s10664-013-9287-3.

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

Schwaber, K., Beedle, M., 2001. Agile Software Development with Scrum. Prentice Hall, New Jersey, United States.

Souza, J.T., Brito Maia, C.L., Ferreira, T.N., Ferreira do Carmo, R.A., Albuquerque Brasil, M. 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 (SBSE '11), pp. 142–157.

Wiegers, K.E., 2003. Software Requirements. Microsoft Press, Redmon. WA, USA.

Zhang, Y., Harman, M., Mansouri, S.A., 2007. The multi-objective next release problem. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation (GECCO '07), New York, pp. 1129–1137.

Zitzler, E., Laumanns, M., Thiele, L., 2002. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Proceedings of EUROGEN'02, pp. 95–100.

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.