Soft Computing
https://doi.org/10.1007/s00500-018-3553-7

METHODOLOGIES AND APPLICATION

Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm

Hamidreza Alrezaamiri[1] · Ali Ebrahimnejad[2] · Homayun Motameni[3]

Abstract
In agile methods, software products are developed in several releases. In each release, a new set of requirements for development is proposed. Due to technical and non-technical problems, it is almost impossible to develop all the proposed requirements in the next release. The select of an optimal subset from among all the proposed requirements has become an important problem for the developer team. The aim of this problem is to select an optimal subset from among the requirements for product development in the next release so that it has the highest satisfaction to the clients and the lowest cost for the manufacturing company. Since this problem faces two conflicting objectives and several constraints, it is placed in the NP-hard problems category. In this paper, we intend to formulate this problem for the first time as a fuzzy multi-objective optimization problem. We intend to use an artificial chemical reaction optimization algorithm to solve this problem. In the implementation stage, we make use of five interactions between requirements as one of the constraints of the problem for the first time. Two randomized fuzzy synthetic datasets are used to do the experiments. The results of the proposed algorithm are evaluated using three criteria of multi-objective problems. The results and diagrams of the proposed algorithm are very reliable and can help the developer team to make a decision.

Keywords Next release problem · Fuzzy numbers · Software requirements · Software engineering · Artificial chemical reaction optimization algorithm

1 Introduction

In today's world, software plays an important role in the industry and the economy. Software can play the role of controlling, monitoring, forecasting and analyzing activities. Software production is a complex process. A defect in any of the software development steps can lead to a failure of the project (Dhanajayan and Pillai 2017). The stage of determining software requirements is a very critical stage. Defect in determining the correct software requirements may not be discovered until the final stages of software production (Chatterjee and Maji 2016). The later the hidden defects are discovered in this process, the greater the cost is imposed on the project. In the incremental software development method, the product develops in several releases. Software developers face a set of requirements in every release that they need to develop. It is almost impossible for all the proposed requirements to be developed in every release. Problems such as lack of funding, lack of time, and intrinsic contradiction between the requirements prevent the development of all the proposed requirements (Thakurta 2017). For developer teams, it is important to select an optimal subset of the proposed requirements which can have maximum satisfactions to clients and minimum cost and time for product development. In large projects with a great number of requirements, it is very difficult to select the optimal subset. For this reason, it is necessary to have a method for determining the optimal

Communicated by V. Loia.

✉ Ali Ebrahimnejad
a.ebrahimnejad@qaemiau.ac.ir

Hamidreza Alrezaamiri
hamidreza.alreza@baboliau.ac.ir

Homayun Motameni
motameni@iausari.ac.ir

[1] Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran

[2] Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran

[3] Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

Published online: 01 October 2018

subset of requirements in order to help the software engineers with easier decision making.

The problem of selecting an optimal subset of requirements that is known as the next release problem (NRP) was first introduced by Bagnall et al. (2001). In this problem, the goal is to select an optimal subset of requirements that can provide the highest satisfactions to clients at the lowest cost and satisfy the constraints of the problem. Restrictions on this problem may include interactions between requirements or the presence of a cost ceiling for the project.

The NRP problem is considered an NP-hard problem because of the two conflicting objectives of reducing the costs and increasing the satisfactions. Due to the complexity of this problem, the single-objective optimization methods cannot find an optimal subset of the requirements (Wessing and Preuss 2016). For this reason, the use of meta-heuristic multi-objective algorithms has been considered for solving this problem (Deb 2001). The meta-heuristic algorithms, due to their high capabilities, could obtain the solutions to the optimization problems at the suitable time due to their high capabilities (Cai et al. 2017; Shen et al. 2017; Guo et al. 2017; Wang et al. 2017).

The developer team should assign a value as the development cost and another value as the rate of satisfactions to each of the proposed requirements for development in every release. In previous papers, allocation of these values was done in crisp form. Considering the presence of clients with different views and desires, the allocation of a crisp value as the satisfaction for each requirement is not logical. Also, due to technical problems occurring at the time of implementing a requirement, the allocation of a crisp value as the cost of developing a requirement is also not reasonable. In this situation, the allocation of values of fuzzy type for the satisfaction and cost of each requirement is more rational and more practical. In this paper, we intend to solve this two-objective problem by assigning fuzzy values. To solve this problem, we use an artificial chemical reaction optimization algorithm (Lam and Li 2010). The artificial chemical reaction optimization algorithm presented in recent years is a powerful meta-heuristic algorithm for solving complex optimization problems. This algorithm searches the problem space well with the help of its various operators. In addition, it has been successful in solving many optimization problems (Nayak et al. 2015; Rao and Banka 2017).

The remainder of the paper is organized as follows. Section 2 reviews the related literature in this scope. Section 3 discusses the multi-objective NRP problem and formulates a multi-objective approach to this problem. Section 4 introduces the proposed method. Section 5 analyzes the experiments and the results. Finally, Sect. 6 concludes the paper and recommends future works in this domain.

2 Literature review

The search-based software engineering (SBSE) is one of the research areas in which search-based optimization algorithms are used to solve software engineering domain problems. The NRP problem belongs also to this area (Sadiq and Jain 2014). The classification, analysis, and evaluation of previous work have been done on problems related to the selection and prioritization of software requirements (Pitangueira et al. 2015). One of the first attempts for solving the NRP problem has been done in the work (Karlsson 1996) where the author used analytic hierarchy process (AHP) and quality function deployment (QFD) methods to select and prioritize software requirements. In the AHP method, requirements were classified and in the QFD method, requirements were prioritized. In projects with a high number of requirements, these two methods are not suitable for their long runtime.

In the work (Lai et al. 2008) a new method of ranking the clients requirements was introduced with respect to the competitive market. In this method, fuzzy QFD was used. In addition to clients' feedback, the authors also considered rival software companies products to rank their requirements. In the research (Sadiq and Jain 2014) a fuzzy method was used for the prioritization of the requirements in the process of eliciting goal-based requirements. This method is based on the combination of weighted relations and the fuzzy analytical hierarchy process (FAHP). The research (Alrashoud and Abhari 2017) considered three criteria when planning for the development of the next release, namely stakeholders' satisfaction, risk, and resource availability. The proposed method of this article used the fuzzy inference system to overcome the uncertainty of the problem and to solve it. The work (De Souza et al. 2011) proposed an optimal release policy for multi-release software system by taking into consideration the testing as well as the operational phase.

The work (Bagnall et al. 2001) introduced three different methods to solve the NRP problem. The first method was to use linear programming to obtain the exact solution to the problem. In the second method, three greedy algorithms were used. In the third method, two local search algorithms were employed. In NRP problems, with a low number of requirements, linear programming finds the exact solution to the problem at the right time. However, if the problem is big, the linear programming method cannot solve it at a reasonable time. The other two methods introduced in the article were not able to provide high-quality solutions because of their low search capabilities in the problem space.

The meta-heuristic optimization algorithms used to solve the NRP problem can be grouped into two categories, namely single-objective and multi-objective categories. In many multi-objective problems, the algorithm transforms the multi-objective problem into a single-objective problem by giving weight to each of the goals and combining them. The work (Araújo et al. 2017) proposed an issue based on genetic algorithm optimization and machine learning to solve the NRP problem. In the research (Jiang et al. 2010), to select the optimal set of requirements, the ACO algorithm is combined with a local search algorithm in order to increase the quality of the solutions to the problem.

Some of the researchers have recently solved the NRP problem based on multi-objective approach. In multi-objective problems, none of the objectives are superior to each other, and each objective has the same importance. The work (Zhang et al. 2007) formulated the NRP problem in a multi-objective manner for the first time. The research (Durillo et al. 2011) analyzed the quality of the solutions obtained by several multi-objective optimization algorithms to solve the NRP problem. The work (Del Sagrado et al. 2015) used the multi-objective ACS algorithm to solve this problem. Here, three types of interactions between requirements are defined and included in the datasets for the first time. The study (Chaves-González and Pérez-Toledano 2015) used the differential evolution algorithm to select the optimal subset of the proposed requirements. The results of this algorithm were of higher quality compared to some other optimization algorithms. In this study, three interactions between requirements are considered and the two dataset evaluated in this article are made available. The work (Chaves-González et al. 2015a) used the multi-objective artificial bee colony optimization algorithm (HMOABC) and the teaching–learning-based optimization algorithm (TLBO) (Chaves-González et al. 2015b) to solve this problem. In all three studies, the datasets are the same and are presented in crisp form. There are also three types of interactions between requirements as problem constraints in datasets. The study (Veerapen et al. 2015) used the integral linear programming method in two single-objective and multi-objective models. They were able to find the exact solution to the problem in a single-objective model and to the multi-objective small problems. However, their proposed method does not have the proper runtime on large-scale multi-objective problems.

In this paper, we intend to use an artificial chemical reaction optimization algorithm to solve the next release problem. With the help of its various operators, this algorithm can search the problem space well. In the software projects, the occurrence of unforeseen events always changes the cost and priority of the requirements. For this reason, the allocation of fuzzy numbers as cost and satisfaction to each requirement is more reasonable than the allocation of crisp numbers. To increase the reliability of the problem solutions, we first create the datasets using a fuzzy method. Also, to make the problem seem more realistic, in addition to the three interactions between requirements, we also define two other interactions and consider them in the implementation.

3 The problem of selecting an optimal subset of multi-objective requirements

In the real world, optimization problems have generally several objectives. Single-objective problems either have an objective intrinsically or create one main objective if there are several objectives in the problem by giving weight to each objective. For example, in a problem with two objectives f_1 and f_2 we intend to change the problem into a single-objective problem by giving weights α and β to these objectives. In Eq. (1), we see that the combination of these two objectives forms the main objective of the problem (F). In single-objective problems, there is usually a unique solution as the absolute answer to the problem.

$$\alpha * f_1 + \beta * f_2 = F \tag{1}$$

In multi-objective problems, objectives are not combined and there is no single solution to the problem. However, there are a set of solutions none of which has superiority over the others. This set of solutions is called the non-nominated solutions (NDS). This set shows the Pareto front in the multi-objective problems diagram (Schütze et al. 2016).

For example, there are k objectives in a problem; and $x = [x_1, \ldots, x_k]$ and $y = [y_1, \ldots, y_k]$ are two solutions to this problem. Solution x dominates solution y only if for all objectives $i = 1, 2, \ldots, k$, x is better or equal to y, and x is exactly better than y in at least one objective. Otherwise, none of the two solutions dominates the other. Figure 1 shows three solutions. Objectives f_1 and f_2 are two minimum objectives. Solution B dominates solution C because solution B is of higher quality than solution C for both objectives, $f_1(B) < f_1(C)$ and $f_2(B) < f_2(C)$. The two solutions A and B are non-dominated because each one dominates the other in one objective, $f_1(A) < f_1(B)$ and $f_2(B) < f_1(A)$.

3.1 Fuzzy concepts

In this subsection, we introduce the definitions and computational operators required in the paper (Ebrahimnejad et al. 2015, 2016).

Definition 1 A triangular fuzzy number such as $\tilde{A}$ is represented in the form of $\tilde{A} = (a_1, a_2, a_3)$ and with the membership function of Eq. (2). Figure 2 shows a triangular number with values (1, 2, 3).

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1}, & a_1 \le x \le a_2, \\ \frac{a_3-x}{a_3-a_2}, & a_2 \le x \le a_3, \end{cases} \tag{2}$$

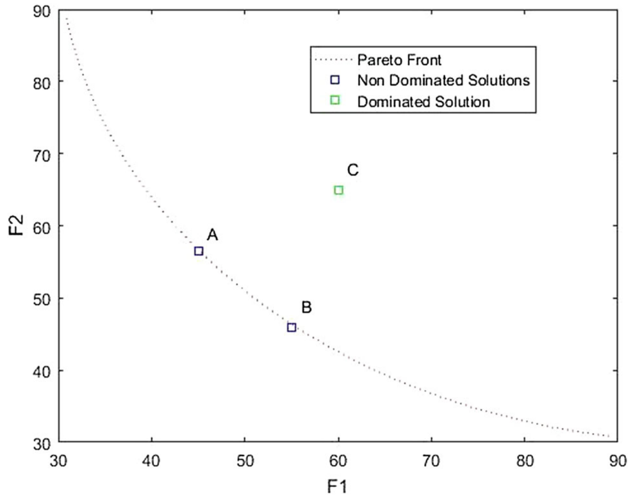


Fig. 1 Pareto front schema

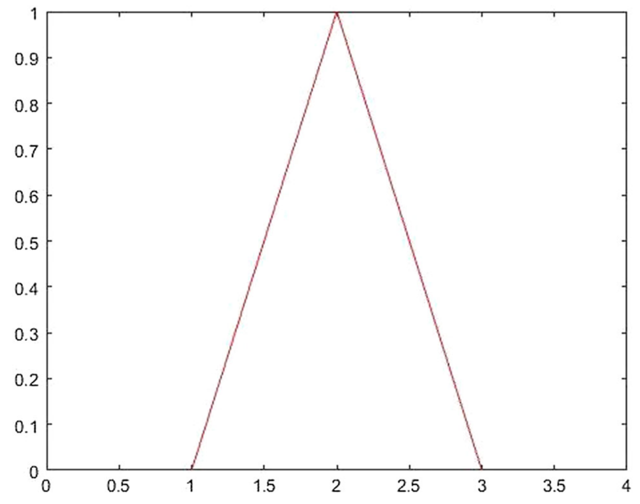


Fig. 2 A triangular number

Let $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$ be two nonnegative triangular fuzzy numbers, then addition and multiplication operations are defined as follows:

$$\tilde{A}\tilde{+}\tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3) \tag{3}$$

$$\tilde{A}\tilde{\times}\tilde{B} = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3). \tag{4}$$

Definition 2 The α-cut of the triangular fuzzy number $\tilde{A} = (a_1, a_2, a_3)$ is given by $\left[\tilde{A}\right]_\alpha = \left[\tilde{A}^L_\alpha, \tilde{A}^R_\alpha\right] = [(a_2 - a_1)\alpha + a_1,\ a_3 - (a_3 - a_2)\alpha]$.

The distance between two fuzzy numbers is determined by Eq. (5).

$$D_{p,q}(\tilde{A},\ \tilde{B}) = \begin{cases} \left[(1-q)\int_0^1 \left|A^-_\alpha - B^-_\alpha\right|^p d\alpha + q\int_0^1 \left|A^+_\alpha - B^+_\alpha\right|^p d\alpha\right], & p < \infty \\ (1-q) \sup\limits_{0<\alpha\le 1} \left|A^-_\alpha - B^-_\alpha\right| + q \inf\limits_{0<\alpha\le 1} \left|A^+_\alpha - B^+_\alpha\right|, & p = \infty \end{cases} \tag{5}$$

where the parameter p denotes the priority weight assigned to the end points of the support. If the expert has no preference, $D_{p,\frac{1}{2}}$ is used. The parameter q determines the analytical properties of $D_{p,q}$. For two fuzzy numbers $\tilde{A}$ and $\tilde{B}$, $D_{p,q}$ is proportional to:

$$D_{p,q}(\tilde{A},\ \tilde{B}) = \left[(1-q)\sum_{i=1}^{n} \left|A^-_{\alpha_i} - B^-_{\alpha_i}\right|^p + q\sum_{i=1}^{n} \left|A^+_{\alpha_i} - B^+_{\alpha_i}\right|^p\right]^{\frac{1}{p}} \tag{6}$$

If $q = \frac{1}{2}$ and $p = 2$, we obtain the following:

$$D_{2,\frac{1}{2}}(\tilde{A},\ \tilde{B}) = \sqrt{\left[\frac{1}{2}\sum_{i=1}^{n} \left|A^-_{\alpha_i} - B^-_{\alpha_i}\right|^2 + \frac{1}{2}\sum_{i=1}^{n} \left|A^+_{\alpha_i} - B^+_{\alpha_i}\right|^2\right]} \tag{7}$$

To compare two fuzzy number $\tilde{A}$ and $\tilde{B}$ using the α_i-cuts, we compare them to $\tilde{0} = (0, 0, 0)$. In fact, Eq. (7) is used to compute $D_{2,\frac{1}{2}}(\tilde{A},\ \tilde{0})$ and $D_{2,\frac{1}{2}}(\tilde{B},\ \tilde{0})$. We can conclude that $\tilde{A} \preceq \tilde{B}$ if $D_{2,\frac{1}{2}}(\tilde{A},\ \tilde{0}) \le D_{2,\frac{1}{2}}(\tilde{B},\ \tilde{0})$.

3.2 The problem of selecting the requirements

In software engineering projects, eliciting and properly selecting the requirements are very important. A lot of research has been done to better understand and elicit the requirements in the early stages of product development. Due to the presence of some problems, the proper selection of requirements has become a major challenge for software engineers. This is a more serious challenge in the incremental software development methods through which the product develops in various releases.

Problems such as conflicting objectives, large number of requirements for large-scale projects, different levels of clients importance for software companies, the presence of clients with different views, companies willingness to meet the needs of new clients, factors affecting the market and political factors make this choice very difficult. Despite these problems, the developer team should select an optimal subset of the requirements which can be of most consent to the clients with the most economical budget (Del Sagrado et al. 2015). Another factor which makes the choice difficult is the interaction between requirements. The interaction between requirements as the limitation of the problem is classified into four general categories (Del Sagrado et al. 2015).

1. Implication $r_i \Rightarrow r_j$. If the requirement r_i does not develop, the requirement r_j cannot develop.
2. Combination $r_i \oplus r_j$. The requirements r_i and r_j either develop together or do not develop.
3. Exclusion $r_i \otimes r_j$. The requirement r_i and r_j that cannot develop simultaneously.

4. Modification. The development of the requirement r_i implies that some other requirements change their satisfaction or implementation effort.

The first three interactions are applied as explicit interactions in the projects. However, the fourth interaction which is implicit is usually ignored. In recent researches (Del Sagrado et al. 2015; Chaves-González and Pérez-Toledano 2015; Chavez-Gonzalez et al. 2015a, b), only the first three interactions are considered in simulations. We intend to divide the fourth interaction into two distinct interactions and apply them for the first time in our simulations. We define the two interactions as follows:

(a) $r_i x\% \Theta r_j$. Impact on satisfaction: If the requirement r_i is developed and requirement r_j is developed, only x% of satisfaction r_j is applied.
(b) $r_i x\% \odot r_j$ Impact on satisfaction: If the requirement r_i develops and requirement r_j does not, x% of satisfaction r_j is applied.

Applying these two interactions together with the first three interactions makes the problem more realistic. Lack of changing the NRP problem into a single-objective problem has a great advantage for the developer team. In multi-objective problems, software engineers face a set of NDS solutions instead of a good solution in the Pareto front diagram. This diagram gives the developer team a great analytic power for decision making. For example, software engineers can conclude from the Pareto front diagram how much satisfactions to client are reduced if they cut down the cost of product by 10%. Or, for example, if they want to increase the satisfactions to clients by X percent, how much extra cost for product development should they consider (Zhang et al. 2007).

3.3 Fuzzy multi-objective NRP formulation

The fuzzy multi-objective NRP problem is the extended version of the original NRP problem. In this problem, n requirements are proposed for development in the next release. We consider this set as $R = \{r_1 \ldots r_n\}$. In the original NRP problem, a crisp number is considered for the degree of satisfaction and the cost of each requirement. Since the priority of the requirements is mostly changed during the execution of the projects, the allocation of fuzzy numbers instead of the crisp numbers is more logical (Turan 2017). Also, usually in projects, due to technical and non-technical problems in implementing the requirements, the development time of each requirement may exceed the amount predicted. This increase in time affects the cost of development of each requirement. Therefore, taking fuzzy numbers as the cost of each requirement by experts is more reasonable than taking crisp numbers. In this problem, we use triangular fuzzy numbers to show the satisfaction rate and cost of each requirement.

Example 1 Assume that seven clients attribute values 6, 9, 8, 1, 7, 4, 9 to the satisfaction rate of the requirement r_i. The lowest and the highest levels of satisfaction rate are 1 and 9, respectively. To convert these numbers to a triangular fuzzy number, the smallest value is taken as number a_1, the mean of numbers as number a_2, and the largest value as number a_3 and yields the fuzzy number $\tilde{A}_i = (1, 6.28, 9)$.

The satisfaction rate of each requirement is valued by tools such as questionnaires or survey by clients. These values are stored in a matrix a_{ij} where $i = 1, 2, 3$ and $j = 1, 2, \ldots, n$ is maintained. The cost of each requirement takes a fuzzy value by the experts and stored in a matrix e_{ij} where $i = 1, 2, 3$ and $j = 1, 2, \ldots, n$. For each developer team, clients have different levels of importance. For each number of clients, the level of client importance is calculated as in Example 1 and is maintained by the vector $\tilde{W} = (w_1, w_2, w_3)$.

The overall satisfaction level of each requirement is stored in the matrix S which is a $3 * n$ dimension matrix. The overall satisfaction level of the ith requirement is calculated by Eq. (8).

$$\tilde{s}_i = \tilde{W} \tilde{\times} \tilde{a}_i \tag{8}$$

In the FMONRP problem, each solution is encoded as a vector X where the ith cell is located in the vector corresponding to the ith requirement in the R set, where i value is $i = 1, 2, \ldots, n$. In Fig. 3, a solution is displayed. In each solution, each requirement that is selected for development in the next release accepts value 1 in its corresponding cell, and otherwise value 0. In the FMONRP problem, the objective is to find a solution that provides the highest level of benefits to clients with the least development cost. In addition to the two objectives mentioned, there are also two limitations in the problem that solutions must satisfy them. The first limitation is five interactions between requirements, as explained in the previous section. The second limitation is the cost threshold; that is, each solution should have a cost less than the threshold value. We define the cost threshold with the fuzzy number $\tilde{T}_c$. For example, $\tilde{T}_{70\%}$ is a fuzzy number that equals 70% of the total cost. Inequality (9) expresses this limitation.

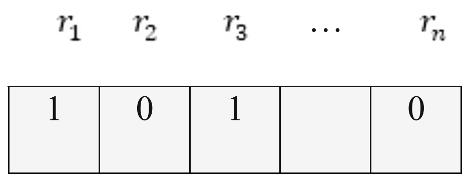


Fig. 3 Structure of a solution

To evaluate this inequality, first we use Formula (7) and then we compare the two sides of the inequality with the fuzzy number $\tilde{0} = (0, 0, 0)$. If the fuzzy number $X \tilde{\times} \tilde{e}_i$ obtained is smaller than the fuzzy number of the cost threshold, it is acceptable. Otherwise, the X solution is invalid.

$$X \tilde{\times} \tilde{e}_i \leq \tilde{T}_c \tag{9}$$

The FMONRP problem is expressed by Eq. (10) as follows:

$$\begin{aligned} &\text{Max } S(x) = \sum_{i=1}^{n} \tilde{s}_i \times x_i \\ &\text{Min } E(x) = \sum_{i=1}^{n} \tilde{e}_i \times x_i \\ &\text{Subject to cost threshold constraint \& \& interaction constraint} \end{aligned} \tag{10}$$

4 Fuzzy multi-objective artificial chemical reaction optimization for NRP

In this paper, we propose a meta-heuristic optimization algorithm based on the artificial chemical reaction optimization algorithm for problem solving. ACRO is a population-based swarm intelligence meta-heuristic inspired by natural chemical reaction (Lam and Li 2010; Rao and Banka 2017). A chemical reaction is a natural process of transforming the unstable chemical substances molecules to the stable ones. A chemical reaction starts with some unstable molecules with excessive energy. The molecules interact with each other. They are converted to those with minimum energy to support their existence. The energy associated with a molecule is called as entropy which can be considered as fitness function. Reactions may be bimolecular or monomolecular depending on the number of molecules taking part in the reaction. This property is embedded in ACRO to solve optimization problems. ACRO algorithm starts with set of initial molecules in a population. Then molecules are consumed and produced via chemical reactions. The algorithm terminates when the criteria of the problem are satisfied (Nayak et al. 2015). The flowchart of the ACRO algorithm is shown in Fig. 4.

In the first step, algorithm parameters such as the number of atoms per molecule, the total number of molecules (NumMole), the boundary of the search space and the algorithm termination conditions are determined. In the second step, a number of molecules equal to NumMole are randomly generated to create an initial population. After generating primary molecules, their fitness amounts are evaluated. At algorithm ACRO, the amount of the fitness of each molecule is called the level of enthalpy of that molecule. In the third step, chemical reactions occur between the molecules. In the ACRO algorithm, chemical reactions act as search operators such as crossover and mutation in the genetic algorithm. After the occurrence of chemical reactions, new molecules are created (Dam et al. 2017).

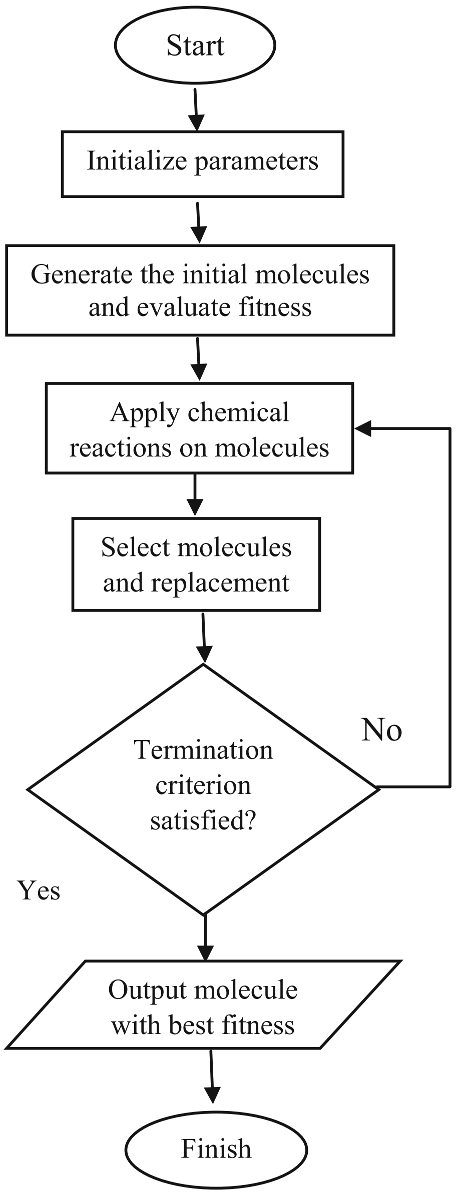

Start
Initialize parameters
Generate the initial molecules and evaluate fitness
Apply chemical reactions on molecules
Select molecules and replacement
Termination criterion satisfied?
No
Yes
Output molecule with best fitness
Finish

Fig. 4 Flowchart of the ACRO algorithm

In the following subsection, we introduce the various chemical reactions of the ACRO algorithm as binary encoded. In the fourth step, new molecules are evaluated in terms of enthalpy, and they are replaced by older molecules if they are better. These steps continue until the conditions for terminating the algorithm are met. The termination condition can be the number of specific iterations or the convergence of the algorithm solutions. Finally, the best algorithm solutions are used as output.

Molecule 1	0	0	1	1	0	1
Molecule 2	1	0	1	0	1	1
Mask bit	1	1	0	1	0	1
New molecule1	1	0	1	0	0	1
New molecule2	0	0	1	1	1	1

Fig. 5 A binary display of the displacement reaction

molecule1	0	0	1	0	0	1
molecule2	1	0	1	1	1	1
New molecule	0	0	1	1	0	1

Fig. 6 A synthesis reaction

4.1 Types of chemical reactions of the ACRO algorithm

Chemical reactions are divided into bimolecular and monomolecular groups depending on the number of participating molecules. Bimolecular reactions involve two molecules such as synthesis, Redox2 and displacement. Monomolecular reactions are those in which only one molecule is involved such as Redox1 and decomposition.

4.1.1 Displacement reaction

In the displacement reaction, two new molecules (two children) are produced by considering two old molecules (two parents). A random binary mask is created of the size of the molecules. If *i*th bit of the mask is 1, *i*th bit of new molecule 1 is copy of *i*th bit of molecule 2, and *i*th bit of new molecule 2 is copy of *i*th bit of molecule 1. Otherwise, if *i*th bit of the mask is 0, *i*th bit of new molecule 1 is copy of *i*th bit of molecule 1, and *i*th bit of new molecule 2 is copy of *i*th bit of molecule 2. Figure 5 shows a binary display of the displacement reaction.

4.1.2 Synthesis reaction

In this reaction, two molecules participate and create a new molecule (child). Non-matching bits of two molecules are determined. Then, one bit from the non-matching bit of the first molecule and one bit from the non-matching bit of the second molecule are consecutively selected to form a new molecule. Figure 6 shows a synthesis reaction.

4.1.3 Redox1 reaction

In this reaction, only one molecule is involved. One of the atoms of this molecule is randomly selected. In case its value is 1, it is changed to 0, and if the value is 0, it is changed to 1. This reaction acts as the mutation operator in the genetic algorithm. Figure 7 shows a binary display of a Redox1 reaction.

1	0	1	0	0	1	molecule
1	0	1	1	0	1	New molecule

Fig. 7 The binary display of a Redox1 reaction

1	0	1	0	0	1	1	molecule1
0	0	0	1	1	1	0	Molecule2
1	0	0	1	1	1	1	New molecule1
0	0	1	0	0	1	0	New molecule2

Fig. 8 The binary display of a Redox2 reaction

1	0	1	0	0	1	1	molecule
1	0	0	1	1	1	1	New molecule

Fig. 9 The binary display of a decomposition reaction

4.1.4 Redox2 reaction

In this reaction, two molecules are involved that produce two new molecules. This reaction acts as a two-point crossover operator in the genetic algorithm. Two indices of molecule are randomly generated and the bits of the molecules between the two indices are exchanged to produce two new molecules. Figure 8 shows the binary display of a Redox2 reaction.

4.1.5 Decomposition reaction

In this reaction, only one molecule is involved. Two atoms are randomly selected from the molecule, and the amount of all the atoms between them is flipped in order to create a new molecule. The amount of the rest of the atoms of the new molecule is the same as those of the old molecule. Figure 9 shows a binary display of a decomposition reaction.

4.2 Fuzzy multi-objective artificial chemical reaction optimization

In the proposed method, the non-dominated sorting technique (Deb et al. 2002) and the non-dominated solution archive technique (Knowles and Corne 1999) are used. By use of the non-dominated sorting technique, we arrange the solutions obtained based on their dominance ranking and their crowding distance. In multi-objective problems, a solution is better than another solution only if it has a lower dominance ranking or, in the case of equality, a greater crowding distance. Using the non-dominated solution archive technique, we store the best solutions found during each iteration in the archive. In the algorithm 1, the FMOACRO pseudo-code is shown to solve the problem of NRP.

Algorithm 1. The FMOACRO pseudo-code for solving the problem of FNRP.

```
Input: NumMoles        /* number molecules population */
Output: NDS-archive
1: R=generate initial molecules.        /* R is the molecules population */
2: R= check(R).    /* The molecules in the R set are sent to the Check function to verify their validity and evaluate their enthalpy. */
3: NDS-archive = empty
4: While (not stop condition satisfied)
5:   newR=empty.    /* newR is new molecules population that creating in Apply chemical reactions */
6:   for i=1: NumMoles
7:     from R random chose two molecules.    /* for example $r_1$, $r_2$ */
8:     r = rand.       /* create a random number from (0,1) and save in r variable */
9:     /* Apply chemical reactions on molecules */
10:        if (r < 0.2)      /* do Synthesis reaction */
11:        newReact = Synthesis reaction ($r_1$, $r_2$)    /* in Synthesis reaction create a new molecule */
12:        newR = newR ∪ check(newReact).
13:        else if (r<0.4)    /* do displacement reaction */
14:        [$newReact_1$, $newReact_2$] = displacement reaction ($r_1$, $r_2$)    /* in displacement reaction create two new molecules */
15:        newR = newR ∪ check(new $molecule_1$) ∪ check(new $molecules_2$)
16:        else if (r<0.6)    /* do redox1 reaction */
17:        newReact = redox1 reaction ($r_1$)    /* in redox1 reaction create a new molecule */
18:        newR = newR ∪ check(newReact).
19:        else if (r<0.8)    /* do redox2 reaction */
20:        [$newReact_1$, $newReact_2$] = redox2 reaction ($r_1$, $r_2$)    /* in redox2 reaction create two new molecules */
21:        newR = newR ∪ check($newReact_1$) ∪ check($newReact_2$)
22:        else    /* do decomposition reaction */
23:        newReact = decomposition reaction ($r_2$)    /* in decomposition reaction create a new molecule */
24:        newR = newR ∪ check(newReact).
25:        end for
26:   R= R ∪ newR   /* union population newR with population R */
27:   R= CalculateRC(R).  /* function CalculateRC calculate rank and crowding distance each molecule in population R */
28:   NDS-archive = update NDS-archive(R).
29:   R= save best NumMoles molecules and remove other molecules.    /* update and select molecules */
30:   remove population newR
31:  end while
```

The input to the algorithm is the NumMoles parameter. This parameter specifies the number of molecules in the population. In the first line, the algorithm initially generates the NumMoles molecule randomly and holds it in the *R* population. In this case, each molecule is an *n*-dimensional vector where n is the number of the proposed requirements. Each molecule is in fact a solution to the problem. How to generate each molecule is described in Sect. 3.3. In the second line, molecules of population *R* are sent to the check function to verify their validity and evaluate their enthalpy. As discussed in Sect. 3.2, the problem has two kinds of limitations. If the molecules violate the limitations of the problem, they are modified within the check function, and then their cost and satisfaction rates are calculated. The presence of the check function ensures that all molecules of the population *R* are valid. In the third line, the NDS-archive set is created with a null value. In each iteration, the discovered NDSs are stored in the NDS-archive. In the fourth line, iterations of the algorithm start until the termination condition. In the fifth line, in each iteration, a new *R* population is defined with empty value. In the new *R* population, new molecules produced from chemical reactions are stored in the current iteration. In the sixth line, a for loop is considered. Because of this loop, a number of chemical reactions equal to NumMoles occur in each iteration. In the seventh line, for each chemical reaction, two molecules are randomly selected from the *R* population. A random and independent selection of two molecules from the population for participating in the reaction may cause some molecules not to be selected in all iterations or enter the reactions more than once. The results (Rada-Vilela et al. 2011) show that this style of selection can lead to faster convergence of solutions. In contrast to this method of selecting molecules, there are many other methods. Two examples of these methods are: 1. Sequential selection of each molecule for participation in chemical reactions. 2. Use of the roulette wheel to consider a greater chance of selecting the most desirable molecules.

In each reaction, one or two new molecules may be generated which will be added to the new *R* population after being sent to the check function. Depending on the random number r produced in the eighth line, one of the five chemical reactions occurs. For example, if the value of *r* is <0.2, the synthesis reaction occurs from line 10 to 12. In the synthesis reaction, two molecules participate and produce a new molecule. Depending on the number of *r*, each of the chemical reactions of lines 10–24 can occur. With the completion of chemical reactions, in line 26, the population of new molecules (new R) is merged with that of the old molecules (*R*). To determine NDSs, the population *R* is sent to the calculate RC function.

In this function, the population *R* solutions are sorted according to non-dominated ranks and crowding distance. One solution is better than the other solution only if it has a lower dominance ranking or, in case of equality, it has more crowding distance. In line 28, the NDSs discovered from population *R* are stored in the NDS-archive. In line 29, to maintain the population size, the NumMole of the more promising molecule of *R* population is kept the rest of the molecules are eliminated.

One of the differences between the FMOACRO algorithm and most of the other evolutionary algorithms such as GA, PSO, and ABC is the number of molecules produced from chemical reactions; that is, for example, in the synthesis chemical reaction, two molecules are involved which produce a new molecule. However, in the displacement chemical reaction, two molecules are involved which produce two new molecules. Yet, in most evolutionary algorithms, the number of operator outputs is exactly specified.

One of the advantages of the FMOACRO algorithm over other meta-heuristic algorithms is having many operators. For example, the PSO algorithm only uses the update operator and the GA algorithm uses the crossover and mutation operators. However, the FMOACRO algorithm uses five different powerful operators. The presence of these diverse operators leads to a complete explore of the search space and finding the optimal solution in a shorter time.

5 Experiments

In this section, we will conduct experiments on two randomized synthetic datasets and present the related results. Since there are no fuzzy datasets in the previous research, we cannot compare the results of our proposed method with them. The experiments were performed with the R2016b version of the MATLAB software on a system with an Intel core i7, a 1.60 GHz processor, a 4 GB RAM, and a 64-bit Windows 10 operating system. Since we have random algorithms in these experiments, each algorithm is run 20 times independently. Subsequently, the mean and the standard deviation of the results of each algorithm are presented. On both datasets, three development cost limits (of 40%, 70%, 100% of the total development cost) are considered.

5.1 Datasets and experiment criteria

The first dataset includes 24 requirements and 12 interactions between requirements. In Table 1, the priority and the cost of development of each requirement are presented as fuzzy numbers. The priority level of each requirement for development in the next release can be asked from clients by means of questionnaires. The questionnaire can be given to clients in the form of crisp numbers and converted by the development team into fuzzy numbers. These scores show the level of clients satisfaction of each requirement for development in the next release. The lowest score given to each require-

Table 1 First dataset

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
a_{1i}	1	2	2	1	2	2	1	3	2	1	2	1
a_{2i}	4	3	4	2	3	3	3	4	4	2	3	2
a_{3i}	5	4	5	4	5	4	5	5	5	3	4	4
c_{1i}	2	1	2	1	3	2	1	2	1	1	2	1
c_{2i}	3	3	3	2	4	3	2	3	3	3	3	3
c_{3i}	5	4	4	3	5	5	3	4	4	4	5	5

	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}	r_{24}
a_{1i}	1	2	1	1	1	1	2	3	1	1	2	2
a_{2i}	2	3	2	2	3	3	3	4	4	4	4	3
a_{3i}	3	5	3	5	5	4	5	5	5	5	5	4
c_{1i}	2	1	2	1	3	1	3	1	1	2	2	2
c_{2i}	3	2	3	2	4	2	4	3	2	3	4	3
c_{3i}	4	3	5	4	5	4	5	5	4	4	5	4

$r_4 \Rightarrow r_{10}$ $r_{11} \Rightarrow r_{17}$ $r_{14} \Rightarrow r_{17}$ $r_{12} \oplus r_{15}$ $r_{22} \oplus r_{19}$ $r_2 \oplus r_9$ $r_5 \oplus r_{24}$
$r_3 60\% \Theta r_{10}$ $r_4 70\% \Theta r_7$ $r_1 50\% \Theta r_{15}$ $r_{11} 40\% \odot r_6$ $r_{21} 25\% \odot r_{22}$

ment is set to 1. In fact, the client has the least interest in developing this requirement in the next release while score 5 shows that the client is most interested in developing this requirement in the next release. Due to the change in the requirements priority that usually occurs when projects are running, considering fuzzy numbers instead of crisp numbers is much more suitable. The developer team can convert the priority scores of any number of clients to a fuzzy number using the method described in Sect. 3.3. The cost of developing any requirement may also change due to technical and unforeseen problems. Using fuzzy numbers instead of crisp numbers is also more reasonable for the cost. The cost of any requirement like r_j is calculated by the system's experts and is considered as (c_{1j}, c_{2j}, c_{3j}).

In this dataset, 12 interactions are considered as a constraint between requirements. We describe two examples of these interactions. The exclusion interaction between the two requirements r_2 and r_9 states that these two requirements cannot be developed together in the next release. If both requirements are simultaneously selected, the requirement r_2 must be deleted. The impact on cost interaction between the two requirements r_4 and r_7 states that if the requirement r_4 develops, and also if the requirement r_7 develops, only 70% of the cost of requirement r_7 is applied.

The second dataset includes 72 requirements and 20 interactions between requirements. In Table 2, fuzzy numbers show the satisfaction rate and cost of each requirement. This dataset is much more complex than the previous dataset. In addition, the range of scores for each requirement is from 1 to 10. For each software company, clients have different importance levels. The clients importance levels can be considered as crisp or fuzzy numbers. In this experiment, we consider the importance level of clients in every dataset as a triangular fuzzy number. Tables 3 and 4 show other parameters related to datasets 1 and 2, respectively.

In each test, we evaluate the quality of the solutions of the proposed method with three indicators of checking the quality of multi-objective problems. The first indicator to be evaluated is the number of non-dominated solutions (NDS) found by the algorithm. The more NDSs found on the Pareto front diagram, the better.

The second quality indicator was the spread achieved by the set of NDS (Δ-spread). This indicator calculates the diversity of the solutions by using the Euclidean distances between consecutive solutions in the Pareto front. Pareto fronts with a smaller spread are preferred. Δ-Spread is defined by Eq. (11).

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (n-1)d} \tag{11}$$

where d_i is the Euclidean distance between two consecutive solutions, $\bar{d}$ is the mean distance between each pair of solutions, N is the 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 third indicator of quality is hypervolume (HV). This indicator calculates the volume covered by the members of the NDS-archive using Eq. (12).

$$\text{HV} = \text{volume}\ (\cup_{i=1}^{|\text{NDS - archive}|} v_i) \tag{12}$$

Table 2 Second dataset

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

$r_{34} \Rightarrow r_{20}$ $r_{21} \Rightarrow r_{47}$ $r_{34} \Rightarrow r_{37}$ $r_{40} \Rightarrow r_{57}$
$r_{12} \oplus r_{25}$ $r_{32} \oplus r_{29}$ $r_{43} \oplus r_{39}$ $r_{52} \oplus r_{69}$
$r_{11} \otimes r_{30}$ $r_{23} \otimes r_{24}$ $r_{55} \otimes r_{51}$ $r_{56} \otimes r_{72}$
$r_{33} 50\% \Theta r_{38}$ $r_{61} 60\% \Theta r_{37}$ $r_{20} 55\% \Theta r_{51}$ $r_{72} 65\% \Theta r_{64}$
$r_{17} 30\% \odot r_{38}$ $r_{30} 40\% \odot r_{27}$ $r_{45} 15\% \odot r_{66}$ $r_{42} 20\% \odot r_{64}$

Table 3 Other parameters of the first dataset

Parameters	Values
$R_{\min}$ (cost, satisfaction)	(0, 0)
$R_{\max 40\%}$ (cost, satisfaction)	(80, 334)
$R_{\max 70\%}$ (cost, satisfaction)	(140, 536)
$R_{\max 100\%}$ (cost, satisfaction)	(201, 699)
W_1 = clients weight	(1, 2.8, 4)
n α-cut	10
Number molecules	40
Iterations	200

Table 4 Other parameters of the second dataset

Parameters	Values
$R_{\min}$ (cost, satisfaction)	(0, 0)
$R_{\max 40\%}$ (cost, satisfaction)	(428, 2218)
$R_{\max 70\%}$ (cost, satisfaction)	(750, 3318)
$R_{\max 100\%}$ (cost, satisfaction)	(1070, 4192)
W_2 = clients weight	(1, 2.6, 5)
n α-cut	10
Number molecules	150
Iterations	200

HV measures the diversity and convergence of the Pareto fronts obtained. One Pareto front has a larger HV than the other if solutions in the better front are more widely distributed than in the other or some solutions in the better front dominate solutions in the other. Algorithms with higher amounts of HV are better. In order to calculate this indicator, two reference points were required. Since the problem we were addressing has two objectives, these points were $r_{\min}$ $(\text{obj}1_{\min}, \text{obj}2_{\min})$ and $r_{\max}$ $(\text{obj}1_{\max}, \text{obj}2_{\max})$. These points contain the maximum and minimum amounts for the two objectives. For maximum the hypervolume, both objective function amounts had to be normalized. The normalization points used for each dataset are presented in Tables 3 and 4.

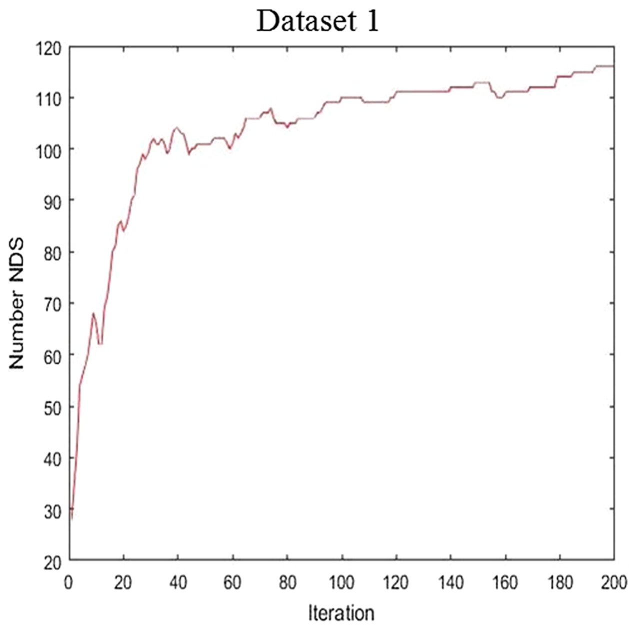
5.2 Experiments and analysis of the results

One of the most important evaluation criteria in multi-objective problems is non-dominated solutions. The more NDSs are discovered by an algorithm in the Pareto front diagram, the better the algorithm worked. In multi-objective problems, the NDS criterion also affects other criteria. Figure 10 shows the graph of the number of detected NDSs per iteration for both datasets.

The fact that the second dataset is larger than the first dataset has made the NDS of the second dataset much larger. In both of these figures, it is seen that in some of the iterations, there has been a slight drop in the graph and a decrease in the number of NDSs. The reason for this is that discovering a new NDS in the current iteration has led to the defeat of a number of discovered NDSs in previous iteration. However, as the trend of the algorithm continued, the number of NDSs is continuously added. In both figures, it can be seen that graphs are converging. It emphasizes the fact that the proposed method has detected almost all NDSs.

Table 5 shows the average and standard deviation of the results from 20 independent implementation of the proposed method on both datasets. The number of molecules of the proposed algorithm in the first and the second dataset tests was 40 and 150, respectively. Also, the number of iterations in all tests equaled 200. Since the proposed algorithm is accidental, considering the standard deviation criterion besides the average indicates the reliability of the results much better. As discussed in the previous sections, one of the limitations of the problem was to consider the maximum cost ceiling. This limitation usually applies when companies plan to save costs. During the experiment, each dataset was examined with three different ceilings.

Another criterion for evaluating the solutions in Table 5 is the Δ-spread criterion. The results of this criterion show the distribution of NDSs in the Pareto front diagram. The lower the Δ-spread criterion, the more helpful the solutions obtained by the algorithm for the developer team in making decision. The results of the proposed method in the Δ-Spread criterion show a fairly uniform distribution of NDS in the diagram. Figure 10 graphs show the distance between each two consecutive NDSs. The results of the HV criterion were calculated in both datasets and all three cost ceilings through normalization. In Tables 3 and 4, there are two reference points for each dataset and the cost ceiling is specified. The runtime of all tests is shown in Table 5. The second dataset had more runtime due to more complexity and more calculations. In both datasets, tests with a 40% cost limit have the highest runtime. The second-long runtime was due to tests with a 70% cost limit. The reason for this is the modification of the molecules by the algorithm. Molecules that violate the problem limit are modified in the check function. Because molecules in tests with 40% limit had the highest need for repair, they also have the highest runtime. In experiments with no cost limits, there is no need for modifying the molecules. Therefore, all molecules are valid in this regard. This has led to their shorter runtime.



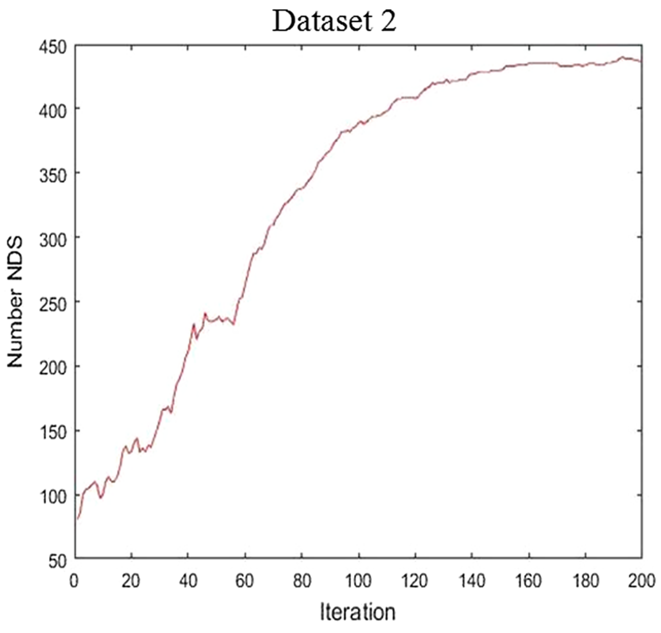


Fig. 10 The graph of the number of detected NDSs per iteration

Table 5 Mean and standard deviation of the solutions of the proposed method based on the criteria of the problem

	NDS	HV	Δ-Spread	Time (s)
Dataset 1				
Cost boundary 40%	48, 2, 87	52.9%, 3.3e−3	0.58, 4.2e−2	2.54, 5.8e−2
Cost boundary 70%	80.2, 4.94	52.3%, 7.7e−3	0.56, 4e−2	2.48, 5e−2
Without cost boundary	115.6, 5.63	55.2%, 2.6e−3	0.55, 2.9e−2	2.35, 4.4e−2
Dataset 2				
Cost boundary 40%	359.8, 30.95	55.2%, 3.3e−2	0.81, 2.8e−2	26.49, 2.70
Cost boundary 70%	380.1, 30.01	56.3%, 7.1e−3	0.79, 3.1e−2	23.7, 1.9e−2
Without cost boundary	454.2, 21.56	56.7%, 5.4e−3	0.78, 5.4e−2	18.55, 3.57

Figure 11 shows the Pareto front diagrams of each test as well as the HV diagram of both datasets without any cost limits. In the HV diagram, all solutions located in the green zone are defeated by at least one of the NDS solutions.

6 Conclusion

In this paper, the FMONRP problem was investigated. The original version of the NRP problem has been one-objective. However, since this problem faces two opposing objectives simultaneously, a multi-objective version of it has also been introduced in previous research. In this paper, we introduced the fuzzy multi-objective version of this problem for the first time. Using fuzzy numbers instead of crisp data is more logical and more practical. Due to the complexity and limitations in this problem, we used an artificial chemical reaction optimization algorithm. We defined two new interactions between requirements. During the implementation, five interactions were used between the requirements as one of the limitations of the problem. The results of the proposed algorithm were evaluated with three criteria of the multi-objective problems. The results and graphs of the proposed algorithm are very reliable and can help the developer team with the decision-making process.

The following works can be done as the new researches in this area:

- The use of other evolutionary algorithms to solve a problem or combine several evolutionary algorithms together.
- Introducing new methods for converting crisp values of questionnaires to fuzzy numbers.
- Allocation of fuzzy numbers of other types to the satisfaction rate and cost of each requirement.

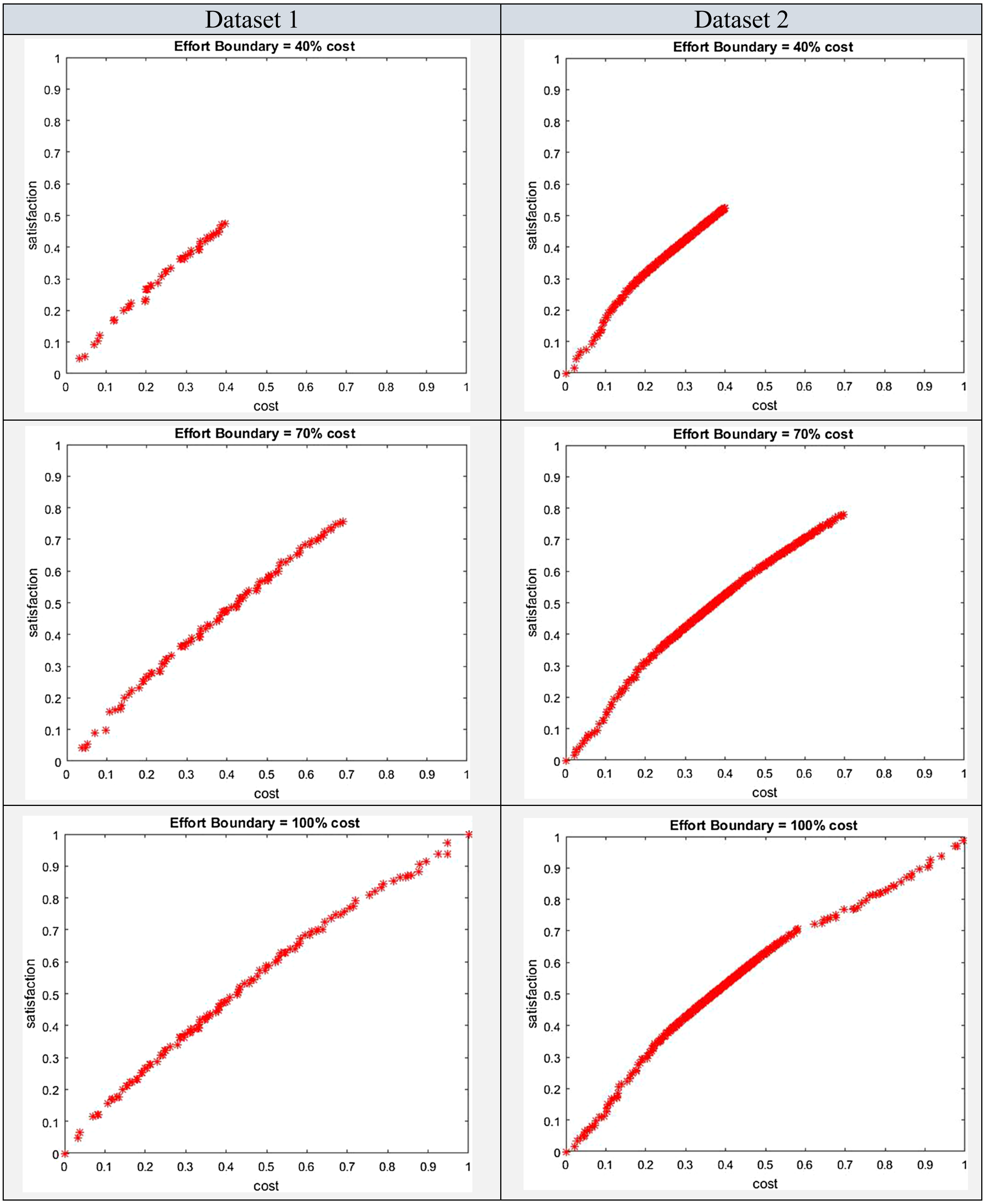


Fig. 11 Pareto front graphs and HV

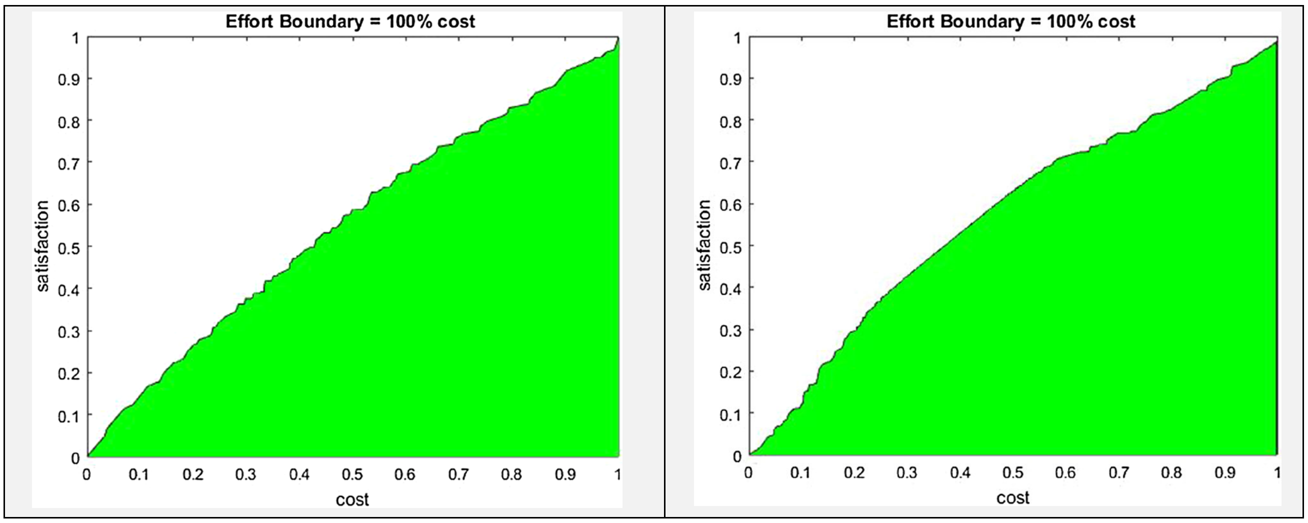


Fig. 11 continued

Acknowledgements The author would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Alrashoud M, Abhari A (2017) Planning for the next software release using adaptive network-based fuzzy inference system. Intell Decis Technol 11:153–165

Araújo AA, Paixao M, Yeltsin I, Dantas A, Souza J (2017) An architecture based on interactive optimization and machine learning applied to the next release problem. Autom Softw Eng 24:623–671

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

Cai X, Cheng X, Fan Z, Goodman E, Wang L (2017) An adaptive memetic framework for multi-objective combinatorial optimization problems: studies on software next release and travelling salesman problems. Soft Comput 21:2215–2236

Chatterjee S, Maji B (2016) A new fuzzy rule based algorithm for estimating software faults in early phase of development. Soft Comput 20:4023–4035

Chaves-González JM, Pérez-Toledano MA (2015) Differential evolution with Pareto tournament for the multi-objective next release problem. Appl Math Comput 252:1–13

Chaves-González JM, Perez-Toledano MA, Navasa A (2015a) Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowl-Based Syst 83:105–115

Chaves-González JM, Pérez-Toledano MA, Navasa A (2015b) Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection. Eng Appl Artif Intell 43:89–101

Dam TL, Li K, Fournier-Viger P (2017) Chemical reaction optimization with unified tabu search for the vehicle routing problem. Soft Comput 21:6421–6433

De Souza JT, Maia CL, do Nascimento Ferreira T, Do Carmo RA, Brasil MM (2011) An ant colony optimization approach to the software release planning with dependent requirements. In: International symposium on search based software engineering, pp 142–157

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

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

Del Sagrado J, Del Aguila IM, Orellana FJ (2015) Multi-objective ant colony optimization for requirements selection. Empir Softw Eng 20:577–610

Dhanajayan RC, Pillai SA (2017) SLMBC: spiral life cycle model-based Bayesian classification technique for efficient software fault prediction and classification. Soft Comput 21:403–415

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

Ebrahimnejad A, Karimnejad Z, Alrezaamiri H (2015) Particle swarm optimisation algorithm for solving shortest path problems with mixed fuzzy arc weights. Int J Appl Decis Sci 8:203–222

Ebrahimnejad A, Tavana M, Alrezaamiri H (2016) A novel artificial bee colony algorithm for shortest path problems with fuzzy arc weights. Measurement 93:48–56

Guo W, Chen M, Wang L, Wu Q (2017) Hyper multi-objective evolutionary algorithm for multi-objective optimization problems. Soft Comput 21:5883–5891

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

Karlsson J (1996) Software requirements prioritizing in requirements engineering. Proc Second Int Conf 1996:110–116

Knowles J, Corne D (1999) The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99, pp 98–105

Lai X, Xie M, Tan KC, Yang B (2008) Ranking of customer requirements in a competitive environment. Comput Ind Eng 54:202–214

Lam AY, Li VO (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14:381–399

Nayak SC, Misra BB, Behera HS (2015) Artificial chemical reaction optimization of neural networks for efficient prediction of stock market indices. Ain Shams Eng J 8:371–390

Pitangueira AM, Maciel RS, Barros M (2015) Software requirements selection and prioritization using SBSE approaches: a systematic review and mapping of the literature. J Syst Softw 103:267–280

Rada-Vilela J, Zhang M, Seah W (2011) Random asynchronous PSO. In: 2011 5th international conference on automation, robotics and applications (ICARA), pp 220–225

Rao PS, Banka H (2017) Novel chemical reaction optimization based unequal clustering and routing algorithms for wireless sensor networks. Wirel Netw 23:759–778

Sadiq M, Jain SK (2014) Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process. Int J Syst Assur Eng Manag 5:711–723

Schütze O, Martín A, Lara A, Alvarado S, Salinas E, Coello CA (2016) The directed search method for multi-objective memetic algorithms. Comput Optim Appl 63:305–332

Shen XN, Han Y, Fu JZ (2017) Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. Soft Comput 21:6531–6554

Thakurta R (2017) Understanding requirement prioritization artifacts: a systematic mapping study. Requir Eng 22:491–526

Turan HH (2017) Stochastic fuzzy multi-objective backbone selection and capacity allocation problem under tax-band pricing policy with different fuzzy operators. Soft Comput 21:4085–4110

Veerapen N, Ochoa G, Harman M, Burke EK (2015) An integer linear programming approach to the single and bi-objective next release problem. Inf Softw Technol 65:1–13

Wang K, Li X, Jia C, Yang S, Li M, Li Y (2017) Multiobjective optimization of the production process for ground granulated blast furnace slags. Soft Comput 4:1

Wessing S, Preuss M (2016) On multiobjective selection for multimodal optimization. Comput Optim Appl 63:875–902

Zhang Y, Harman M, Mansouri SA (2007) The multi-objective next release problem. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, pp 1129–1137