Journal of Intelligent & Fuzzy Systems 40 (2021) 5027–5041
DOI:10.3233/JIFS-201759
IOS Press

A novel approach for the next software release using a binary artificial algae algorithm

Poria Pirozmand[a], Ali Ebrahimnejad[b,*], Hamidreza Alrezaamiri[c] and Homayun Motameni[d]
[a]*School of Computer and Software, Dalian Neusoft University of Information, Dalian, China*
[b]*Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran*
[c]*Young Researchers and Elite Club, Babol Branch, Islamic Azad University, Babol, Iran*
[d]*Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran*

Abstract. In software incremental development methodology, the product develops in several releases. In each release, one set of the requirements is suggested for development. The development team must select a subset of the proposed requirements for development in the next release such that by consideration the limitation of the problem provides the highest satisfaction to the customers and the lowest cost to the company. This problem is known as the next release problem. In complex projects where the number of requirements is high, development teams cannot choose an optimized subset of the requirements by traditional methods, so an intelligent algorithm is required to help in the decision-making process. The main contributions of this study are fivefold: (1) The customer satisfaction and the cost of every requirement are determined by use of fuzzy numbers because of the possible changing of the customers' priorities during the product development period; (2) An improved approximate approach is suggested for summing fuzzy numbers of different kinds, (3) A new metaheuristic algorithm namely the Binary Artificial Algae Algorithm is used for choosing an optimized subset of requirements, (4) Experiments performed on two fuzzy datasets confirm that the resulted subsets from the suggested algorithm are free of human mistake and can be a great guidance to development teams in making decisions.

Keywords: Next release problem, software requirements, fuzzy numbers, binary artificial algae algorithm

1. Introduction

In software engineering, one of the first steps towards software development is the stage of elicitation and determination of requirements. These requirements are usually elicited from clients, product development teams, and market conditions [1]. After elicitation, a list of requirements for product development is specified. In software engineering, there are many methodologies for software development. One of the most popular methodologies is the incremental development method, which is very suitable for large projects. In the incremental development method, software is developed in several releases. The software company is confronted with a set of requirements in every release that need to be developed. Due to problems such as project budget constraints, proximity of project delivery time, technical problems, and inherent conflicts between requirements, it is impossible to develop all the proposed requirements [2].

In large projects, the choice of an optimal subset of the requirements is traditionally very difficult

*Corresponding author. Ali Ebrahimnejad, Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran. E-mail: aemarzoun@gmail.com; a.ebrahimnejad@qaemiau.ac.ir

and prone to error. For this reason, in order to help the development team with easier decision making, it is necessary to have a method for determining an optimal subset of the requirements. The problem of selecting an optimal subset of requirements is called the Next Release Problem (NRP) [3]. In this problem, the goal is to select an optimal subset of requirements that can satisfy the constraints of the problem so that it could create, with lowest cost, the highest degree of satisfaction for users. Different constraints could be considered for this problem. Some of these constraints are inherent while others are non-essential. One of the inherent constraints of this problem is the interaction between the requirements themselves. For example, two of the requirements for development may complement each other and two other requirements may conflict with one another. Sagrado et al. [4] categorized the interaction between requirements, as one of the constraints of the problem, in four general categories.

Implication $r_i \Rightarrow r_j$. If the requirement r_i is not developed, the requirement r_j cannot be developed either.
Combination $r_i \oplus r_j$. A requirement r_i cannot be chosen separately from a requirement r_j.
Exclusion $r_i \otimes r_j$. A requirement r_i cannot be chosen together with a requirement r_j.
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 in projects as explicit interactions. However, the fourth interaction, being implicit in nature, is usually ignored. Non-essential restrictions such as the product delivery time or the maximum allowable cost of product development are usually defined by manufacturing companies. The existence of these constraints and the two conflicting goals (increasing user satisfaction and reducing the cost of product development) place this problem in the NP-hard category. In spite of the complexity of the problem, metaheuristic algorithms can provide a quasi-optimal solution to this problem in a short time.

In each release, the development team should assign a value to satisfaction and another to implementation cost for each of the requirements for development [5, 6]. The level of satisfaction with any proposed requirement for development in the next release is asked from the users through a questionnaire and a survey. Because of different tastes and different scoring for each user requirement, assigning a crisp number as the level of satisfaction with each requirement is not sensible, while the allocation of a normal or triangular fuzzy number can give a much better view of the spectrum. For example, suppose four customers participate in a questionnaire and each one chooses a number in the range of [1, 10] as the requirement satisfaction score. If the amount of requirement satisfaction is given by a crisp number, the average of the selected numbers is computed. If customers choose 4, 3, 4, 5 or 3, 3, 9, 1, the average satisfaction of both groups is 4. This method is not acceptable for situations that the customers are uncertain about assigning an accurate satisfaction score. In fact, in such situations the satisfaction scores are expressed by fuzzy statements "very close to 4", "approximately 3", "near to five" and so on. Hence, they can be conceptualized by certain "appropriate" fuzzy numbers. Therefore, the development team determines a suitable fuzzy number according to the users' scores as the satisfaction level of each requirement [7]. Moreover, sometime the satisfaction levels are assessed by customers using fuzzy linguistic terms such as Excellent, Very Good, Good, Average, Poor and Very Poor and then the assessment results by customers are weighted and averaged. Therefore, there is a need to study the behavior of the set for these variables also. Furthermore, in our real-life situations, various social and natural phenomena belong to the normal distribution which has the continuous higher derivative of its membership function. Hence, in the presented study, the satisfaction levels of customers are represented in terms of trapezoidal fuzzy numbers and normal fuzzy numbers. Also, given the time and expertise necessary to implement each requirement as well as unforeseen problems during development, they also define a suitable fuzzy number as developing cost of each requirement.

In previous papers, the allocation of these values was done in the form of crisp, which, for the reasons given, is not a logical and practical approach [8, 9]. The existence of different conditions makes the development team consider different types of fuzzy numbers for the cost and satisfaction level of the requirements. Since the cost and satisfaction of a subset of the proposed requirements are obtained from the addition of every single one of them, the sum of fuzzy numbers of different types together creates another challenge.

In this paper, we use the least squares model proposed by Hassanzadeh et al. [10] to sum different types of fuzzy numbers together. We also use a Binary Artificial Alga Algorithm [11] (BAAA) in a bound

fashion to find the optimal subset of the requirements. The Artificial Alga Algorithm is a powerful metaheuristic algorithm for solving continuous optimization problems [12]. Korkmaz and Kiran [11] proposed the binary version of this algorithm which can solve discrete optimization problems such as the NRP problem very well.

The challenges that currently exist in solving the next release problem are summarized as below:

- Introducing an intelligent algorithm to select an optimal subset of the proposed requirements. This algorithm should achieve the highest level of customer satisfaction and the lowest cost for the company while taking into account all the limitations of the problem. To resolve this issue we use BAAA to find the optimal subset of the requirements. This algorithm is a new and powerful metaheuristic algorithm that could solve many complex engineering problems. In NRP, if the set chosen by the algorithm is not the best, it will cause dissatisfaction to the customers and a decrease in the company's credit.
- Because the problem under consideration is an NP-hard problem, the execution time of the algorithm can be one of the challenges for solving the problem. Therefore, the algorithm should provide acceptable solutions in a short time. To address this issue, we use BAAA that is an agile algorithm with acceptable run time and use the improved least squares model that extremely reduces computation time.
- Most often, customer priority changes during product development. Assigning fuzzy numbers instead of crisp numbers for satisfaction of requirements will alleviate this problem. On the other hand, the use of different kinds of fuzzy numbers leads to a better mapping of customers' views. In this situation we face the challenge of summing up different kinds of fuzzy numbers. To address this issue, we use an approximate approach for summing up two fuzzy numbers with different membership functions.

On such motivation basis, the main contributions of this study are summarized as follows: (1) In contrast to the existing approaches that uses crisp numbers or fuzzy numbers with the same membership functions to express customers' opinions, according to the proposed approach in this study fuzzy numbers with different membership functions are used to express customers' opinions in the case that their priorities are changed during product development, (2) To calculate the sum of different fuzzy numbers, we have improved an approximate summation method that significantly reduces the computational load, and (3) To the best of our knowledge, this study is the first attempt for solving the NRP using BAAA algorithm as a powerful meta-heuristic algorithm.

The remainder of the article is organized as follows. Section 2 gives a detailed account of the works done in this area. Section 3 describes the fuzzy concepts required in the present paper. Section 4 explores the proposed method for solving the next release problem. Section 5 reports and analyzes the results of the tests done on two fuzzy datasets. Section 6 concludes the paper and indicates future works.

2. Related works

Different methods have so far been introduced to solve the next release problem [13, 14]. Most of the ways introduced for solving this problem fall into two general categories. The first category consists of the Software Engineering Decision Support (SEDS) methods [15]. SEDS is a research field in which decision-making algorithms are used to help solve problems in the field of software engineering. This category includes methods such as QFD, AHP and fuzzy logic that usually focus on prioritizing and ranking the requirements.

The second category consists of the Search Based Software Engineering (SBSE) methods. SBSE is a research field in which search based optimization algorithms are used to solve problems in the field of software engineering [16]. Optimization methods such as linear programming, heuristic and metaheuristic algorithms belong to this category.

Sadiq and Jain [17] used a fuzzy method for eliciting goal-based requirements and prioritizing them. This method was formed out of a combination of fuzzy analytical hierarchy process and weighted relationships. Chopra et al. [18] used the AHP method based on cost-value prioritization techniques in several large projects. The authors of this paper prioritized both the functional and non- functional requirements [18]. Ramzan et al. [19] introduced the concept of requirement value for prioritizing requirements. In that paper, an intelligent technique was used for prioritizing the requirements based on a multi-level value using fuzzy logic [19]. Alrashoud and Abhari [20] used the Fuzzy Inference Engine to overcome the uncertainty of the problem and prioritize the requirements. The authors used three criteria,

namely risk, value of each requirement and effort, for decision-making. Alrashoud and Abhari [21] used ANFIS primarily to create membership functions and fuzzy rules in order to plan for development in the next release. Then, they used the fuzzy inference system to solve the problem. Authors of this paper considered three criteria, namely risk, stakeholders satisfaction and availability of resources. Lai et al. [22] proposed a new way to rank client requirements with respect to the competitive market. In that method, fuzzy QFD was used. In addition to client feedback, the authors also considered the products of rival software companies to rank the requirements. Souza et al. [23] presented an optimal release policy for multi release software system has been proposed by taking into consideration the test as well as the operational phase. Veerapen et al. [24] used the integer linear programming method in two-objective and single-objective models. The authors could find the exact solution to the single-objective problem and small multi-objective problems. However, in large-scale multi-objective problems, their proposed algorithm had a very high runtime. Mougouei [25] proposed a linear programming model for selecting requirements according to the interactions between them. In his paper, he also presented a technique for modeling the interaction between requirements by a graph. Araújo et al. [26] proposed an architecture based on machine learning and genetic algorithm to solve this problem. Jiang et al. [27] used a combination of the ACO algorithm with a local search algorithm to solve the next release problem. Masadeh et al. [28] used the gray wolf optimization algorithm to select the best subset of the proposed requirements. Sagrado et al. [4] used the ACS algorithm as a two-objective (increased satisfaction and cost reduction). In that paper, for the first time, three types of interactions between requirements were defined and considered in the implementation. Del Águila et al. [29] introduced a requirements management tool that uses artificial intelligence algorithms. This tool uses three algorithms Greedy Randomized Adaptive Search Procedure (GRASP), Ant Colony System (ACS) and NSGA II to solve the next release problem. Chaves et al. [30] used the multi-objective differential evolution algorithm, the multi-objective TLBO algorithm [8], and the multi-objective bee colony optimization algorithm [9], respectively to solve the next release problem. In all three of these papers, two datasets with crisp values and three types of interactions between requirements were considered. Pitangueira et al. [31] proposed a risk-aware multi-objective method to solve the next release problem aimed at reducing the risk of discontent among stakeholders. The authors used the NSGA II algorithm. Nayebi and Ruhe [32] considered two criteria, namely client satisfaction and dissatisfaction, for the first time and introduced the Asymmetric Release Planning (ARP) concept. Their goal was to select a subset of requirements so that their development would yield the highest degree of satisfaction, and lack of development of the rest of the requirements would create the least amount of dissatisfaction. To explain the ARP concept, they stated that the satisfaction level achieved by the development of a requirement does not correspond to the level of dissatisfaction achieved by a lack of development, and vice versa. Alrezaamiri et al. [7] introduced a fuzzy artificial chemical reaction optimization algorithm for solving the next release problem. The authors used fuzzy datasets for the first time. All datasets values were triangular fuzzy numbers. The authors reduced the uncertainty of the data by using the fuzzy triangular numbers instead of the crisp numbers [7].

3. Fuzzy concepts

In this section, we review some primary concepts of fuzzy sets [33–41].

Definition 1: Suppose X be a set and define the fuzzy set $\tilde{A}$ in X by its membership function $\mu_{\tilde{A}} : X \rightarrow [0,\ 1]$, where a real number $\mu_{\tilde{A}}(x)$ is assigned to each element $x \in X$ in the interval $[0,\ 1]$.

Definition 2: A normal fuzzy number is represented by $\tilde{A} = (m\ ,\ \sigma)$, where m is the median and σ is the standard deviation. The membership function $\tilde{A}$, defined by the formula (1). Figure 1.a shows a normal fuzzy number with values (4, 3).

$$\mu_{\tilde{A}}(x) = e^{-(\frac{x-m}{\sigma})^2},\ x \in R \tag{1}$$

Definition 3: A triangular fuzzy number is represented in the form of $\tilde{A} = (a_1, a_2, a_3)$ and with the membership function of formula (2). Figure 1.b shows a triangular fuzzy number with values (2, 4, 5).

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

Definition 4: A trapezoidal fuzzy number $\tilde{A}$ is represented by $\tilde{A} = (a_1\ ,\ a_2,\ a_3,\ a_4)$, with the mem-

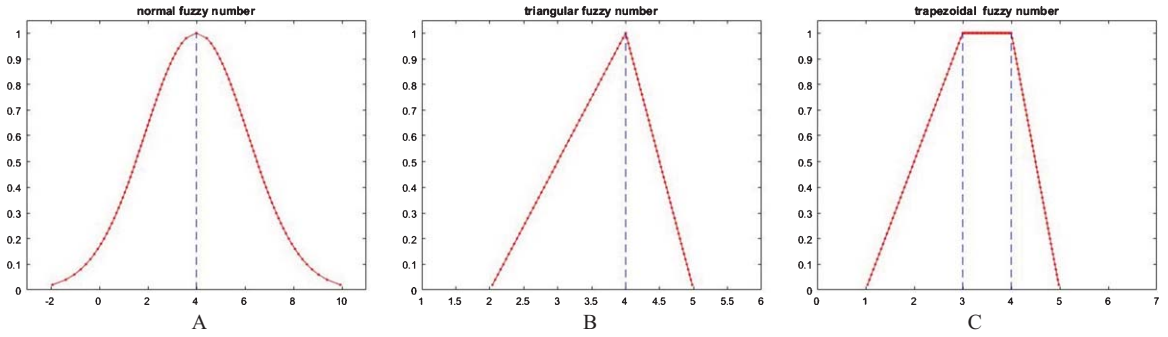

Fig. 1. (a) A normal fuzzy number. (b) A triangular fuzzy number. (c) A trapezoidal fuzzy number.

bership function of formula (3). Figure 1.c shows a trapezoidal fuzzy number with values (1, 3, 4, 5). The trapezoidal fuzzy number can be a triangular fuzzy number if $a_2 = a_3$.

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1}, & a_1 \le x \le a_2, \\ 1, & a_2 \le x \le a_3, \\ \frac{a_4-x}{a_4-a_3}, & a_3 \le x \le a_4. \end{cases} \quad (3)$$

The fuzzy addition on two trapezoidal fuzzy numbers $\tilde{A} = (a_1, a_2, a_3, a_4)$ and $\tilde{B} = (b_1, b_2, b_3, b_4)$ is given as:

$$\tilde{A} \tilde{+} \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4) \quad (4)$$

The fuzzy addition on two normal fuzzy numbers $\tilde{A} = (m_1, \sigma_1)$ and $\tilde{B} = (m_2, \sigma_2)$ is given as:

$$\tilde{A} \tilde{+} \tilde{B} = (m_1 + m_2, \sigma_1 + \sigma_2) \quad (5)$$

Definition 5: The α -cut of a fuzzy set $\tilde{A}$ is defined as a crisp set $\left[\tilde{A}\right]_\alpha$ in which the membership degrees of its elements exceed the level α, i.e. $\left[\tilde{A}\right]_\alpha = \left\{x \in X; \mu_{\tilde{A}}(x) \ge \alpha\right\} = \left[\tilde{A}^L_\alpha,\ \tilde{A}^R_\alpha\right]$.

Definition 6: The α –cut of the trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is given by $\left[\tilde{A}\right]_\alpha = \left[\tilde{A}^L_\alpha,\ \tilde{A}^R_\alpha\right] = [(a_2 - a_1)\alpha + a_1,\ a_4 - (a_4 - a_3)\alpha]$.

Definition 7: The α –cut of the normal fuzzy number $\tilde{A} = (m\ ,\ \sigma)$ is given by $\left[\tilde{A}\right]_\alpha = \left[\tilde{A}^L_\alpha,\ \tilde{A}^R_\alpha\right] = \left[m - \sigma\sqrt{-\ln(\alpha)},\ m + \sigma\sqrt{-\ln(\alpha)}\right]$.

The sum of two fuzzy numbers of the same type is easily done at time order O(1). However, the sum of two fuzzy numbers of different types cannot be easily accomplished and approximate sum methods should be used.

3.1. The improved fuzzy approximate sum operator

Hassanzadeh et al. [10], proposed a method for approximating the sum of normal and trapezoidal fuzzy numbers. In this model, approximate the sum and its corresponding membership function by dividing the α –interval, [0, 1], into n subintervals and letting $\alpha_0 = 0$; $\alpha_i = \alpha_{i-1} + \Delta\alpha_i$; $\Delta\alpha_i = \frac{1}{n}$; and $n = 1, 2, \ldots, n$.

Suppose $\tilde{A} = (a_1\ ,\ a_2,\ a_3,\ a_4)$ be trapezoidal fuzzy number and $\tilde{B} = (m\ ,\ \sigma)$ be normal fuzzy number. Given $\alpha_i \in (0,\ 1]$, $1 \le i \le n$, the α_i –cut sum of these fuzzy numbers using Definitions 6 and 7 is obtained as follows:

$$\left[\tilde{C}\right]_{\alpha_i} = \left[\tilde{C}^L_{\alpha_i},\ \tilde{C}^R_{\alpha_i}\right] = \left[\tilde{A}^L_{\alpha_i} + \tilde{B}^L_{\alpha_i},\quad \tilde{A}^R_{\alpha_i} + \tilde{B}^R_{\alpha_i}\right] =$$

$$\Big[(a_2 - a_1)\alpha_i + a_1 + m - \sigma\sqrt{-\ln(\alpha_i)},$$

$$a_4 - (a_4 - a_3)\alpha_i + m + \sigma\sqrt{-\ln(\alpha_i)}\Big] \quad (6)$$

Formula (6) can be used to obtain n points for $\tilde{C}^L_{\alpha_i}$ and n points for $\tilde{C}^R_{\alpha_i}$ using α_i, $1 \le i \le n$. In this method, approximated the membership function of the sum using the resulting points via the α –cut and Crammer's approach for fitting an exponential membership function for the sum. Let $x_i = \tilde{C}^R_{\alpha_i}$ and $y_i = \mu(\tilde{C}^R_{\alpha_i})$, and for n points $(x_i,\ y_i)$, consider the fitting model to be $y = e^{-\left(\frac{x-\lambda}{\beta}\right)^2}$. They proposed a least squares model to approximate the right membership function for the considered addition, and determined the unknown parameters λ and β as formulas (7) and

(8) [10, 41]:

$$\beta = \frac{n\sum_i \left(x_i \times \sqrt{-\ln y_i}\right) - \sum_i \sqrt{-\ln y_i} \times \sum_i x_i}{-n\sum_i \sqrt{-\ln y_i} - \sum_i \sqrt{-\ln y_i} \times \sum_i \sqrt{-\ln y_i}} \tag{7}$$

$$\lambda = \frac{\sum_i \ln y_i \left(-\sum_i x_i\right) - \sum_i \left(x_i \times \sqrt{-\ln y_i}\right) \times \sum_i \sqrt{-\ln y_i}}{-n\sum_i \sqrt{-\ln y_i} - \sum_i \sqrt{-\ln y_i} \times \sum_i \sqrt{-\ln y_i}} \tag{8}$$

Now, let $x_i = \tilde{C}^L_{\alpha_i}$ and $y_i = \mu(\tilde{C}^L_{\alpha_i})$, and consider the fitting model $y = e^{-\left(\frac{x-\lambda'}{\beta'}\right)^2}$. The least squares model for approximating the left membership function of the considered addition results in the unknown parameters λ' and β' as follows:

$$\beta' = \frac{n\sum_i \left(x_i \times \sqrt{-\ln y_i}\right) - \sum_i \sqrt{-\ln y_i} \times \sum_i x_i}{n\sum_i \sqrt{\ln y_i} + \sum_i \sqrt{-\ln y_i} \times \sum_i \sqrt{-\ln y_i}} \tag{9}$$

$$\lambda' = \frac{\sum_i \ln y_i \times \sum_i x_i + \sum_i \left(x_i \times \sqrt{-\ln y_i}\right) \times \sum_i \sqrt{-\ln y_i}}{n\sum_i \sqrt{\ln y_i} + \sum_i \sqrt{-\ln y_i} \times \sum_i \sqrt{-\ln y_i}} \tag{10}$$

Hence, the approximate membership function for the approximating sum of trapezoidal and normal fuzzy numbers is given as formula (11):

$$\mu_{\tilde{c}}(x) = \begin{cases} e^{-(\frac{\lambda'-x}{\beta'})^2}, & x < \lambda', \\ 1, & \lambda' \le x \le \lambda, \\ e^{-(\frac{x-\lambda}{\beta})^2}, & x > \lambda. \end{cases} \tag{11}$$

3.2. Distance between fuzzy numbers

The following formula [10, 41] can be used to compare two fuzzy numbers:

$$D_{p,q}(\tilde{A},\ \tilde{B}) = \begin{cases} \left[(1-q)\int_0^1 \left|A^-_\alpha - B^-_\alpha\right|^p d\alpha + q\int_0^1 \left|A^+_\alpha - B^+_\alpha\right|^p d\alpha\right], & p < \infty \\ (1-q)\sup_{0<\alpha\le 1}\left|A^-_\alpha - B^-_\alpha\right| + q\inf_{0<\alpha\le 1}\left|A^+_\alpha - B^+_\alpha\right|, & p = \infty \end{cases} \tag{12}$$

where the parameter p symbolizes the preference weight assigned to the end points of the support. If the expert has no priority, $D_{p,\frac{1}{2}}$ 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 given by:

$$D_{p,q}(\tilde{A},\ \tilde{B}) = \left[(1-q)\sum_{i=1}^{n}\left|A^-_{\alpha_i} - B^-_{\alpha_i}\right|^p + q\sum_{i=1}^{n}\left|A^+_{\alpha_i} - B^+_{\alpha_i}\right|^p\right]^{\frac{1}{p}} \tag{13}$$

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

$$D_{2,\frac{1}{2}}(\tilde{A},\ \tilde{B}) = \sqrt{\left[\frac{1}{2}\sum_{i=1}^{n}|A^-_{\alpha_i} - B^-_{\alpha_i}|^2 + \frac{1}{2}\sum_{i=1}^{n}|A^+_{\alpha_i} - B^+_{\alpha_i}|^2\right]} \tag{14}$$

To compare two fuzzy numbers $\tilde{A}$ and $\tilde{B}$ using the α_i -cuts, we compare them to $\tilde{0} = (0, 0, ..., 0)$. In fact, formula (14) 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 $D_{2,\frac{1}{2}}(\tilde{A},\ \tilde{0}) \le D_{2,\frac{1}{2}}(\tilde{B},\ \tilde{0})$.

4. Artificial Alga Algorithm to solve the next release problem

In this section, we first introduce the Artificial Alga Algorithm. Then, we will solve the next release problem with different fuzzy arcs using this algorithm.

4.1. Binary Artificial Alga Algorithm (BAAA)

The Artificial Algae Algorithm (AAA) is an algorithm that inspired by behavior of the algae colonies. The binary artificial alga algorithm first begins by adjusting the parameter values and generating the initial population. An alga structure is defined to represent the solutions of the problem. The fitness value of each alga is calculated and maintained in the colony. The algorithm has a specified number of iterations, and iterations continue until the termination condition is reached. An update operator is applied to the algae in each iteration. The algorithm has two types of updating mechanisms. The

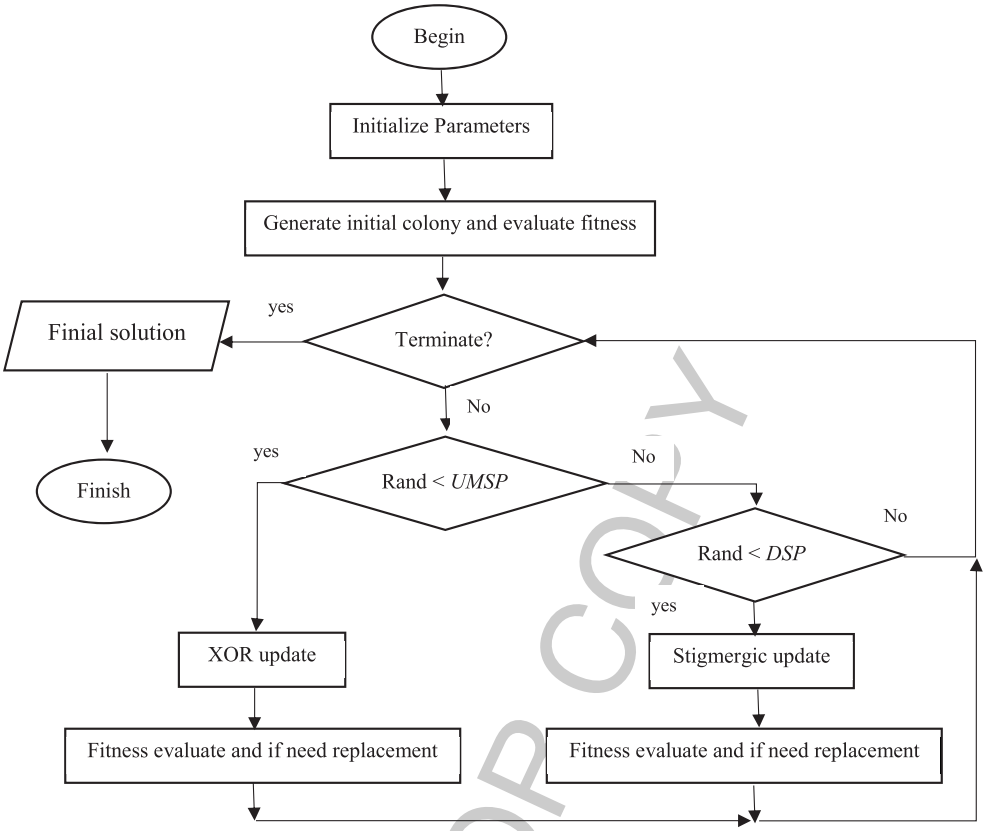

Fig. 2. BAAA flowchart.

first kind is logic XOR and the second kind is based on stigmergic behavior.

After each update, if the fitness value of the new alga was more favorable than that of the old alga, it replaces the old one. Otherwise, the algorithm continues with the same old alga. After the algorithm reaches the end, the most desirable alga is selected as the solution to the problem [42]. Figure 2 shows the (BAAA) flowchart.

4.2. *The BAAA for solving next release problem with fuzzy arc lengths*

In this subsection, we describe an encoding scheme and initialization for the BAAA. Then, we express our new idea for calculating the fitness of each individual. BAAA updates operators including the logic XOR and stigmergic behavior.

4.2.1. *BAAA encoding scheme and population initialization*

In the NRP problem, each solution is encoded as an n-dimensional vector in which the *i-th* cell on the

r_1	r_2	r_3	...	r_n
1	0	1		0

Fig. 3. Structure an alga.

vector corresponds to the *i-th* requirement in the R set, where $i = 1, 2, 3, ..., n$. In Fig. 3, a solution is given. In each solution, each requirement that is selected for development in the next release accepts value 1, or otherwise 0 in its corresponding cell.

Initial colony of the algorithm are generating according to formula (15). Here, $rand_{i,j}$ is a random number between [0,1] and $X_{i,j}$ is the *j-th* dimension of the *i-th* algae colony.

$$X_{i,j} = \begin{cases} 0, & if(rand_{i,j} < 0.5) \\ 1, & otherwise \end{cases}$$

$$i = 1, 2, ..., n \ \ and \ \ j = 1, 2, ..., d \qquad (15)$$

After generating the initial colony, in each stage of the algorithm, in case the algae values violate the con-

straints defined in the problem, that alga is modified in a validation function.

4.2.2. Fitness evaluation

In this problem, each alga is a solution to the NRP problem. In fact, each solution shows the selection of a subset of the proposed requirements (m requirements). The number of m related to the selected requirement in each alga is a value between 0 and n ($0 \leq m \leq n$). The cost and satisfaction level of each alga is obtained from the sum of each cost value and the individual satisfaction values of the selected requirements. To add these fuzzy numbers together, we used the least squares model proposed by Hassanzadeh et al. [10] optimally. Hassanzadeh et al. used $m-1$ approximations in accordance with formulas (6-11) to add m various fuzzy numbers (trapezoidal and normal numbers) together. That is, they selected two fuzzy numbers (regardless of the fuzzy type of those two numbers) at first, and subjected the two numbers to alpha-cutting. Then, they used the sum of alpha-cuts on the left and right sides of the two numbers for approximation. Subsequently, they approximated the number obtained from the first approximation in the same way using the third fuzzy number. These approximations continued until the last fuzzy number was reached. Finally, formula (14) was used to calculate the amount of fitness resulting from the approximation of the entire fuzzy numbers.

In this study, an interesting idea was used to dramatically reduce the time it takes to perform these calculations. In our proposed approach for each alga, first, the trapezoidal numbers of requirements are added together in O(1) according to formula (4). Then, the values of the normal numbers are added together in O(1) according to formula (5). In the end, to calculate the total amount of satisfaction or cost, we obtain the approximate sum (according to formulas 6-11) of the sum of trapezoidal numbers and the sum of normal numbers. Finally, we use formula (14) to calculate the fitness of the approximate sum. In fact, unlike Hassanzadeh et al. who used $m-1$ approximations to add m fuzzy numbers of different types, we only use 1 approximation.

4.2.3. Logic XOR operator

Algorithm have two kind of update mechanism. First approach uses logic XOR operator for producing children solutions and second approach is based on stigmergic behavior. Formula (16) is utilized for determining which update mechanism is used for obtain a candidate solution [11].

$$update\,rule = \begin{cases} \log ic\ XOR, & if(rand < UMSP) \\ \quad and\ C_{01}(t) \neq 0\ and\ C_{10}(t) \neq 0 & \\ stigmergic, & otherwise \end{cases} \tag{16}$$

Where, *UMSP* is the update mechanism selection probability as a control parameter and *rand* is a random number between [0,1]. C_{01} and C_{10} are integer numbers. In logic XOR update, for each algae colony, at the first step select a neighbor via random choice. At the second step, randomly select three alga dimension of colony. At the third step, modify the colony using formula (17).

$$\begin{aligned} T &= Y_i \\ T_j &= Y_{i,j} \oplus [\varphi\,(Y_{i,j} \oplus Y_{z,j})] \\ T_k &= Y_{i,k} \oplus [\varphi\,(Y_{i,k} \oplus Y_{z,k})] \\ T_l &= Y_{i,l} \oplus [\varphi\,(Y_{i,l} \oplus Y_{z,l})] \\ i, z &\in \{1, 2, ..., SizeColony\}, \quad i \neq z, \\ j, k, l &\in \{1, 2, ..., n\}\ ,\ j \neq k \neq l \end{aligned} \tag{17}$$

where, T is the child solution, Y is the individual in the colony, φ is the logic NOT operator with 50% probability, $\oplus$ is the logic XOR operator, and n is the number of requirements. Therefore, T_j, T_k, T_l sequential determines the *j-th*, *k-th* and *l-th* requirement of the child solution. After update, the fitness value of the child solution is compared with the parent. If fitness value of the child solution is better than the parent, the child solution is replaced to the parent alga and the information to be used for the second update mechanism is obtained at this stage [11]. Algae of parent and child solutions are compared one by one and changed decision variables are determined. If the parent value of the decision variable is 1 and the child value of the decision variable is 0, the C_{10} counter is addition by 1. On the contrary, if the parent value of the decision variable is 0 and the child value of the decision variable is 1, the C_{01} counter is addition by 1. In other words, C_{10} indicates the count of changed value 1 to 0 and C_{01} indicates the count of altered value 0 to 1. C_{01} and C_{10} counters are calculated given as formulas (18) and (19):

$$C_{01}(t+1) = \begin{cases} C_{01}(t) + 1, & fitness(T) < fitness(Y_i) \\ and\ Y_{i,d} = 0 \quad and\ T_d = 1 & \\ C_{01}(t), & otherwise \end{cases} \tag{18}$$

$$C_{10}(t+1) = \begin{cases} C_{10}(t) + 1, & fitness(T) < fitness(Y_i) \\ \quad and\ Y_{i,d} = 1\ and\ T_d = 0 \\ C_{10}(t), & otherwise \end{cases} \tag{19}$$

Here, T is the child solution and Y is the individual in the colony [11].

4.2.4. Stigmergic behavior operator

In stigmergic update rule, indicators C_{01} and C_{10} are used to update the solutions in the colony given as formulas (20) and (21):

$$P_{01}(t+1) = \frac{C_{01}(t)}{C_{01}(t) + C_{10}(t)} \tag{20}$$

$$P_{10}(t+1) = \frac{C_{10}(t)}{C_{01}(t) + C_{10}(t)} \tag{21}$$

Where, $P_{01}(t+1)$ is the probability rate of C_{01} in iteration $t+1$ and $P_{10}(t+1)$ is the probability rate of C_{10} in iteration $t+1$. These probabilities are used to calculate the child solution. *Dimension selection probability (DSP)* is a control parameter in the algorithm. If random number generated for dimension in range of [0, 1] is less than the *DSP* parameter, movement is performed at this dimension. Assume i and j are the index of ones and zeros in child solution V, respectively. Let i and j be random integers between 1 and sizes of i and j, respectively. Therefore, V_I indicates random dimension that have a value of 1 and V_J indicates a random dimension that have value of 0. The child solution V is calculated given as formulas (22) and (23):

$$V_I = \begin{cases} 0 & , rand < P_{10} \\ V_I & , otherwise \end{cases} \tag{22}$$

$$V_J = \begin{cases} 1 & , rand \geq P_{10} \\ V_J & , otherwise \end{cases} \tag{23}$$

where, *rand* is a random number generated between [0,1]. If *rand* value is less than P_{10} probability value, random decision variable in child solution with value 1 V_I is set to 0. If *rand* value is equal or more than the P_{10} probability value, random decision variable in child solution with value 0 V_J is set to 1. If fitness value of the child solution is better than the parent, the child solution is replaced to the parent alga colony [11].

The algorithm will continue to reach the condition criteria and then will output best solution.

5. Tests and results

In this section, we will do tests on two randomized datasets. Since there are no fuzzy datasets with different arcs in the related works, we cannot compare the results of our method with them. All tests were carried out by using the MATLAB. The first dataset includes 24 requirements and 12 interactions between requirements. In Table 1, the cost and the satisfaction of development of each requirement are presented as various kinds of fuzzy numbers (trapezoidal and normal fuzzy numbers). The satisfaction level of requirements is taken from clients by surveys or other requirement engineering tools. To make it easier for clients to understand, these surveys are conducted using crisp numbers. For example, in a poll, clients are requested to assign a score between 0 and 10 as their satisfaction with the development of each requirement. The score 0 is the lowest level of satisfaction and the score 10 is the highest level of satisfaction for the development of each requirement in the next release. The values obtained from the requirements satisfaction survey will be converted to fuzzy numbers at the discernment of the development team. The development team also considers a fuzzy number as the cost of developing each requirement and defines the interaction between requirements.

Remark 1: Expressing satisfaction levels in a questionnaire can be considered as a complex task as customers have multiple opinions under uncertainty and also the variability, diversity and subjectivity associated with an accurate satisfaction level is usually lost. An additional concern arising with expressing overall satisfactions is the fact that the opinions of customers during requirement development are not homogeneous as time goes on. Moreover, if satisfaction levels are heterogeneous, it matters how and to what extent it influences the overall satisfaction of requirement. Average scores are supposed to hide the real situation. To manage these disadvantages there is an alternate approach which takes into account that the nature of most attributes related to evaluations, judgements involve subjectivity and certain imprecision. In fact, fuzzy numbers are expressive enough to find a value in it fitting appropriately the valuation, opinion, judgement involving subjective perceptions in most real life situations.

Table 1
Dataset 1

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
Satisfaction Cost	(1,3,8,9)	(5,3)	(0,3,5,9)	(4,5,6,7)	(5,7)	(1,3,5,9)	(1,1)	(0,3,4,5)
	(2,4)	(1,3)	(1,2,3,4)	(5,5)	(2,4,6,7)	(3,5)	(2,1)	(1,2)
	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}
Satisfaction Cost	(6,7,8,9)	(10,2)	(1,3,6,7)	(2,1)	(2,3,6,8)	(0,2,3,4)	(8,6)	(4,6,7,10)
	(6,6)	(2,3,4,5)	(2,6)	(1,2,3,4)	(5,6)	(1,1)	(2,3,5,6)	(1,3,4,5)
	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}	r_{24}
Satisfaction Cost	(3,4,6,8)	(0,2,3,8)	(4,5,6,7)	(10,3)	(2,3,6,6)	(8,7)	(5,7,8,9)	(10,5)
	(1,2,3,4)	(4,3)	(2,5)	(3,4,5,6)	(1,4)	(1,4,5,6)	(3,3)	(1,3,4,5)
$r_4 \Rightarrow r_{10}$	$r_{11} \Rightarrow r_{17}$		$r_{14} \Rightarrow r_{17}$	$r_{12} \Rightarrow r_{15}$		$r_4 \oplus r_7$	$r_1 \oplus r_{15}$	
$r_{11} \oplus r_6$	$r_{21} \oplus r_{22}$		$r_3 \otimes r_{10}$	$r_{22} \otimes r_{19}$		$r_2 \otimes r_9$	$r_5 \otimes r_{24}$	

They have the ability to model the imprecision of human satisfaction levels, formalize them mathematically, to 'precisiate' them in a continuous way, and to develop mathematical computation with them. This approach leads to a fuzzy-valued response format enabling a level of variability and accuracy which would not be captured when using a crisp number.

Remark 2: The membership functions of fuzzy numbers can have various shapes. The literatures in fuzzy expressing of satisfaction levels mainly employ trapezoidal membership functions, however, they are not always consistent with human thinking and judgement, since in these cases the membership functions have the same slope on the whole interval. In reality, the customer's opinions change only slightly around points that represent the worst and the best possible views on the satisfaction levels. In such cases, normal membership function together with trapezoidal one are used to enhance the reliability of opinions. That is, the slope of the normal function is not constant; therefore, applying different membership functions to establish fuzzy numbers results in a more precise reflection of human thinking and judgement.

In this test, we plan to select the best subset of the requirements that yields the highest degree of satisfaction by taking into account a different budget constraint each time. Table 2 considers 8 different cost limits. For example, within an 80% cost limit, a subset of requirements should be selected with a cost of development of less than 292.6 units so that it yields the maximum satisfaction level. Within this cost limit, the binary artificial alga algorithm can select a subset with a development cost of 289.07 units and a satisfaction level of 499.57 units. Table 2 shows the approximate value of the total sum of cost and satisfaction of this subset of the requirements in the form of $[\lambda,\ \beta,\ \lambda',\ \beta']$ according to formula (11). The crisp values of satisfaction and total cost (columns 2 and 3 of Table 2), are obtained by applying formula (14) to the fuzzy number of approximate sum of cost and satisfaction (columns 4 and 5 of Table 2). Figure 4 shows the bar graphs of satisfaction and cost of the best selected subsets in every iteration. Figure 4.a is with 60% cost limit and Fig. 4.b is without cost limits.

Table 2 shows the runtime of the algoriwith any cost limit. As can be seen, as the cost limit gets more severe, the runtime of the algorithm also increases. The reason for this is the algae validation function. After generating the initial colony and any update operator, the algae are sent to the validation function to verify the validity and non-violation of cost limits and interaction. In this function, if the cost of developing the selected requirements is greater than the limit value, a requirement is randomly selected and removed from the subset. This trend continues until the amount of alga cost violates the amount of budget constraints. The stronger the cost limit, the more the number of calls for this recursive function which increases the runtime of the entire algorithm. In the absence of any cost limit, the algorithm has the fastest runtime. In this case, if the algorithm can select all the requirements in the selected subset, it will provide the highest satisfaction. The finish criterion of the algorithm was performing 100 iterations. Table 3 shows other parameters used when simulating the first and second datasets.

Figure 5 shows a comparison of the runtime of the algorithm in two modes of the approximate sum of fuzzy numbers. The red chart shows the algorithm runtime together with approximate sum of Hassanzadeh et al. [10] in which $m - 1$ approximations were used for adding m fuzzy numbers together. In our proposed method, all trapezoidal numbers are first added together at time O(1), and then all of the normal numbers are added together at time O(1). The total sum is obtained with 1 approximation from the sum of the

Table 2
Test results on the first data

Cost limit	Crisp satisfaction	Crisp cost	Fuzzy number cost	Fuzzy number satisfaction	Time (s)
Without cost limit	546.61	365.7	[59.78, 53.81, 67.48, 51.21]]	[101.42, 39.32, 123.51, 39.97]	0.83
90%=329.1	522.08	318.79	[50.78, 47.81, 58.48, 45.21]	[97.23, 40.37, 117.90, 39.07]	0.95
80%=292.6	499.57	289.07	[45.71, 44.16, 52.55, 41.56]	[91.29, 39.02, 111.64, 39.67]	1.19
70%=256.0	476.19	255.67	[40.71, 38.16, 47.55, 35.56]	[88.23, 38.37, 105.77, 38.37]	1.42
60%=219.4	427.9	219.31	[33.58, 33.86, 39.61, 31.91]	[79.97, 35.76, 96.29, 33.16]	1.62
50%=182.8	389.80	175.95	[29.65, 23.51, 35.61, 20.91]	[75.78, 33.81, 86.55, 30.56]	1.78
40%=146.3	332.17	146.15	[24.52, 19.21, 29.68, 17.25]	[64.32, 29.25, 72.87, 27.30]	2.03
30%=109.7	261.10	108.02	[19.58, 12.86, 23.68, 10.25]	[52.19, 21.95, 55.81, 21.95]	2.47

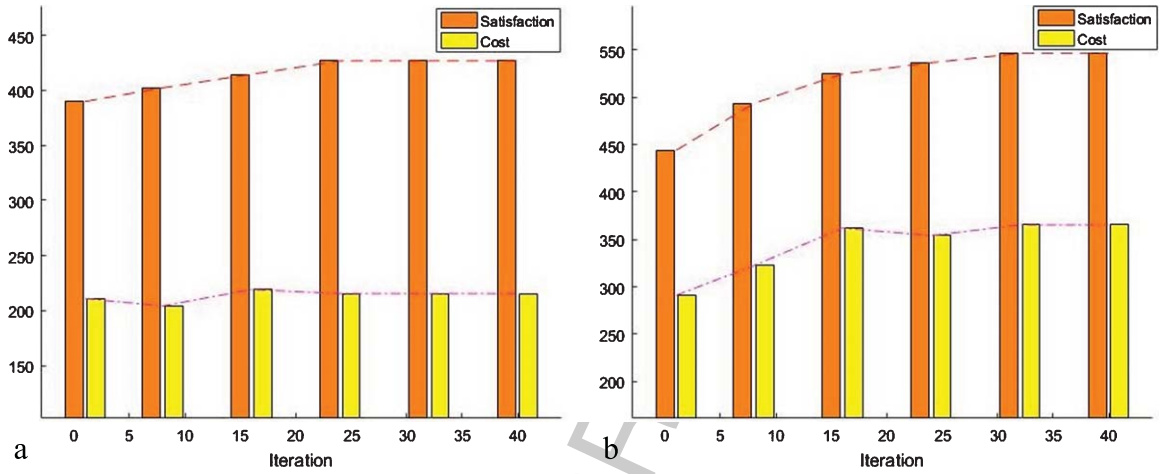

Fig. 4. (a) The graph of the highest satisfaction level and its related costs in iterations within the 60% cost limit. (b) The graph of the highest satisfaction level and its related costs in iterations without a cost limit.

Table 3
Other parameters of the simulation

Parameters	Values
Number of α-cut	20
Size colony	40
Iterations	100
UMSP	0.5
DSP	0.66

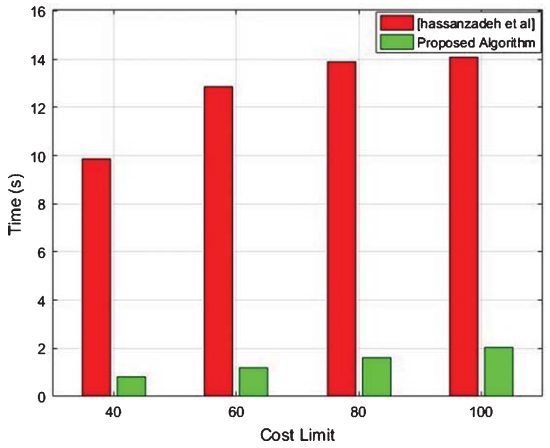

Fig. 5. A comparison of the runtime of the algorithm with two different approximate sum methods.

trapezoidal numbers and the sum of normal numbers. The green graph shows the runtime of our proposed algorithm.

In Fig. 5, within a 100% cost limit (or in other words, without cost limit), the difference between the runtime of the two algorithms is more than 12 seconds. While all implementation conditions are equal in the two algorithms, our idea of an optimal approximate sum method leads to an 85% reduction of the runtime. Again, it is worth emphasizing that in Figs. 5 and 7, the only difference is the method of adding the different fuzzy numbers together. In addition, the metaheuristic algorithm used in the two is the same.

To prove the quality of the proposed algorithm, we repeated all previous tests on a larger and more complex dataset. The second dataset includes 72 requirements and 20 interactions between requirements. In this dataset, the range of clients' scores

Table 4
Dataset 2

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
Satisfaction	(11,13,18,19)	(15,7)	(0,3,5,9)	(4,5,8,11)	(5,7)	(11,14,15,18)	(11,10)	(0,3,4,5)
Cost	(11,4)	(6,3)	(1,2,3,4)	(9,5)	(2,4,6,7)	(17,11)	(12,11)	(1,2)
	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}
Satisfaction	(16,17,18,19)	(10,15)	(1,3,6,7)	(2,1)	(4,5,6,9)	(0,2,3,4)	(18,6)	(7,8,9,12)
Cost	(19,14)	(12,13,14,15)	(2,6)	(1,3)	(5,6)	(2,4)	(12,13,15,16)	(6,7,9,10)
	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}	r_{24}
Satisfaction	(13,14,16,20)	(0,2,3,8)	(4,5,6,9)	(19,3)	(2,3,7,8)	(18,7)	(5,7,8,9)	(20,2)
Cost	(10,12,13,14)	(9,3)	(8,5)	(8,8,9,10)	(1,4)	(10,14,15,16)	(13,3)	(7,8,9,13)
	r_{25}	r_{26}	r_{27}	r_{28}	r_{29}	r_{30}	r_{31}	r_{32}
Satisfaction	(1,3,8,9)	(11,6)	(0,3,5,9)	(4,5,6,7)	(15,17)	(10,13,15,19)	(1,1)	(0,3,4,5)
Cost	(2,4)	(7,3)	(1,2,3,4)	(5,5)	(2,4,6,7)	(13,11)	(2,1)	(1,2)
	r_{33}	r_{34}	r_{35}	r_{36}	r_{37}	r_{38}	r_{39}	r_{40}
Satisfaction	(16,17,18,19)	(18,2)	(1,3,6,7)	(2,1)	(2,3,6,8)	(10,12,13,14)	(8,5)	(5,6,7,10)
Cost	(17,6)	(8,9,10,11)	(2,6)	(1,3)	(5,6)	(11,11)	(4,8,12,16)	(1,2,4,5)
	r_{41}	r_{42}	r_{43}	r_{44}	r_{45}	r_{46}	r_{47}	r_{48}
Satisfaction	(3,4,6,8)	(0,2,3,7)	(2,3,6,7)	(12,8)	(1,3,6,9)	(16,7)	(5,7,8,9)	(13,9)
Cost	(8,9,10,12)	(5,4)	(4,3)	(5,7,8,9)	(1,4)	(10,14,15,16)	(6,4)	(4,7,8,9)
	r_{49}	r_{50}	r_{51}	r_{52}	r_{53}	r_{54}	r_{55}	r_{56}
Satisfaction	(1,3,8,9)	(14,8)	(0,3,5,9)	(7,8,9,13)	(5,7)	(8,13,15,19)	(3,2)	(0,3,4,5)
Cost	(2,4)	(13,5)	(1,2,3,4)	(7,5)	(2,4,6,7)	(13,14)	(2,1)	(1,2)
	r_{57}	r_{58}	r_{59}	r_{60}	r_{61}	r_{62}	r_{63}	r_{64}
Satisfaction	(15,17,19,20)	(20,2)	(1,3,6,7)	(2,2)	(1,4,7,8)	(0,2,3,4)	(8,6)	(4,6,7,10)
Cost	(16,17)	(12,13,14,15)	(5,6)	(1,3)	(4,6)	(3,2)	(2,3,5,6)	(1,3,4,5)
	r_{65}	r_{66}	r_{67}	r_{68}	r_{69}	r_{70}	r_{71}	r_{72}
Satisfaction	(3,4,6,8)	(4,6,8,10)	(5,6,7,8)	(10,3)	(0,3,5,7)	(18,17)	(5,7,8,9)	(17,9)
Cost	(1,2,3,4)	(10,15)	(7,6)	(3,4,5,6)	(6,3)	(2,4,5,6)	(3,3)	(2,4,6,8)
$r_{34} \Rightarrow r_{20}$		$r_{21} \Rightarrow r_{47}$		$r_{34} \Rightarrow r_{37}$		$r_{40} \Rightarrow r_{57}$		$r_{12} \Rightarrow r_{25}$
$r_{32} \Rightarrow r_{29}$		$r_{43} \oplus r_{39}$		$r_{52} \oplus r_{69}$		$r_{11} \oplus r_{30}$		$r_{23} \oplus r_{24}$
$r_{55} \oplus r_{51}$		$r_{56} \oplus r_{72}$		$r_{33} \oplus r_{38}$		$r_{61} \otimes r_{37}$		$r_{20} \otimes r_{51}$
$r_{72} \otimes r_{64}$		$r_{17} \otimes r_{38}$		$r_{30} \otimes r_{27}$		$r_{45} \otimes r_{66}$		$r_{42} \otimes r_{64}$

Table 5
Test results on second dataset

Max cost (s)	Crisp satisfaction	Crisp cost	Fuzzy number cost	Fuzzy number satisfaction	Time
Without cost limit	2881.0	2308.0	[440.78, 250.99, 471.87, 244.48]	[565.78, 218.17, 632.02, 220.12]	0.89
90%=2130.0	2725.4	2122.5	[408.7, 222.3, 438.9, 215.8]	[535.0, 209.3, 595.7, 211.9]	1.08
80%=1892.8	2538.08	1882.54	[363.8, 189.0, 394.9, 182.4]	[495.9, 201.0, 553.0, 203.0]	1.16
70%=1656.2	2334.83	1652.26	[322.52, 159.39, 350.13, 152.88]	[454.78, 192.99, 501.70, 198.20]	1.52
60%=1419.6	2114.14	1409.71	[278.65, 124.69, 307.25, 115.57]	[410.91, 186.29, 452.09, 186.29]	1.80
50%=1183.0	1844.62	1176.21	[228.33, 105.43, 255.19, 100.23]	[364.46, 159.73, 393.54, 159.73]	2.09
40%=946.4	1587.07	942.0	[187.07, 74.83, 210.64, 67.67]	[313.07, 139.83, 338.00, 139.17	2.65
30%=709.8	1314.78	703.43	[141.00, 52.18, 160.77, 44.37]	[257.87, 109.88, 283.06, 110.53]	3.16

of satisfaction with the proposed requirements in the polls is between 0 and 20. Table 4 shows the second dataset.

In the second dataset, a test is performed with eight different cost limits. Table 5 tabulates the values obtained from this test. For example, within the 60% cost limit equal to 1419.6 units, the satisfaction level of the requirements selected by the algorithm for development in the next release is equal to 2114.14 units and the related cost of development is 1409.71 units. In Fig. 6, the bar charts show the satisfaction level and cost of the best selected subsets for each iteration. Figure 6.a is related to the 70% cost limit and Fig. 6.b is related to the unlimited cost situation.

The second dataset has a longer runtime due to having a higher number of requirements and more restrictions than the first dataset. Figure 7 shows a comparison of the runtime of the algorithm in two modes of approximate sum of fuzzy numbers. The red chart is related to the runtime of the algorithm with approximate sum used in Hassanzadeh et al. [10], while the turquoise colored chart is related to the runtime of our proposed algorithm. While the runtime of the algorithm is approximated to be 1 minute

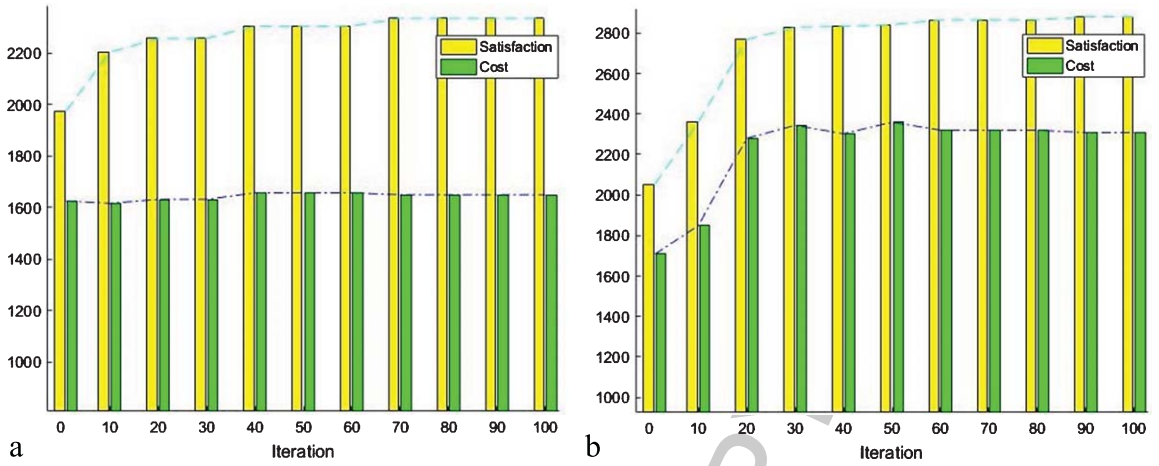

Fig. 6. (a) The graph of the most satisfaction level and its related costs in iterations within the 70% cost limit. (b) The graph of the most satisfaction level and its related costs in iterations without cost limit.

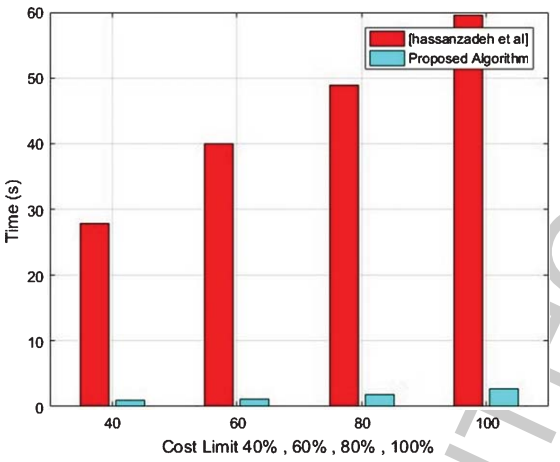

Fig. 7. A comparison of the runtime of the two algorithms.

using Hassanzadeh et al.'s approximate sum method within a 100% time limit, the runtime of the proposed algorithm is less than 3 seconds. This is indicative of a 20-fold reduction of runtime, which is very important for metaheuristic algorithms.

Figure 8 shows the convergence diagrams of the binary artificial alga algorithm in unlimited cost mode on both datasets. Figure 8.a and 8.b are related to the first and second datasets, respectively. In these two figures, convergence refers to the convergence of the satisfaction level of the subsets selected by the algorithm. The algorithm converges in the 33rd iteration due to the smallness of the first dataset, but converges in the second dataset in the 82nd iteration.

6. Conclusion

In this paper, we provided a solution procedure for one of the problem during software development. The next release problem is one of the problems facing development teams in incremental software development methods. In each release, a set of requirements is suggested by clients for development. Usually, due to constraints such as lack of funds, lack of time and technical difficulties, the development team cannot develop all the proposed requirements. In each release, the development team must select a subset of the proposed requirements so that it can provide the highest degree of satisfaction for its clients and the lowest cost to the company by considering the constraints of the problem. In large projects there should be a reliable instrument to help with the decision-making process and selection the development team makes. In this problem, the development team should assign a value of satisfaction, as well as another value of cost to each requirement. In order for the problem to be more applicable, allocation of fuzzy values is better than the allocation of crisp ones. In this paper, an optimized approximate sum model was used to calculate the summation of different kinds of fuzzy numbers, which greatly reduced the load of computations compared to the previous method. We also used a binary artificial alga algorithm to solve the problem. To test the quality of the proposed method, we conducted complete experiments on two dummy fuzzy datasets. The results of the proposed algorithm are very reliable, lack human errors, and can help the development team with their decision-making. The

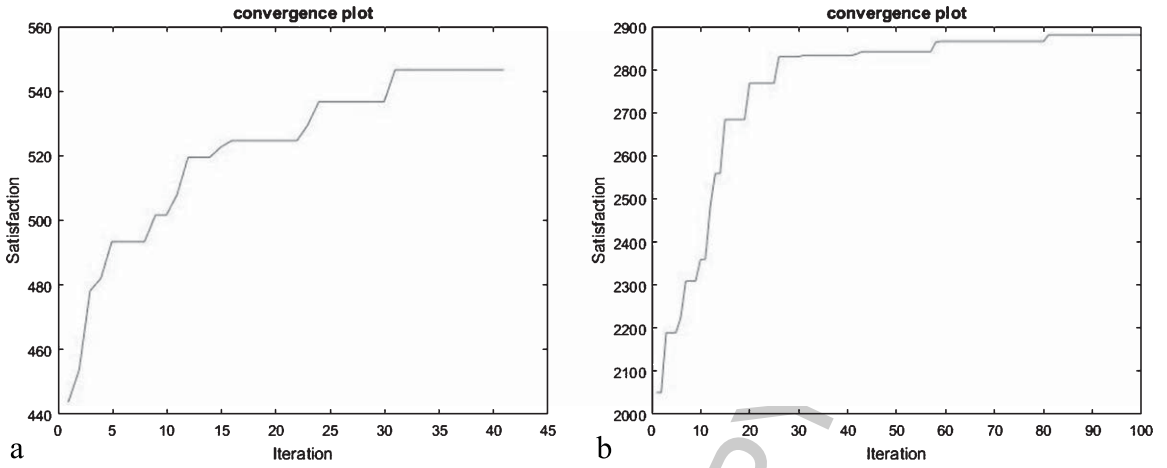

Fig. 8. (a) The convergence diagram of the binary artificial alga algorithm when solving the first dataset. (b) The convergence diagram of the binary artificial alga algorithm when solving the second dataset.

runtime of the proposed algorithm is very short. The subset introduced by the proposed algorithm is without human error and provides the highest level of client satisfaction. Future researches in this scope are as follows.

- Introducing new methods for approximating fuzzy numbers with less computation.
- Introducing automated methods for converting crisp values obtained from surveys to fuzzy numbers.
- Using other metaheuristic algorithms or combining them to solve the problem.

Acknowledgment

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

References

[1] G. Brau, J. Hugues and N. Navet, Towards the systematic analysis of non-functional properties in Model-Based Engineering for real-time embedded systems. *Science of Computer Programming* **156** (2018), 1–20.

[2] J. Jia, X. Yang, R. Zhang and X. Liu, Understanding software developers' cognition in agile requirements engineering. *Science of Computer Programming*, **178** (2019), 1-19.

[3] A.J. Bagnall, V.J. Rayward-Smith and I.M. Whittley, The next release problem, *Information and software technology* **43**(14) (2001), 883-890.

[4] J. Del Sagrado, I.M. Del Águila and F.J. Orellana, Multi-objective ant colony optimization for requirements selection, *Empirical Software Engineering* **20**(3) (2015), 577–610.

[5] O. Malgonde and K. Chari, An ensemble-based model for predicting agile software development effort, *Empirical Software Engineering* **24**(2) (2019), 1017-1055.

[6] L.L. Minku, A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation, *Empirical Software Engineering* (2019), 1–52.

[7] H. Alrezaamiri, A. Ebrahimnejad and H. Motameni, Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm, *Soft Computing* **23** (2019), 9979-9994.

[8] J.M. Chaves-González, M.A. Perez-Toledano and A. Navasa,Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection, *Engineering Applications of Artificial Intelligence* **43** (2015), 89-101.

[9] J.M. Chaves-González, M.A. Perez-Toledano and A. Navasa, Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm, *Knowledge-Based Systems* **83** (2015), 105-115.

[10] R. Hassanzadeh, I. Mahdavi, N. Mahdavi-Amiri and A. Tajdin, A genetic algorithm for solving fuzzy shortest path problems with mixed fuzzy arc lengths, *Mathematical and Computer Modelling* 57(1–2) (2013), 84-99.

[11] S. Korkmaz and M.S. Kiran, An artificial algae algorithm with stigmergic behavior for binary optimization. *Applied Soft Computing* (2018).

[12] S.A. Uymaz, G. Tezel and E. Yel, Artificial algae algorithm (AAA) for nonlinear global optimization, *Applied Soft Computing* **31** (2015), 153–171.

[13] A.M. Pitangueira, R.S.P. Maciel and M. Barros, Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature, *Journal of Systems and Software* **103** (2015), 267-280.

[14] A. Hudaib, R. Masadeh, M.H. Qasem and A. Alzaqebah, Requirements prioritization techniques comparison, *Modern Applied Science* **12**(2) (2018), 62.

[15] M.A.A. Féris, O. Zwikael and S. Gregor, QPLAN: Decision support for evaluating planning quality in software development projects, *Decision Support Systems* **96** (2017), 92-102.

[16] Y. Li, T. Yue, S. Ali and L. Zhang, Zen-ReqOptimizer: a search-based approach for requirements assignment optimization, *Empirical Software Engineering* **22**(1) (2017), 175-234.

[17] M. Sadiq and S.K. Jain, Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process, *International Journal of System Assurance Engineering and Management* **5**(4) (2014), 711-723.

[18] R.K. Chopra, V. Gupta and D.S. Chauhan, Experimentation on accuracy of non-functional requirement prioritization approaches for different complexity projects, *Perspectives in Science* **8** (2016), 79-82.

[19] M. Ramzan, M.A. Jaffar and A.A. Shahid, Value based intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique, *International Journal of Innovative Computing, Information and Control* **7**(3) (2011).

[20] M. Alrashoud and A. Abhari, Perception-based software release planning, *Intelligent Automation & Soft Computing* **21**(2) (2015), 175-195

[21] M. Alrashoud and A. Abhari, Planning for the next software release using adaptive network-based fuzzy inference system, *Intelligent Decision Technologies* **11**(2) (2017), 153-165.

[22] X. Lai, M. Xie, K.C. Tan and B. Yang, Ranking of customer requirements in a competitive environment, *Computers & Industrial Engineering* **54**(2) (2008), 202-214.

[23] J.T. Souza, C.L. Brito Maia, T.N. Ferreira, R.A. Ferreira and M.M. Albuquerque, An ant colony optimization approach to the software release planning with dependent requirements, in: Proc. of the 3rd Int. Symposium on Search Based Software Engineering (SBSE'11), 2011, pp. 142–157.

[24] N. Veerapen, G. Ochoa, M. Harman and E.K. Burke, An integer linear programming approach to the single and bi-objective next release problem, *Information and Software Technology* **65** (2015), 1–13.

[25] D. Mougouei, Factoring requirement dependencies in software requirement selection using graphs and integer programming. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering* (2016), 884–887. ACM.

[26] A.A. Araújo, M. Paixao, I. Yeltsin, A. Dantas and J. Souza, An architecture based on interactive optimization and machine learning applied to the next release problem, *Automated Software Engineering* **24**(3) (2017), 623-671.

[27] H. Jiang, J. Zhang, J. Xuan, Z. Ren and Y. Hu, A hybrid ACO algorithm for the next release problem. In *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on* (2010), pp. 166–171. IEEE. 171.

[28] R. Masadeh, A. Alzaqebah, A. Hudaib and A.A. Rahman, Grey wolf algorithm for requirements prioritization, *Modern Applied Science* **12**(2) (2018), 54.

[29] I.M. Del Águila and J. Del Sagrado, Three steps multi-objective decision process for software release planning, *Complexity* **21**(S1) (2016), 250-262.

[30] J.M. Chaves-González and M.A. Pérez-Toledano, Differential evolution with Pareto tournament for the multi-objective next release problem, *Applied Mathematics and Computation* **252** (2015), 1-13.

[31] A.M. Pitangueira, P. Tonella, A. Susi, R.S.P. Maciel and M. Barros, Minimizing the stakeholder dissatisfaction risk in requirement selection for next release planning, *Information and Software Technology* **87** (2017), 104-118.

[32] M. Nayebi and G. Ruhe, Asymmetric release planning-compromising satisfaction against dissatisfaction, *IEEE Transactions on Software Engineering* (2018), 1–5.

[33] A. Ebrahimnejad, A simplified new approach for solving fuzzy transportation problems with generalized trapezoidal fuzzy numbers, *Applied Soft Computing* **19** (2014), 171–176

[34] A. Ebrahimnejad, Z. Karimnejad and H. Alrezaamiri, Particle swarm optimisation algorithm for solving shortest path problems with mixed fuzzy arc weights, *International Journal of Applied Decision Sciences* **8**(2) (2015), 203-222.

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

[36] A. Ebrahimnejad and J.L. Verdegay, *Fuzzy Sets-Based Methods and Techniques for Modern Analytics*, volume 364 of Studies in Fuzziness and Soft Computing. (1st ed.). Springer International Publishing (2018).

[37] A. Ebrahimnejad, J.L. Verdegay and H. Garg, Signed distance ranking based approach for solving bounded interval valued fuzzy numbers linear programming problems. *International Journal of Intelligent Systems* **9**(34) (2019), 2055-2076.

[38] M. Enayattabar, A. Ebrahimnejad, H. Motameni and H. Garg, A novel approach for solving all-pairs shortest path problem in an interval-valued fuzzy network, *Journal of Intelligent & Fuzzy Systems* **37** (2019) 6865-6877.

[39] A. Abbaszadeh Sori, A. Ebrahimnejad and H. Motameni, The fuzzy inference approach to solve multi-objective constrained shortest path problem, *Journal of Intelligent & Fuzzy Systems* **32** (2020), 4711-4720.

[40] A. Abbaszadeh Sori, A. Ebrahimnejad and H. Motameni, Elite artificial bees' colony algorithm to solve robot's fuzzy constrained routing problem, *Computational Intelligence* **36** (2020), 659–681.

[41] A. Tajdin, I. Mahdavi, N. Mahdavi-Amiri and B. Sadeghpour-Gildeh, Computing a fuzzy shortest path in a network with mixed fuzzy arc lengths using α-cuts, *Computers & Mathematics with Applications* **60**(4) (2010), 989-1002.

[42] M.A. Tawhid and V. Savsani, A novel multi-objective optimization algorithm based on artificial algae for multi-objective engineering design problems, *Applied Intelligence* **48**(10) (2018), 3762-3781.