Alexandria Engineering Journal (2022) **61**, 3403–3415

HOSTED BY

ELSEVIER

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights

Debora Di Caprio [a], Ali Ebrahimnejad [b,*], Hamidreza Alrezaamiri [c], Francisco J. Santos-Arteaga [d,*]

[a] *Department of Economics and Management, University of Trento, Trento, Italy*
[b] *Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran*
[c] *Babol Branch, Islamic Azad University, Babol, Iran*
[d] *Faculty of Economics and Management, Free University of Bolzano, Bolzano, Italy*

Received 28 January 2021; revised 5 July 2021; accepted 22 August 2021
Available online 03 September 2021

KEYWORDS

Fuzzy shortest path problem;
Ant colony optimization;
Fuzzy numbers;
Fuzzy directed graph

Abstract The shortest path (SP) problem constitutes one of the most prominent topics in graph theory and has practical applications in many research areas such as transportation, network communications, emergency services, and fire stations services, to name just a few. In most real-world applications, the arc weights of the corresponding SP problems are represented by fuzzy numbers. The current paper presents a fuzzy-based Ant Colony Optimization (ACO) algorithm for solving shortest path problems with different types of fuzzy weights. The weights of the fuzzy paths involving different kinds of fuzzy arcs are approximated using the α-cut method. In addition, a signed distance function is used to compare the fuzzy weights of paths. The proposed algorithm is implemented on three increasingly complex numerical examples and the results obtained compared with those derived from a genetic algorithm (GA), a particle swarm optimization (PSO) algorithm and an artificial bee colony (ABC) algorithm. The results confirm that the fuzzy-based enhanced ACO algorithm could converge in about 50% less time than the alternative metaheuristic algorithms.

1. Introduction

The problem of finding the shortest path between two vertices is one of the most prominent research topics in graph theory [1]. A path in a graph is a sequence of vertices and edges (or arcs) with a specific origin and destination. When there are multiple paths between two specific vertices, finding a route with the lowest cost becomes a fundamental objective and, in turn, creates the problem of finding the shortest path [2–3].

* Corresponding authors at: Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran and Faculty of Economics and Management, Free University of Bolzano, Piazza Università 1, 39100 Bolzano, Italy.
E-mail addresses: debora.dicaprio@unitn.it (D. Di Caprio), a.ebrahimnejad@qaemiau.ac.ir, aemarzoun@gmail.com (A. Ebrahimnejad), hamidreza.alrezaa@qaemiau.ac.ir (H. Alrezaamiri), fsantosarteaga@unibz.it (F.J. Santos-Arteaga).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2021.08.058
1110-0168

In traditional shortest path problems, there is exact information about the parameters of the problem such as time, costs, and risk. However, real-world environments require dealing with uncertain parameter values. For example, parameters such as time or arc costs are usually affected by traffic or weather conditions. Therefore, it is not practical to assign them a certain value. In such cases, an imprecise type of problem arises where the arc parameters are not specified in advance. In these cases, a suitable modeling approach to deal with the uncertainty of parameters consists of using fuzzy numbers that convert the initial scenario into a fuzzy shortest path problem [4]. Moreover, there is generally a need to allocate different types of fuzzy numbers to these parameters.

Shortest path problems have been used to formalize and evaluate many real-life settings. For example, air pollution constitutes a fundamental environmental problem in metropolitan areas. Understanding the factors affecting air pollution can be effective in reducing its destructive effects and finding suitable solutions. Vehicles represent one of the main factors contributing to the pollution of big cities. Research has shown that traffic conditions are directly correlated with the amount of air pollution produced. That is, environmental pollution is not just caused by the number of vehicles. The amount of pollutants produced is larger in heavy traffic than in light traffic with a larger number of vehicles. It can be concluded that if the traffic flow were lighter in cities, air pollution and the waste of citizens' time would be significantly reduced [5].

In transportation systems such as those where taxi companies operate, the cost of the route varies according to the weather or time of the day [6]. In these systems, the use of fuzzy numbers to define the cost of the paths is much more reasonable than the assignment of crisp numbers. It is important for drivers to find the path that results in the lowest cost. At fire stations or emergency services, the time factor is crucial. When a station is contacted, relief vehicles must reach the destination site through the fastest route. The transit time of a specific route within a city varies through the different hours of the day. It is therefore natural to consider fuzzy numbers for the duration of the paths. Note how, in most situations, finding the optimal path is equivalent to identifying the least costly or fastest route.

The problem of finding the shortest path, in both its fuzzy and non-fuzzy forms, could indeed be considered an optimization problem. In fact, the optimal path is the shortest one. The complexity of the subsequent NP optimization problem has constrained traditional methods from finding the optimal solution within reasonable time. The use of metaheuristic algorithms to identify the shortest path provides a suitable solution technique aimed at improving the efficiency of the identification process.

Meta-heuristics have the capability to deal with additional constraints and deliver optimal or near optimal path solutions within a reasonable computational time both in small and large scale networks.

Meta-heuristics such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO) algorithms, and Ant Colony Optimization (ACO) algorithms have been widely used to approach shortest path problems in very different research fields. For instance, Kumar and Kumar [7] used GA to determine the shortest path in data networks. Rares [8] considered the shortest path routing problem for highly evolving networks, with a high load of traffic, and used an improved GA based on an adaptive mutation operator. Mohiuddin et al. [9] developed a fuzzy evolutionary PSO (FEPSO) algorithm to determine optimized routing paths and enhance the operational use of the network. Dudeja [10] proposed a fuzzy-based modified PSO algorithm as a strategy to overcome the shortest path problem with uncertain edges and reduce cost and time consumption. Gupta and Srivastava [11] solved the distance optimization problem using both PSO and ACO, and conducted a quantitative comparison between their performances with the simulated results showing the superiority of the latter optimization algorithm. Wang et al. [12] proposed an improved ACO algorithm for time-triggered flows in time-sensitive networks. Zangina et al. [13] used an improved non-dominated sorting genetic algorithm (INSGA-III) to design a robust vehicle routing problem scheme for an autonomous robot navigation strategy able to optimize both crop yield and quality subject to a minimum costs.

Regarding the incorporation of uncertain or imprecise data that are naturally involved in many network designs, the literature presents an increasing number of hybrid algorithms aiming at increasing the performance of a system, both locally and globally. Among the most recent studies, Dib et al. [14] propose a solution method where a genetic algorithm (GA) is coupled with a variable neighborhood search (VNS). Dib et al. [15] design an advanced GA-VNS heuristic method to deal with multicriteria shortest path problems in multimodal networks. Garg [16] presents a hybrid algorithm that combines GA with the gravitational search algorithm (GSA) to increase the performance of a system whose analysis, based on uncertain data, focuses on the most critical components for saving money, manpower, and time. Garg [17] presents a hybrid PSO-GA technique for solving constrained optimization problems. Patwal et al. [18] design an integrated heuristic approach that combines a time varying acceleration coefficient PSO algorithm with mutation strategies (TVAC-PSO-MS) to study the optimal power generation schedule for renewable energy sources. Garg [19] uses a hybrid GSA-GA algorithm for constrained nonlinear optimization problems with mixed variables. De Santis et al. [20] consider the problem of minimizing the travel distances of pickers in manual warehouses and develop a metaheuristic routing algorithm that integrates the ACO metaheuristic and the Floyd–Warshall algorithm. Finally, Sedighizadeh and Mazaheripour [21] present a hybrid algorithm based on a combination of PSO and an artificial bee colony (ABC) algorithm to solve the multi objective vehicle routing problem subject to Precedence constraints among customers.

In the current paper, we find the shortest (optimal) path using the navigation capacity of ants through the graph as determined by the rules of the ACO algorithm within a fuzzy setting. The fuzziness of single edges and paths is incorporated in an approximate manner, building on techniques introduced recently in the literature. The results obtained from running simulations of the proposed ACO algorithm on a small, medium and large size graphs have been compared with those obtained when applying GA [22], PSO [23], and ABC [24] algorithms. The criteria evaluated include convergence time, number of iterations needed to converge, and total running time. Each algorithm has been run 10 times on each of the analyzed graphs, obtaining the minimum, average, and maximum values for each one of the criteria considered. The numerical results show the superior performance of the proposed ACO algorithm in terms of number of iterations and convergence time.

The reminder of this paper is organized as follows. In the next section, we review the related literature and describe the main contribution of the paper. Section 3 introduces the main fuzzy concepts used through the paper. In Section 4, we describe the ACO algorithm proposed to solve the shortest path problem. Section 5 implements the enhanced optimization technique and analyzes the results comparing them to those obtained when applying GA, PSO, and the ABC algorithm. The last section concludes and suggests future research directions.

2. Literature review

In this section, we examine the problem of finding the shortest path in a directed graph. We start by reviewing several traditional and innovative methods designed to find the shortest path in directed and weighted graphs. Then, several methods that have been developed recently to find shortest paths within fuzzy environments will be analyzed.

2.1. Traditional algorithms

Dijkstra's algorithm is one of the main graph traversal algorithms. This algorithm solves the shortest path problem in weighted graphs that do not contain arcs with negative weights. Dijkstra's algorithm, with greedy policy, creates the shortest path tree of a graph specifying the shortest path from the source to all the vertices (or nodes) [25].

The Bellman-Ford algorithm is another graph traversal algorithm that solves the shortest path problem in weighted graphs where arc weights may be negative. Dijkstra's algorithm solves a similar problem requiring in less execution time but requires non-negative arc weights. In practice, the Bellman-Ford algorithm is only used for graphs displaying arcs with negative weights [26].

Floyd's algorithm is a graph analysis algorithm that finds the shortest path in a directed and weighted graph [27]. This algorithm constitutes an example of dynamic programming, comparing all the possible routes in a graph between every pair of vertices.

2.2. Applications to fuzzy environments

The Floyd-Warshall algorithm is implemented by [28] as a dynamic programming technique along with a ranking method to solve a shortest chain problem between all the vertices of a directed graph with fuzzy weights. Tajdin et al. [29] applied Floyd's algorithm and an approximation method for the addition of fuzzy edges to find the shortest path in a wireless sensor network with combined fuzzy edges. The scenario analyzed involved a mobile service provider company planning to deploy 23 centers in one area. Dou et al. [30] identified the shortest path in a multi-constrained network using a multi-criteria decision making technique based on a vague similarity measure. Deng et al. [31] extended the Dijkstra algorithm to solve the shortest path problem with fuzzy arc weights. Their method is based on the graded mean integration representation of fuzzy numbers.

2.3. Evolutionary fuzzy algorithms

Zhang et al. [32] proposed a biologically inspired algorithm called the fuzzy physarum algorithm for fuzzy shortest path problems based on a path finding model. Hassanzadeh et al. [22] used an evolutionary algorithm to find the shortest path in a directed graph with fuzzy weights. They were able to identify the optimal route by converting the shortest path problem into an optimization problem. Ebrahimnejad et al. [23] applied a PSO algorithm to reduce the run time and increase the speed of convergence relative to those obtained by GA proposed in [22]. Similarly, Ebrahimnejad et al. [24] applied an ABC algorithm to reduce the time required to find the shortest path in complex graphs with fuzzy arcs. Their algorithm used the mutation operator to carry out the process of searching for employed and onlooker bees.

2.4. Ant colony optimization

Calle et al. [33] extended the ACO algorithm by endowing ants with the sense of smell. Even though it may not identify the shortest path, their algorithm quickly finds a route between two nodes through a dynamic graph, providing a useful response in problems where optimality is not required. Ashour et al. [34] presented a novel algorithm to solve the Traveling Salesman Problem. These authors grouped vertices into clusters using Adaptive Affinity Propagation and then found the optimal route for each cluster separately through ACO. Changdar et al. [35] designed a genetic-ACO algorithm to solve solid multiple travelling salesman problems in a fuzzy environment. Each salesman selects his path using an enhanced version of ACO and the paths of different salesmen are controlled by GA.

2.5. Contribution

In the current paper, the ACO algorithm has been generalized to solve fuzzy shortest path problems with the lowest cost and in less time than any other standard metaheuristic algorithm. These latter include GA as well as the PSO and ABC algorithms. We find the shortest (optimal) path using the navigation capacity of ants through the graph as determined by the rules of the ACO algorithm within a fuzzy setting. The weights of the fuzzy paths involving different kinds of fuzzy arcs are approximated using the α-cut method, while a signed distance function is defined to compare the fuzzy weights of paths.

The proposed algorithm is applied to graphs of varying complexity and the results obtained compared with those derived from the other metaheuristic algorithms, namely, GA [22], PSO [23], and ABC [24] algorithms. The criteria evaluated include convergence time, number of iterations needed for each algorithm to converge, and total running time. Each algorithm has been run 10 times on each of the analyzed graphs, obtaining the minimum, average, and maximum values for each one of the criteria considered. The capacity of proposed fuzzy-based ACO to converge in 50% less time than the other algorithms should be particularly highlighted.

The arc weights are formalized through trapezoidal and normal fuzzy numbers. There are two reasons behind validating this choice. From a technical viewpoint, both trapezoidal and normal fuzzy weights allow for computing α-cuts. Thus, mixed sums of trapezoidal and normal fuzzy weights can be approximated using α-cuts [29]. From a practical viewpoint, the combined use of trapezoidal and normal fuzzy numbers opens the way to applications to a wide range of real-life shortest path problems. Indeed, trapezoidal fuzzy numbers include triangular fuzzy numbers as a special case and are the operational numbers to which both LR fuzzy numbers and Type-2 fuzzy sets/numbers are usually reduced when formalizing real-life situations. At the same time, normal fuzzy numbers can be regarded as the fuzzy counterpart of one of the most common probabilistic approaches to the analysis of the performance of a system.

In the simulations, the single arcs of the initial small size graph have been assigned either a triangular or a normal fuzzy weight, while the arcs of the medium and large size graphs have been assigned either a trapezoidal or a normal fuzzy weight. The type of fuzzy weight of the single arc has been chosen so as to guarantee variety through the graphs and foresee possible real applications.

3. Basic fuzzy concepts

In this section, we present some basic definitions and arithmetic operations on fuzzy numbers [28,36].

Definition 1:. *The trapezoidal fuzzy number $\tilde{a}$ is denoted by $\tilde{a} = (a_1, a_2, a_3, a_4)$ and its membership function is defined as follows*:

$$\mu_{a^{\sim}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1} & a_1 < x \leqslant a_2 \\ 1 & a_2 \leqslant x \leqslant a_3 \\ \frac{a_4-x}{a_4-a_3} & a_3 \leqslant x < a_4 \end{cases} \tag{1}$$

An example of a trapezoidal fuzzy number is shown in Fig. 1.

Triangular fuzzy numbers are a special type of trapezoidal number where the values of the two median trapezoidal parameters are equal. This number is denoted by $\tilde{a} = (a_1, a_2, a_3)$, and its membership function is defined as follows.

$$\mu_{a^{\sim}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1} & a_1 < x \leqslant a_2 \\ \frac{a_3-x}{a_3-a_2} & a_2 \leqslant x < a_3 \end{cases} \tag{2}$$

Fig. 2 provides an example of this type of number.

Definition 2:. *The membership function of a normal fuzzy number $\tilde{a} = (m, \sigma)$ is defined as follows*:

$$\mu_{a^{\sim}}(x) = e^{-\left(\frac{x-m}{\sigma}\right)^2} \quad , \quad x \in \Re \tag{3}$$

Fig. 3 illustrates the membership function of a normal fuzzy number.

Definition 3:. *Given a fuzzy set $\tilde{A}$ defined on the universal set of real numbers Xand any number $0 < \alpha < 1$, the α-cut of is defined as $\tilde{A}_\alpha = \{ x \mid \mu_{\tilde{A}}(x) \geqslant \alpha , x \in X\}$.*

Remark 1:. *The α-cut of a trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ is given by the real interval $[\tilde{A}]_\alpha = [\tilde{A}^-_\alpha, \tilde{A}^+_\alpha] = [(a_2 - a_1)\alpha + a_1, a_4 - (a_4 - a_3)\alpha]$.*

Remark 2:. *The α-cut of a normal fuzzy number $\tilde{A} = (m, \sigma)$ is given by the real interval $[\tilde{A}]_\alpha = [\tilde{A}^-_\alpha, \tilde{A}^+_\alpha] = [m - \sigma\sqrt{-\ln(\alpha)}, m + \sigma\sqrt{-\ln(\alpha)}]$.*

A standard procedure for approximating the sum of a trapezoidal fuzzy number $\tilde{A} = (a_1, a_2, a_3, a_4)$ and a normal fuzzy number $\tilde{B} = (m, \sigma)$ is as follows. The sum and its corresponding membership function are approximated by dividing the α–interval, [0, 1], into n subintervals and letting $\alpha_0 = 0$, $\alpha_i = \alpha_{i-1} + \Delta\alpha_i$ where $\Delta\alpha_i = \frac{1}{n}$ and $i = 1, 2, \ldots, n$.

Given $\alpha_i \in (0, 1]$, $1 \leqslant i \leqslant n$, the α_i–cut sum of these fuzzy numbers is derived using Remarks 1 and 2 as follows:

$$\begin{gathered} [\tilde{C}]_{\alpha_i} = \left[\tilde{C}^L_{\alpha_i}, \tilde{C}^R_{\alpha_i}\right] = \left[\tilde{A}^L_{\alpha_i} + \tilde{B}^L_{\alpha_i}, \tilde{A}^R_{\alpha_i} + \tilde{B}^R_{\alpha_i}\right] = \\ \left[(a_2 - a_1)\alpha_i + a_1 + m - \sigma\sqrt{-\ln(\alpha_i)},\ a_4 - (a_4 - a_3)\alpha_i + m + \sigma\sqrt{-\ln(\alpha_i)}\right] \end{gathered} \tag{4}$$

Eq. (4) is used to obtain n points for $\tilde{C}^L_{\alpha_i}$ and n points for $\tilde{C}^R_{\alpha_i}$ using α_i, $1 \leqslant i \leqslant n$. A similar intuition applies when deriving the α_i–cut sum of triangular and normal fuzzy numbers.

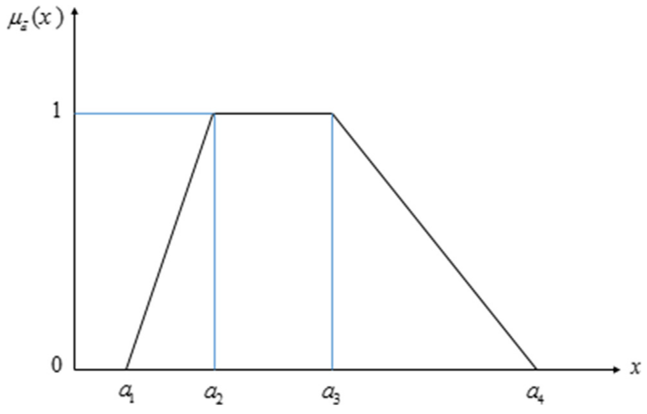

Fig. 1 A trapezoidal fuzzy number.

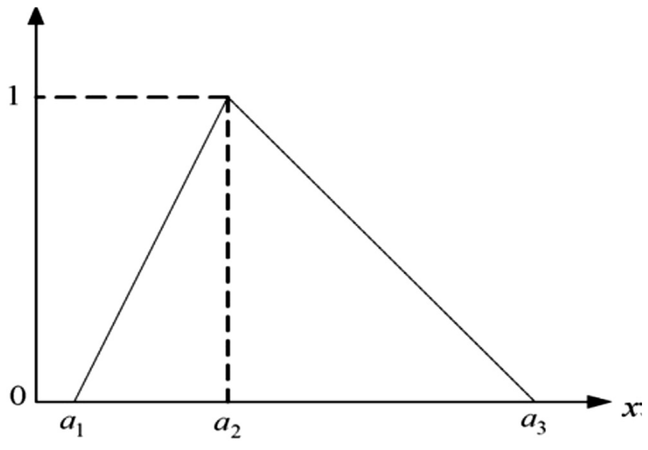

Fig. 2 A triangular fuzzy number.

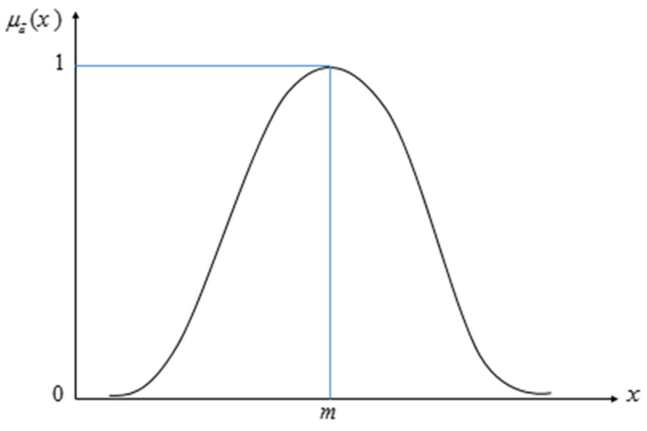

Fig. 3 A normal fuzzy number.

Hassanzadeh et al. [22] approximated the membership function of the sum using the points obtained via the α–cut and Crammer's approach to fit an exponential membership function for the whole sum.

Let $x_i = \overset{\sim R}{C}_{\alpha_i}$ and $y_i = \mu(\overset{\sim R}{C}_{\alpha_i})$, and for n points (x_i, y_i), consider the fitting model to be $y = e^{-\left(\frac{x-\lambda}{\beta}\right)^2}$. These authors proposed a least squares model to approximate the right membership function for the α–cut-based addition, and determined the unknown parameters λ and β as follows [27,31]:

$$\beta = \frac{n\sum_i\left(x_i \times \sqrt{-\ln y_i}\right) - \sum_i\sqrt{-\ln y_i} \times \sum_i x_i}{-n\sum_i\sqrt{-\ln y_i} - \sum_i\sqrt{-\ln y_i} \times \sum_i\sqrt{-\ln y_i}} \tag{5}$$

$$\lambda = \frac{\sum_i \ln y_i\left(-\sum_i x_i\right) - \sum_i\left(x_i \times \sqrt{-\ln y_i}\right) \times \sum_i\sqrt{-\ln y_i}}{-n\sum_i\sqrt{-\ln y_i} - \sum_i\sqrt{-\ln y_i} \times \sum_i\sqrt{-\ln y_i}} \tag{6}$$

Similarly, let $x_i = \overset{\sim L}{C}_{\alpha_i}$ and $y_i = \mu(\overset{\sim L}{C}_{\alpha_i})$, 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 α–cut-based addition results in the unknown parameters λ' and β' being defined as follows [22,29]:

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

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

Therefore, the approximate membership function for the approximating sum of trapezoidal and normal fuzzy numbers is given by:

$$\mu_{c^\sim}(x) = \begin{cases} e^{-\left(\frac{\lambda'-x}{\beta'}\right)^2}, & x < \lambda', \\ 1, & \lambda' \leqslant x \leqslant \lambda, \\ e^{-\left(\frac{x-\lambda}{\beta}\right)^2}, & x > \lambda. \end{cases} \tag{9}$$

In what follows, we define the distance between two fuzzy numbers using the resulting points from the α–cut process [27,31]. In particular, the distance between two fuzzy numbers $\tilde{A}$ and $\tilde{B}$, $D_{p,q}$, is defined as follows:

$$D_{p,q}(\tilde{A}, \tilde{B}) = \begin{cases} \left[(1-q)\int_0^1 \left|A_\alpha^- - B_\alpha^-\right|^p d\alpha + q\int_0^1 \left|A_\alpha^+ - B_\alpha^+\right|^p d\alpha\right], & p < \infty \\ (1-q)\sup\limits_{0<\alpha\leqslant 1}\left|A_\alpha^- - B_\alpha^-\right| + q\inf\limits_{0<\alpha\leqslant 1}\left|A_\alpha^+ - B_\alpha^+\right|, & p = \infty \end{cases} \tag{10}$$

where the first parameter p denotes the priority weight assigned to the end points of the support (for instance, the A_α^- and A_α^+ components of the fuzzy numbers). The second parameter q determines the analytical properties of $D_{p,q}$. If the expert has no preference, $D_{p,\frac{1}{2}}$ is used. Given two fuzzy numbers $\tilde{A}$and $\tilde{B}$, $D_{p,q}$ is proportional to:

$$D_{p,q}(\tilde{A}, \tilde{B}) = \left[(1-q)\sum_{i=1}^{n}\left|A_{\alpha_i}^- - B_{\alpha_i}^-\right|^p + q\sum_{i=1}^{n}\left|A_{\alpha_i}^+ - B_{\alpha_i}^+\right|^p\right]^{\frac{1}{p}} \tag{11}$$

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

$$D_{2,\frac{1}{2}}(\tilde{A}, \tilde{B}) = \sqrt{\left[\frac{1}{2}\sum_{i=1}^{n}\left|A_{\alpha_i}^- - B_{\alpha_i}^-\right|^2 + \frac{1}{2}\sum_{i=1}^{n}\left|A_{\alpha_i}^+ - B_{\alpha_i}^+\right|^2\right]} \tag{12}$$

We will compare two fuzzy arc weights $\tilde{A}$ and $\tilde{B}$ to $\tilde{0} = (0, 0, \ldots, 0)$ using the α_i-cuts. This reference benchmark has been chosen because both weights are supposed to represent positive values. In fact, Eq. (10) is used to compute $D_{2,\frac{1}{2}}(\tilde{A}, \tilde{0})$ and $D_{2,\frac{1}{2}}(\tilde{B}, \tilde{0})$. In this case, we can conclude that $\tilde{A} \preceq \tilde{B}$ if and only if $D_{2,\frac{1}{2}}(\tilde{A}, \tilde{0}) \leqslant D_{2,\frac{1}{2}}(\tilde{B}, \tilde{0})$.

4. Proposed ant colony optimization algorithm

Ants and some species of bees live in large groups called colonies. Insect swarms can work together to resolve problems that none of the members of the swarm could solve on their own. The ACO algorithm is an excellent example of swarm intelligence [37]. Fig. 4 shows an ant colony.

The ACO algorithm consists of a given number of ants. The main objective of the ants is to find the shortest path between

Fig. 4 Ant colony.

the nest and a food source. Suppose the ant path is defined within a graph $G = (V, E)$. Using this algorithm, one can find the shortest existing path between two arbitrary vertices of graph G. Any edge that connects vertex i to vertex j is denoted by l_{ij}. An amount of pheromone τ_{ij} is assigned to each edge l_{ij}. The amount of pheromone must be read or modified by the ants. In addition, the amount of pheromone existing on an edge is used as a measure assessing the desirability of the edge. It is also intended to be selected by ants to build better pathways. At the beginning of an algorithm, all edges have an equal amount of pheromone amounting to τ_0 [38].

In the ACO algorithm, ants add an amount of pheromone to an edge when passing through it. This amount is defined as $\Delta\tau_{ij}$. That is, if an ant passes through an edge located between vertices i and j at time t, it modifies the amount of pheromone associated to the edge as follows:

$$\tau_{ij}(t) \quad \leftarrow \quad \tau_{ij}(t) + \Delta\tau_{ij} \tag{13}$$

In order to conduct an adequate search between different paths, pheromones existing on graph edges evaporate, just as real pheromones do. The amount of pheromone on the edges decreases automatically, in such a way that ants become more interested in searching for new and probably better paths. The evaporation process is usually defined as a decreasing function. The overall form of this formula is defined as follows:

$$\tau_{ij}(t) \quad \leftarrow \quad (1-\rho)\tau_{ij}(t) \quad , \quad \rho \in (0,1] \tag{14}$$

where ρ is a parameter known as the coefficient of evaporation. Relatively low values of the evaporation coefficient ρ result in fast convergence, leading the ants to lose interest in exploring the edges of the graph and rely on their initial responses. In contrast, assuming a relatively high value of ρ leads to a very slow convergence process, while, in many cases, the algorithm will not converge. That is, the coefficient of evaporation directly affects the convergence time of the algorithm [39].

In some problems, initial parameters are defined as quantities associated with the edges. These quantities, which are called subjective values, are denoted as η_{ij} for edge l_{ij}. Regarding the problem of finding the shortest path, subjective values are defined as $\eta_{ij} = 1d_{ij}$, where d_{ij} is the distance between vertex i and vertex j.

Ants existing in colonies have the following properties:

- Ants began to move from a source vertex. At each step, one of the adjacent nodes is selected on the basis of a probability formula. If ant k is in vertex i, the probability that vertex j will be selected as the next destination is determined based on the following relationship

$$P_{ij}^k = \begin{cases} \frac{(\tau_{ij})^\alpha (\eta_{ij})^\alpha}{\sum_{m \in N_i^k} (\tau_{im})^\alpha (\eta_{im})^\alpha} & j \in N_i^k \\ 0 & j \notin N_i^k \end{cases} \tag{15}$$

where N_i^k denotes vertices that are in the neighborhood of vertex i.

- In equation (19), α and β are constant positive numbers used for weighting the pheromone and subjective information, respectively. The higher the weight of a certain kind of information, the more effective the ants' decision-making, which affects their answers through the path selection process. In this case, assuming $\alpha = 0$, the probability of selection of closer vertices is higher. In fact, in this case, the ACO algorithm turns into a random, and of course greedy, search algorithm. In contrast, assuming $\beta = 0$, only the pheromone information will be used. In this case, there will be a more rapid convergence, causing stagnation in the algorithm procedure.
- Ants continue moving until at least one of the termination conditions is met. This creates a feasible path for the problem.
- While moving from vertex i toward vertex j, each ant modifies the amount of pheromone τ_{ij} on edge l_{ij}. This procedure is called step-by-step pheromone renewal. Over time, the amount of pheromones available on paths decreases with evaporation.
- Each ant has a personal memory denoted by M that stores each path along which it passes. This memory can be used to create feasible solutions, evaluate the solutions found, and move along the path in reverse.

The flowchart of the ACO algorithm designed to solve the problem of finding the shortest path is illustrated in Fig. 5.

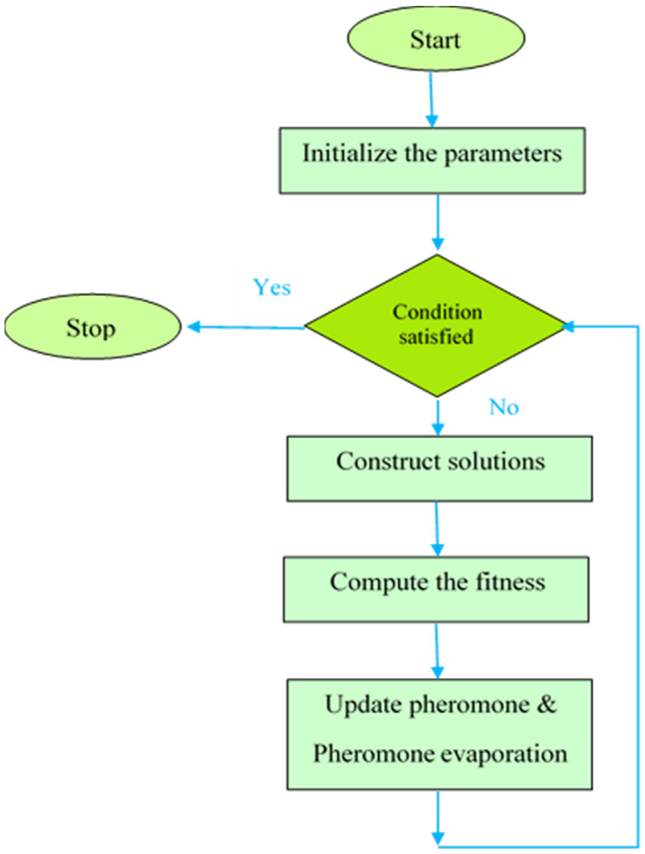

Fig. 5 Flowchart describing the implementation of the ACO algorithm.

5. Implementation

In this section, we implement and analyze the ACO algorithm on three directed and weighted graphs with varying complexities. MATLAB R2010b run on a computer with a Windows 7 operating system, an Intel core i7 1.6 GHz processor, and a 4 GB RAM was used for its implementation.

Fig. 6 describes the first graph on which the algorithm was run, encompassing 11 vertices and 25 edges. The source vertex of the graph is vertex 1, and the destination vertex is number 11. The weight of the fuzzy edges is described in Table 1. This graph has edges of triangular and normal types. The shortest path in the graph is given by $1 \rightarrow 3 \rightarrow 8 \rightarrow 7 \rightarrow 11$ with a fitness value of 1129.48. As explained in Section 3, the solution process requires adding the fuzzy edges of a path, which provides an approximate fuzzy number. Then, the fitness value of this fuzzy number is compared to others using the respective crisp distances, $D_{p,q}$, defined in Section 3.

The results from running simulations of the ACO algorithm on this graph have been compared with the results obtained when applying GA [22], PSO [23], and ABC [24] algorithms. In order to allow for a fair comparison of the results across algorithms, absolutely identical conditions were considered for their implementation. For instance, a total of 30 iterations have been simulated for each algorithm. We have also assumed

- a colony composed by a total of 10 ants;
- a population consisting of 10 chromosomes in GA;
- 10 swarm particles in the PSO algorithm;
- 10 clone solutions in the bee colony algorithm.

The remaining parameters used in the ant colony simulation are described in Table 2.

The criteria evaluated were the convergence time, the number of iterations needed for each algorithm to converge, and the total running time of each algorithm. Table 3 presents the results from the implementation of the different algorithms on the graph illustrated in Fig. 6. Each algorithm was run 10 times, and the minimum, average, and maximum values obtained for each criterion are provided in Table 3.

This table illustrates how all the algorithms were able to detect the shortest path of the graph i.e. $1 \rightarrow 3 \rightarrow 8 \rightarrow 7 \rightarrow 11$. Regarding the number of iterations needed for convergence, the ACO algorithm converged on average in the third iteration. The convergence rate obtained was lower than that of the other three algorithms. Regarding the run time convergence criterion, the ACO algorithm required the least time to converge on average. Compared to the performance of the ABC algorithm, the difference in average convergence times is insignificant. Yet, this difference is relatively large for the PSO algorithm (48% less) and substantially larger for GA (80% less). Regarding the overall run time criterion, the ACO algorithm had the fastest run. Its runs ended almost 50% faster than those of the bee colony and PSO algorithms.

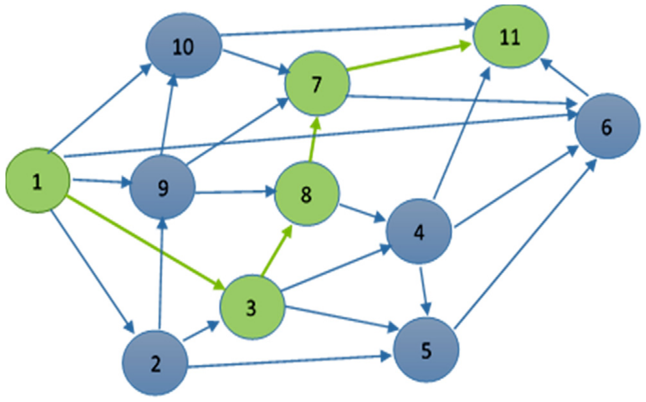

Fig. 6 Initial graph setting.

These results are illustrated in Fig. 7, which describes the average convergence time and average runtime of all the algorithms, and Fig. 8, which presents a convergence graph of the simulated algorithms. Note that the fastest convergence pattern is that of the ACO algorithm, which is described by a purple dotted curve.

The relative performance of the proposed ACO algorithm will be examined through increasingly complex evaluation structures. We will illustrate how, as the complexity of the graphs increases, the performance of the ACO algorithm improves relative to that of the alternative metaheuristic algorithms. Fig. 9 presents a graph with 23 vertices and 40 edges. The source vertex of the graph is denoted by 1, while the destination is vertex 23. The shortest path is given by $1 \rightarrow 5 \rightarrow 12 \rightarrow 15 \rightarrow 18 \rightarrow 23$, with a fitness value of 184.24. Table 4 describes the weight of the fuzzy edges. Note that the current graph could be used to describe a real instance of a fire station or a transportation system requiring a vehicle to reach a given destination incurring the lowest possible cost.

Table 5 illustrates the results obtained from 10 runs of the different algorithms. As in the previous example, a total of 30 iterations have been simulated per algorithm. Given the complexity of the current graph, we have increased the operational capacity of the different algorithms by assuming

- A colony composed by a total of 22 ants;
- A population consisting of 22 chromosomes in GA;
- 22 swarm particles in the PSO algorithm;
- 22 clone solutions in the ABC algorithm.

The ACO algorithm converged on average to the best path after 2.5 iterations. In this regard, it performed better than the ABC and PSO algorithms, and substantially better than GA. When considering the run time convergence criterion, the ACO algorithm converged much faster than the others on average. This means that the algorithm finds the optimal path in complex graphs much faster than any of the other algorithms. Regarding average convergence time, the ACO algorithm reached convergence 35% faster than the ABC algorithm, 55% faster than the PSO algorithm, and 71% faster than GA.

These results are illustrated in Fig. 10, which describes the average convergence time and average runtime of all the algorithms. Fig. 11 presents a convergence graph of the simulated algorithms, with the ACO algorithm displaying the fastest convergence pattern.

Finally, we analyze the behavior of the different algorithms when dealing with a large network structure. Fig. 12 describes the third graph on which the algorithms were run, encompassing 30 vertices and 71 edges. The source vertex of the graph is

Table 1 Fuzzy weights of the edges in the initial graph setting.

Edges	Fuzzy number	Edges	Fuzzy number	Edges	Fuzzy number
(1, 2)	(800, 820, 840)	(3, 5)	(730, 748, 870)	(8, 4)	(710, 730, 835)
(1, 3)	(35, 11)	(3, 8)	(42, 14)	(8, 7)	(230, 242, 355)
(1, 6)	(650, 677, 783)	(4, 5)	(190, 199, 310)	(9, 7)	(120, 130, 250)
(1, 9)	(290, 300, 350)	(4, 6)	(310, 340, 460)	(9, 8)	(13, 4)
(1, 10)	(420, 450, 570)	(4, 11)	(71, 23)	(9, 10)	(23, 7)
(2, 3)	(180, 186, 293)	(5, 6)	(610, 660, 790)	(10, 7)	(330, 342, 450)
(2, 5)	(495, 510, 625)	(6, 11)	(23, 7)	(10, 11)	(125, 41)
(2, 9)	(90, 30)	(7, 6)	(390, 410, 540)	(3, 4)	(650, 667, 983)
(7, 11)	(45, 15)				

Table 2 Simulation parameters.

Parameter	Value
Number of Iterations	30
Number of Ants	10
α	1
β	1
τ	0.1
ρ	Rand (0,1]
Δ	0.15
P_c	0.40
P_m	0.10
C_1	2.0
C_2	2.0
W_{max}	0.90
W_{min}	0.20
Limit	No. Iterations/6

Legend: P_C and P_m: GA crossover and mutation rates, respectively; C_1, C_2, W_{max} and W_{min}: PSO particles' update parameters; Limit: number of employed bees converted into scouts after an unsuccessful search within the ABC.

vertex 1, and the destination vertex is number 30. The length of the fuzzy edges is described in Table 6. The shortest path has been highlighted in green and is given by the following vertex sequence

$1 \rightarrow 2 \rightarrow 11 \rightarrow 13 \rightarrow 23 \rightarrow 25 \rightarrow 28 \rightarrow 30$,

which delivers a fitness value of 63.72.

Table 7 presents the results obtained from 10 runs of the different algorithms. Due to the complexity of this graph, we have increased the number of iterations and the size of the population. In order to perform a fair comparison among algorithms, the number of iterations in all of them has been set equal to 60. In addition, the population size of each algorithm has been set equal to these values

- A colony composed by a total of 40 ants;
- A population consisting of 40 chromosomes in GA;
- 40 swarm particles in the PSO algorithm;
- 40 clone solutions in the ABC algorithm.

The rest of the simulation parameters correspond to the values described in Table 2.

As illustrated in Table 7, the ACO algorithm displays the best convergence rate. Given its capacity to explore potential paths, the ACO algorithm converged on average after 8.9 iter-

Table 3 Implementation results in the initial graph setting.

	Iteration	SP	Number of iterations to converge				Convergence time span (s)				Total time (s)			
			GA	PSO	ABC	ACO	GA	PSO	ABC	ACO	GA	PSO	ABC	ACO
1	30	1–3-8–7-11	5	4	4	3	0.55	0.28	0.21	0.14	2.54	1.69	1.57	0.87
2	30	1–3-8–7-11	9	2	4	1	0.99	0.17	0.22	0.04	3.02	1.77	1.68	0.85
3	30	1–3-8–7-11	3	5	2	5	0.28	0.33	0.09	0.25	2.81	1.67	1.61	0.86
4	30	1–3-8–7-11	15	4	2	4	1.55	0.25	0.09	0.25	2.64	1.68	1.59	0.91
5	30	1–3-8–7-11	6	8	2	3	0.59	0.54	0.08	0.13	3.06	1.77	1.56	0.85
6	30	1–3-8–7-11	2	1	3	6	0.26	0.09	0.17	0.28	3.00	1.68	1.66	0.88
7	30	1–3-8–7-11	1	2	4	1	0.18	0.18	0.19	0.05	3.01	1.67	1.56	0.90
8	30	1–3-8–7-11	9	6	1	2	0.93	0.37	0.06	0.07	2.94	1.75	1.88	0.80
9	30	1–3-8–7-11	10	7	3	4	1.09	0.48	0.33	0.19	3.07	1.63	1.75	0.85
10	30	1–3-8–7-11	11	3	7	1	1.11	0.22	0.19	0.05	3.02	1.58	1.69	0.86
Min	–	–	1	1	1	1	0.18	0.09	0.06	0.04	2.54	1.58	1.56	0.80
Max	–	–	15	8	7	6	1.55	0.54	0.33	0.28	3.07	1.77	1.88	0.91
Avg	–	–	7.10	4.20	3.20	3.0	0.75	0.29	0.16	0.15	2.91	1.69	1.66	0.86

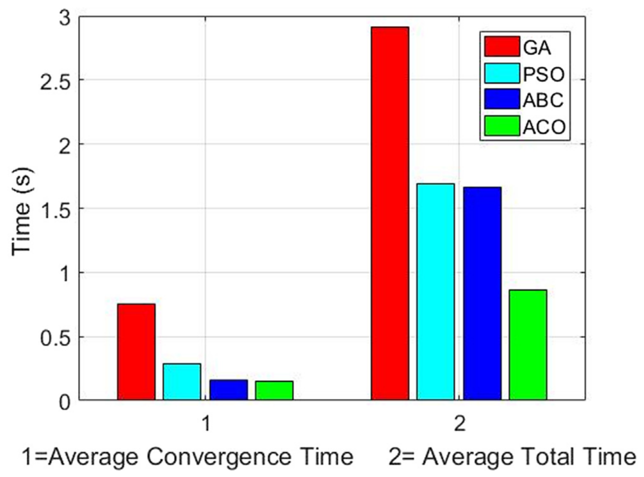

Fig. 7 Average convergence time and average runtime of the algorithms implemented within the initial graph setting.

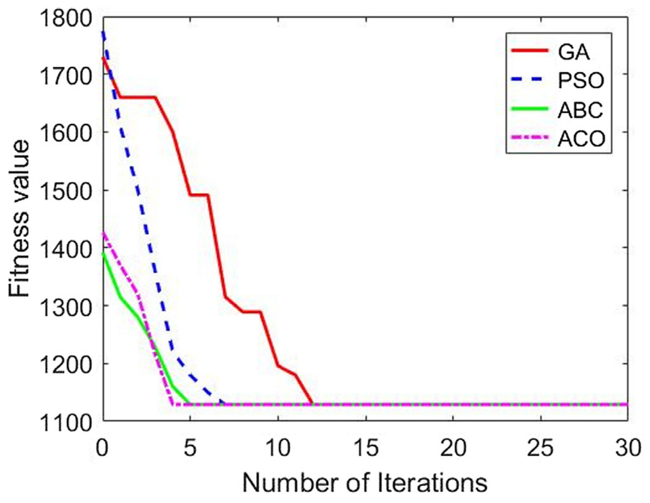

Fig. 8 Convergence of the algorithms implemented within the initial graph setting.

ations, a much lower number than the ones displayed by the other metaheuristic algorithms. We should highlight that the PSO algorithm was unable to find the shortest path in three of the runs, a fact shown in the table with a dash.

In terms of convergence time, the ACO algorithm was 74% faster than the GA, 65% faster than the ABC algorithm and 40% faster than the PSO algorithm. The mean convergence time of the PSO was obtained from the average of the seven - runs in which the algorithm converged. The total run time of the algorithms is also shown in Table 7. As in the previous cases, the ACO algorithm displays the shortest run time due to its agility. Fig. 13 compares the convergence and run times of the algorithms using a bar graph, while Fig. 14 presents a convergence graph illustrating the fastest convergence pattern exhibited by the ACO algorithm.

We conclude the analysis by summarizing the main convergence results of the algorithms within the different evaluation scenarios. Table 8 highlights the dominance of ACO relative to the alternative metaheuristic algorithms in terms of average iterations and convergence times. This dominance becomes particularly evident as the complexity of the evaluation scenarios increases, as can be observed when comparing the behavior of ACO with that of ABC, its closest rival in terms of performance.

6. Conclusion

We have applied the ACO algorithm to find the shortest path within a graph whose edges consist of normal, triangular, and trapezoidal fuzzy numbers. Three graphs of varying complexity have been used to illustrate the capacity of this metaheuristic algorithm to solve complex shortest path problems. The results obtained were compared with those derived from the implementation of genetic, PSO, and ABC algorithms. We have illustrated numerically the superior performance of the ACO algorithm in terms of number of iterations and convergence time. This feature is particularly relevant when the complexity of the graph increases, in which case the convergence behavior of the ACO algorithm clearly outperforms that of the other metaheuristic algorithms. In particular, the ACO

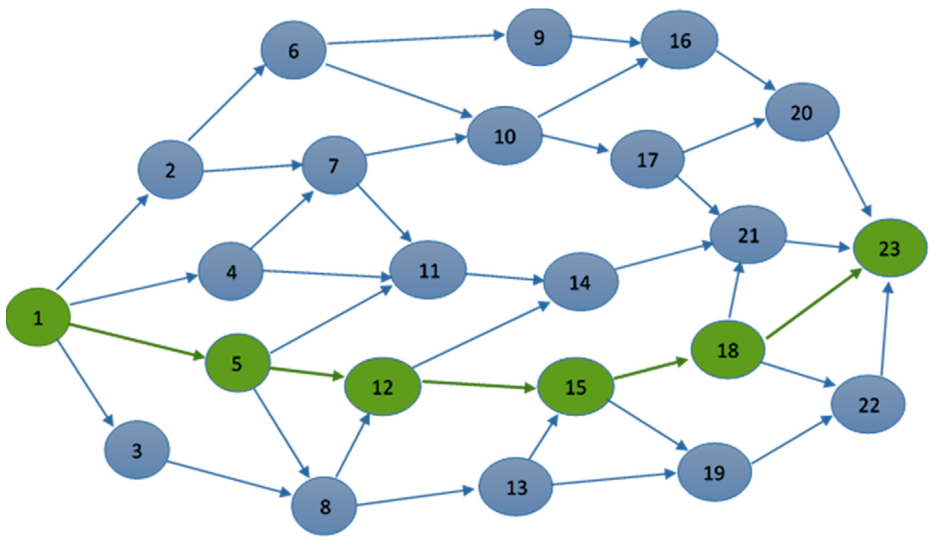

Fig. 9 Enhanced graph setting.

Table 4 Fuzzy weights of the edges in the enhanced graph setting.

Edges	Fuzzy number	Edges	Fuzzy number	Edges	Fuzzy number
(1, 2)	(12, 13, 15, 17)	(1, 3)	(40, 11)	(1, 4)	(8, 10, 12, 13)
(1, 5)	(7, 8, 9, 10)	(2, 6)	(35, 10)	(2, 7)	(6, 11, 11, 13)
(3, 8)	(40, 11)	(4, 7)	(17, 20, 22, 24)	(4, 11)	(6, 10, 13, 14)
(5, 8)	(29, 9)	(5, 11)	(7, 10, 13, 14)	(5, 12)	(10, 13, 15, 17)
(6, 9)	(6, 8, 10, 11)	(6, 10)	(35, 11)	(7, 10)	(9, 10, 12, 13)
(7, 11)	(6, 7, 8, 9)	(8, 12)	(5, 8, 9, 10)	(8, 13)	(50, 5)
(9, 16)	(6, 7, 9, 10)	(10, 16)	(40, 13)	(10, 17)	(15, 19, 20, 21)
(11, 14)	(8, 9, 11, 13)	(11, 17)	(28, 9)	(12, 14)	(13, 14, 16, 18)
(12, 15)	(12, 14, 15, 16)	(13, 15)	(37, 12)	(13, 19)	(17, 18, 19, 20)
(14, 21)	(12, 12, 13, 14)	(15, 18)	(8, 9, 11, 13)	(15, 19)	(25, 7)
(16, 20)	(38, 12)	(17, 20)	(7, 10, 11, 12)	(17, 21)	(6, 7, 8, 10)
(18, 21)	(15, 17, 18, 19)	(18, 22)	(16, 5)	(18, 23)	(15, 5)
(19, 22)	(5, 16, 17, 19)	(20, 23)	(13, 14, 16, 17)	(21, 23)	(12, 15, 17, 18)
(22, 23)	(20, 5)				

Table 5 Implementation results in the enhanced graph setting.

	Iteration	SP	Number of iteration to converge				Convergence time span (s)				Total time (s)			
			GA	PSO	ABC	ACO	GA	PSO	ABC	ACO	GA	PSO	ABC	ACO
1	30	1–5-12–15-18–23	5	4	2	3	1.76	1.80	0.69	0.96	9.19	7.44	5.33	3.49
2	30	1–5-12–15-18–23	3	2	3	2	1.61	1.00	0.98	0.49	9.63	7.33	5.26	3.48
3	30	1–5-12–15-18–23	8	3	1	4	2.15	1.41	0.46	0.91	9.88	7.41	5.54	3.41
4	30	1–5-12–15-18–23	4	1	5	1	1.68	0.51	1.97	0.34	9.55	7.29	5.41	3.63
5	30	1–5-12–15-18–23	2	5	3	1	1.35	1.91	1.03	0.25	9.42	7.65	5.28	3.55
6	30	1–5-12–15-18–23	10	1	1	6	2.88	0.55	0.47	1.43	9.81	7.31	5.21	3.43
7	30	1–5-12–15-18–23	17	8	2	3	4.87	2.12	0.63	0.64	9.34	7.55	5.61	3.36
8	30	1–5-12–15-18–23	12	4	1	1	3.13	1.78	0.41	0.24	9.30	7.34	5.42	3.48
9	30	1–5-12–15-18–23	9	6	6	3	2.64	1.97	2.14	0.71	9.35	7.11	5.51	3.46
10	30	1–5-12–15-18–23	6	1	2	1	2.11	0.52	0.68	0.25	9.51	7.47	5.31	3.54
Min	–	–	2	1	1	1	1.09	0.51	0.41	0.24	9.19	7.11	5.21	3.36
Max	–	–	17	8	6	6	4.62	2.12	2.14	1.43	9.88	7.65	5.61	3.63
Avg	–	–	7.6	3.5	2.6	2.5	2.13	1.36	0.95	0.62	9.50	7.39	5.39	3.48

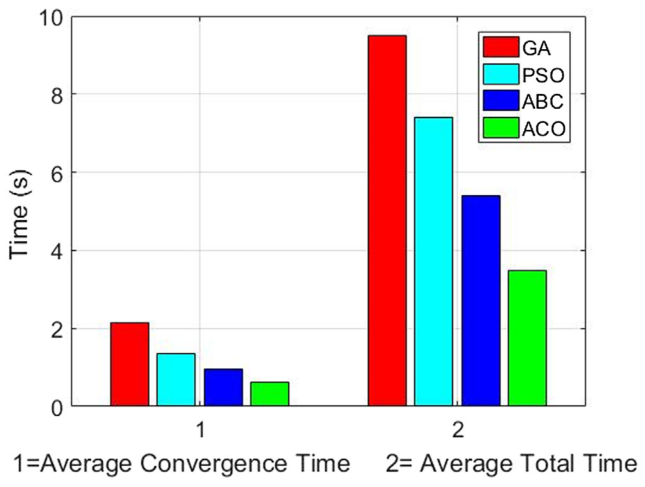

Fig. 10 Average convergence time and average runtime of the algorithms implemented within the enhanced graph setting.

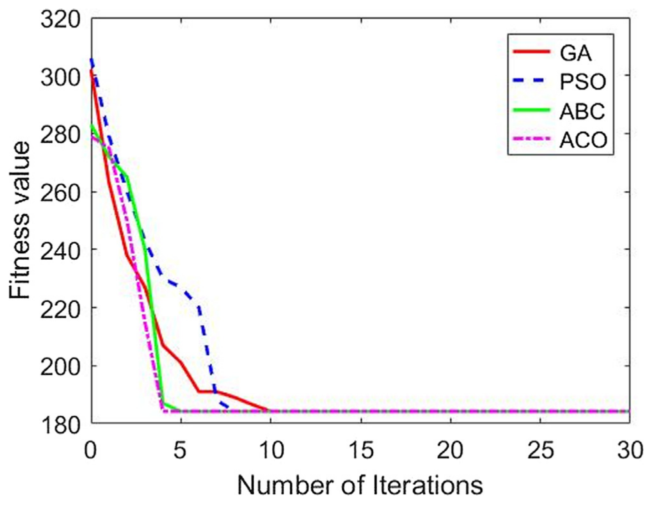

Fig. 11 Convergence of the algorithms implemented within the enhanced graph setting.

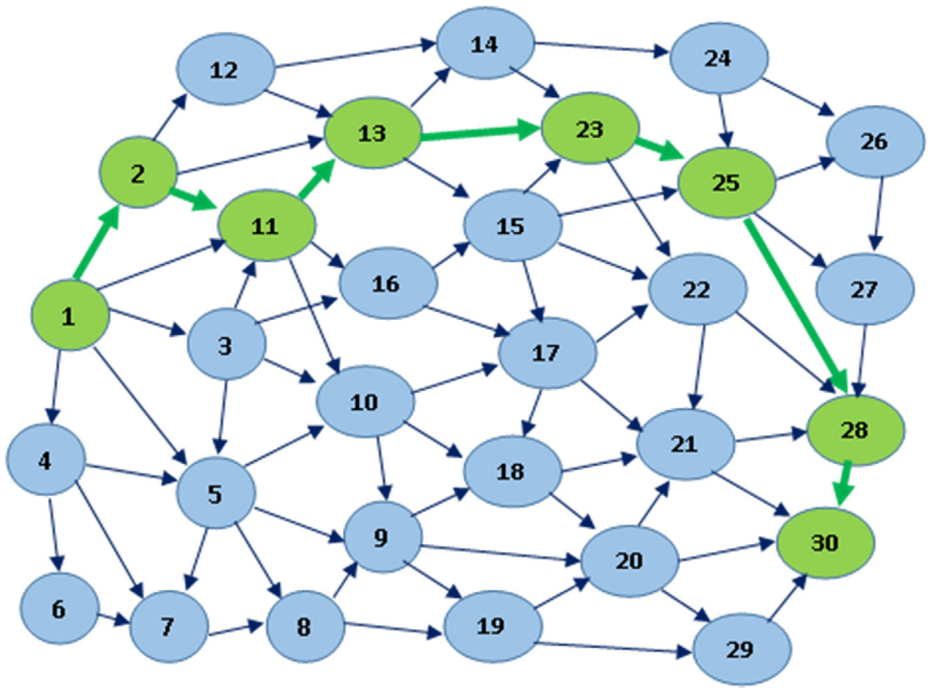

Fig. 12 Large graph setting.

Table 6 Fuzzy length of edges in the large graph setting.

Edges	Fuzzy Number	Edges	Fuzzy Number	Edges	Fuzzy number	Edges	Fuzzy number
(1, 2)	(2, 4, 5, 7)	(1, 3)	(3, 6)	(1, 4)	(4,5)	(1, 5)	(3, 5, 7, 9)
(1,11)	(1, 4, 9, 10)	(2, 11)	(2, 3)	(2, 12)	(3,7)	(2,13)	(6, 7, 8, 9)
(3, 5)	(4, 11)	(3, 10)	(6, 8, 10, 12)	(3, 11)	(2,3,4,5)	(3,16)	(4, 6, 7, 8)
(4,5)	(7, 9)	(4,6)	(5,5)	(4,7)	(3,5,7,9)	(5,7)	(1, 3, 5, 6)
(5,8)	(6, 8)	(5,9)	(5, 11)	(5, 10)	(4,8)	(6,7)	(7,2)
(7, 8)	(6, 7, 8, 9)	(8, 9)	(2,3)	(8, 19)	(4, 6, 8, 13)	(9,18)	(3, 5)
(9, 19)	(4,3)	(9,20)	(5,6,8,9)	(10, 9)	(8,2)	(10,17)	(5, 6, 7, 8)
(10, 18)	(8, 9, 11, 13)	(11, 10)	(6,7,8, 9)	(11, 13)	(3,3)	(11,16)	(2,4)
(12, 13)	(2,1)	(12, 14)	(3,2)	(13, 14)	(2,1)	(13,15)	(4,5)
(13, 23)	(1,1,2,3)	(14, 23)	(3,4)	(14, 24)	(2,4,5,8)	(15,17)	(3,4,5,6)
(15, 22)	(2,3,4,5)	(15, 23)	(1,1,2,3)	(15, 25)	(1,2)	(16,15)	(6, 7)
(16,17)	(9,4)	(17,18)	(1, 5)	(17, 21)	(1,2,3,4)	(17,22)	(1, 4)
(18, 20)	(1,3,4,5)	(18,21)	(2,4,5,7)	(19,20)	(2,2)	(19,29)	(8, 9, 10, 11)
(20, 21)	(6,5)	(20,29)	(6,6)	(20,30)	(3,4,5,6)	(21,28)	(4,6)
(21,30)	(3,4,5,6)	(22,21)	(7,7,8,9)	(22,28)	(4,5,7,8)	(23,22)	(1,2,3,3)
(23,25)	(1,2)	(24,25)	(2,3)	(24,26)	(1,3,4,5)	(25,22)	(1,3)
(25,26)	(2,2)	(25,27)	(3,4,5,7)	(25,28)	(3,5,7,9)	(26,27)	(3,3)
(27,28)	(4,3)	(28,30)	(1,3)	(29,30)	(3,5)		

algorithm was 65% faster than the ABC algorithm, 40% faster than the PSO algorithm, and 74% faster than GA in reaching convergence within the most complex evaluation scenario. Regarding the total runtime criterion, the ACO algorithm was between 35% and 65% faster than the other metaheuristic algorithms under identical implementation conditions in all three graphs. These results highlight the capacity of the algorithm to deal with complex evaluation scenarios in fuzzy routing environments.

Ethical approval

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

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 7 Implementation results in the large graph setting.

	Iteration	SP	Number of iterations to converge				Convergence time span (s)				Total time (s)			
			GA	PSO	ABC	ACO	GA	PSO	ABC	ACO	GA	PSO	ABC	ACO
1	60	1–2-11–13-23–25-28–30	11	25	21	4	2.09	2.54	1.99	0.39	15.47	9.87	9.15	5.36
2	60	1–2-11–13-23–25-28–30	9	8	38	2	1.73	0.87	3.58	0.21	15.11	9.67	8.87	5.45
3	60	1–2-11–13-23–25-28–30	13	4	6	6	2.42	0.55	0.66	0.55	15.48	9.62	8.99	5.12
4	60	1–2-11–13-23–25-28–30	26	–	33	4	4.14	–	3.24	0.39	15.86	9.56	9.17	5.38
5	60	1–2-11–13-23–25-28–30	44	3	22	16	10.9	0.36	1.96	1.43	15.22	9.81	8.64	5.63
6	60	1–2-11–13-23–25-28–30	7	27	6	5	1.27	2.69	0.62	0.46	15.77	9.11	8.66	5.60
7	60	1–2-11–13-23–25-28–30	5	–	46	7	1.04	–	4.24	0.71	15.36	9.69	9.16	5.62
8	60	1–2-11–13-23–25-28–30	17	6	23	15	3.37	0.80	2.16	1.42	15.23	9.23	8.79	5.38
9	60	1–2-11–13-23–25-28–30	28	–	51	27	4.41	–	4.88	2.57	15.12	9.24	8.91	5.41
10	60	1–2-11–13-23–25-28–30	3	17	4	3	0.79	1.88	0.46	0.32	14.98	9.38	9.28	6.09
Min	–	–	3	3	4	2	0.79	0.36	0.46	0.21	14.98	9.11	8.64	5.12
Max	–	–	44	–	51	27	10.9	–	4.88	2.57	15.86	9.87	9.28	6.09
Avg	–	–	16.3	–	25.0	8.90	3.22	1.38	2.38	0.84	15.36	9.53	8.96	5.50

Table 8 Average convergences within the different graph settings.

Graph	Number of iterations to converge				Convergence time span (s)				Total time (s)			
	GA	PSO	ABC	ACO	GA	PSO	ABC	ACO	GA	PSO	ABC	ACO
Initial	7.10	4.20	3.20	3.0	0.75	0.29	0.16	0.15	2.91	1.69	1.66	0.86
Enhanced	7.6	3.5	2.6	2.5	2.13	1.36	0.95	0.62	9.50	7.39	5.39	3.48
Large	16.3	–	25.0	8.90	3.22	1.38	2.38	0.84	15.36	9.53	8.96	5.50

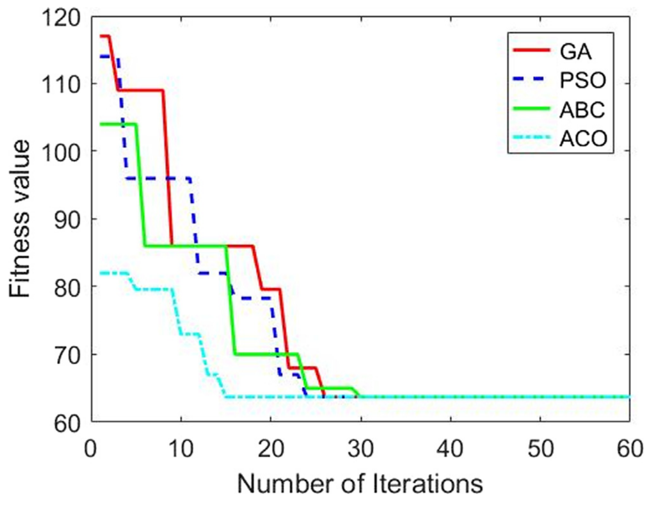

Fig. 14 Convergence of the algorithms implemented within the large graph setting.

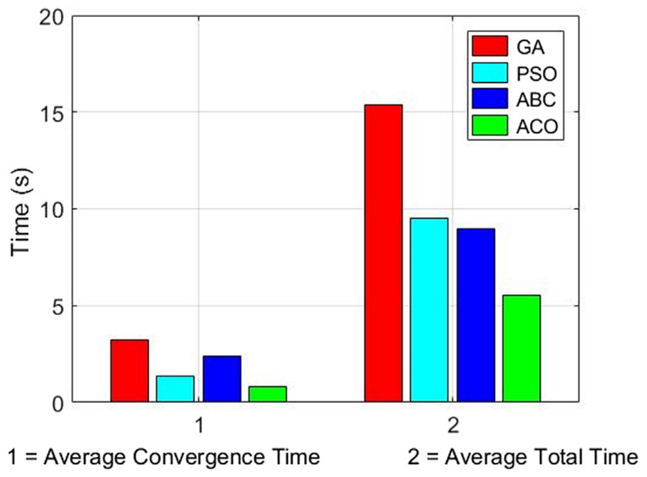

Fig. 13 Average convergence time and average runtime of the algorithms implemented within the large graph setting.

Acknowledgement

This work was supported by the Open Access Publishing Fund of the Free University of Bozen-Bolzano.

References

[1] X. Gao, Y. Xianzang, X. You, Y. Dang, G. Chen, X. Wang, Reachability for airline networks: fast algorithm for shortest path problem with time windows, Theoret. Comput. Sci. 749 (2018) 66–79.

[2] F.F. Dragan, A. Leitert, On the minimum eccentricity shortest path problem, Theoret. Comput. Sci. 694 (2017) 66–78.

[3] S. Mozes, Y. Nussbaum, O. Weimann, Faster shortest paths in dense distance graphs with applications, Theoret. Comput. Sci. 711 (2018) 11–35.

[4] J. Brito, F.J. Martínez, J.A. Moreno, J.L. Verdegay, An ACO hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints, Appl. Soft Comput. 32 (2015) 154–163.

[5] G. Bowatte, C.J. Lodge, L.D. Knibbs, B. Erbas, J.L. Perret, B. Jalaludin, R. Wood-Baker, Traffic related air pollution and development and persistence of asthma and low lung function, Environ. Int. 113 (2018) 170–176.

[6] W. Yu, Z. Liu, X. Bao, New LP Relaxations for Minimum Cycle/Path/Tree Cover Problems, Theoret. Comput. Sci. (2019).

[7] Kumar, R., & Kumar, M. (2010). Exploring genetic algorithm for shortest path optimization in data networks. Global Journal of Computer Science and Technology.

[8] Rares, M. (2015, June). Adaptive mutation in genetic algorithms for shortest path routing problem. In 2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) (pp. S-69). IEEE.

[9] M.A. Mohiuddin, S.A. Khan, A.P. Engelbrecht, Fuzzy particle swarm optimization algorithms for the open shortest path first weight setting problem, Appl. Intelligence 45 (3) (2016) 598–621.

[10] C. Dudeja, Fuzzy-based modified particle swarm optimization algorithm for shortest path problems, Soft. Comput. 23 (17) (2019) 8321–8331.

[11] A. Gupta, S. Srivastava, Comparative analysis of ant colony and particle swarm optimization algorithms for distance optimization, Procedia Comput. Sci. 173 (2020) 245–253.

[12] Y. Wang, J. Chen, W. Ning, H. Yu, S. Lin, Z. Wang, G. Pang, C. Chen, A time-sensitive network scheduling algorithm based on improved ant colony optimization, Alexandria Eng. J. 60 (1) (2021) 107–114.

[13] U. Zangina, S. Buyamin, M.S.Z. Abidin, M.S.A. Mahmud, Agricultural rout planning with variable rate pesticide application in a greenhouse environment, Alexandria Eng. J. 60 (3) (2021) 3007–3020.

[14] O. Dib, M.A. Manier, L. Moalic, A. Caminada, Combining VNS with genetic algorithm to solve the one-to-one routing issue in road networks, Comput. Oper. Res. 78 (2017) 420–430.

[15] O. Dib, L. Moalic, M.A. Manier, A. Caminada, An advanced GA–VNS combination for multicriteria route planning in public transit networks, Expert Syst. Appl. 72 (2017) 67–82.

[16] H. Garg, A hybrid GA-GSA algorithm for optimizing the performance of an industrial system by utilizing uncertain data, in: Handbook of research on artificial intelligence techniques and algorithms, IGI Global, 2015, pp. 620–654.

[17] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Appl. Math. Comput. 274 (2016) 292–305.

[18] R.S. Patwal, N. Narang, H. Garg, A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units, Energy 142 (2018) 822–837.

[19] H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems, Inf. Sci. 478 (2019) 499–523.

[20] R. De Santis, R. Montanari, G. Vignali, E. Bottani, An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses, Eur. J. Oper. Res. 267 (1) (2018) 120–137.

[21] D. Sedighizadeh, H. Mazaheripour, Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints, Alexandria Eng. J. 57 (4) (2018) 2225–2239.

[22] R. Hassanzadeh, I. Mahdavi, N. Mahdavi-Amiri, A. Tajdin, A genetic algorithm for solving fuzzy shortest path problems with mixed fuzzy arc lengths, Math. Comput. Modell. 57 (1) (2013) 84–99.

[23] A. Ebrahimnejad, Z. Karimnejad, H. Alrezaamiri, Particle swarm optimisation algorithm for solving shortest path problems with mixed fuzzy arc weights, Int. J. Appl. Decision Sci. 8 (2) (2015) 203–222.

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

[25] E.W. Dijkstra, A Note on Two Problems in Connection with Graphs, Numer. Math. 1 (1959) 269–271.

[26] L.R. Ford Jr, D.R. Fulkerson, A suggested computation for maximal multi-commodity network flows, Manage. Sci. 5 (1) (1958) 97–101.

[27] R.W. Floyd, Algorithm 97: shortest path, Commun. ACM 5 (6) (1962) 345.

[28] I. Mahdavi, R. Nourifar, A. Heidarzade, N.M. Amiri, A dynamic programming approach for finding shortest chains in a fuzzy network, Appl. Soft Comput. 9 (2) (2009) 503–511.

[29] A. Tajdin, I. Mahdavi, N. Mahdavi-Amiri, B. Sadeghpour-Gildeh, Computing a fuzzy shortest path in a network with mixed fuzzy arc lengths using α-cuts, Comput. Math. Appl. 60 (4) (2010) 989–1002.

[30] Y. Dou, L. Zhu, H.S. Wang, solving the fuzzy shortest path problem using multi-criteria decision method based on vague similarity measure, Appl. Soft Comput. 12 (6) (2012) 1621–1631.

[31] Y. Deng, Y. Chen, Y. Zhang, S. Mahadevan, Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment, Appl. Soft Comput. 12 (3) (2012) 1231–1237.

[32] Y. Zhang, Z. Zhang, Y. Deng, S. Mahadevan, A biologically inspired solution for fuzzy shortest path problems, Appl. Soft Comput. 13 (5) (2013) 2356–2363.

[33] J. Calle, J. Rivero, D. Cuadra, P. Isasi, Extending ACO for fast path search in huge graphs and social networks, Expert Syst. Appl. 86 (2017) 292–306.

[34] W. Ashour, R. Muqat, H. Al-Talli, Optimization of Traveling Salesman Problem based on Adaptive Affinity Propagation and Ant Colony Algorithms, Int. J. Comput. Appl. 181 (2018) 25–31.

[35] C. Changdar, R.K. Pal, G.S. Mahapatra, A genetic ant colony optimization based algorithm for solid multiple travelling salesmen problem in fuzzy rough environment, Soft. Comput. 21 (16) (2017) 4661–4675.

[36] H. Alrezaamiri, A. Ebrahimnejad, H. Motameni, Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm, Soft. Comput. (2018) 1–16.

[37] J. Wang, J. Cao, R.S. Sherratt, J.H. Park, An improved ant colony optimization-based approach with mobile sink for wireless sensor networks, J. Supercomput. (2017) 1–13.

[38] J. Qin, X. Shen, F. Mei, Z. Fang, An Otsu multi-thresholds segmentation algorithm based on improved ACO, J. Supercomput. (2018) 1–13.

[39] F. Tirado, R.J. Barrientos, P. González, M. Mora, Efficient exploitation of the Xeon Phi architecture for the Ant Colony Optimization (ACO) metaheuristic, J. Supercomput. 73 (11) (2017) 5053–5070.