Int J Adv Manuf Technol (2014) 70:1939–1954
DOI 10.1007/s00170-013-5383-1

ORIGINAL ARTICLE

Genetic and differential evolution algorithms for the allocation of customers to potential distribution centers in a fuzzy environment

A. Mahmoodi-Rad · S. Molla-Alizadeh-Zavardehi · R. Dehghan · M. Sanei · S. Niroomand

Received: 25 March 2012 / Accepted: 1 October 2013 / Published online: 5 November 2013

Abstract In order to serve the customers' demands in a supply chain, one of the important decisions is to select some candidate places as distribution centers (DCs) in the network. For opening a potential DC and also shipping from the DC to the customers, there are two types of costs named fixed and variable costs, respectively. Contrary to previous work, we consider fuzzy costs and utilize differential evolution (DE) algorithm for the first time for the given problem. In addition, some new crossover and mutation operators are proposed in DE. We also address the problem with genetic algorithm (GA) and compare the results with the presented DE algorithm. In the both presented algorithms, Prüfer number representation is employed. Besides, the Taguchi experimental design method is employed to study the behavior of the parameters dealing with the problem. To evaluate the performance of proposed algorithms, various problem sizes are considered and the computational results are analyzed. Finally, the impact of the rise in the problem size on the performance of the algorithms is investigated. The DE depicts a superior performance over GA in all problem sizes.

Keywords Allocation · Differential evolution (DE) algorithm · Genetic algorithm · Prüfer number · Fuzzy costs

A. Mahmoodi-Rad · R. Dehghan
Department of Mathematics, Islamic Azad University, Masjed Soleiman Branch, Masjed Soleiman, Iran

S. Molla-Alizadeh-Zavardehi (✉)
Department of Industrial Engineering, Islamic Azad University, Masjed Soleiman Branch, Masjed Soleiman, Iran
e-mail: saber.alizadeh@gmail.com

M. Sanei
Department of Mathematics, Central Tehran Branch, Islamic Azad University, Tehran, Iran

S. Niroomand
Department of Industrial Engineering, Istanbul Aydin University, Florya Istanbul, Turkey

1 Introduction

In recent years, the supply chain (SC) design problem has been gaining importance due to increasing competitiveness introduced by the market globalization [53]. Studies about SC have grown very rapidly over the last two decades, and an exponential growth in number of papers in different journals can be viewed. A supply chain is a combination of facilities and plants as a network that performs the operation of purchasing and supply, transformation of row materials into intermediate or finished products, and the distribution of these products to customers. Supply chains are generally characterized by numerous activities and operation among multiple functions and organizations.

One of the most interesting topics in Industrial Engineering research area has been supply chain management (SCM). SCM deals with a wide range of subject areas including procurement, logistics and transportation, behavior, marketing, network, strategic management, management information systems, and operations management. All of these areas have been cited in the literature dealing with SCM. A complete survey is available in Chen and Paulraj [16]. Today, SCM is still largely dominated by logistics and endeavors to observe the entire scope of the supply chain. Efficient coordination of material suppliers, manufacturing plants, wholesalers, retailers, warehouses, and distribution centers, to deliver products as scheduled, is the ultimate goal of SCM. In this

connection, many industrialists and academicians are keen to develop optimization methodologies and algorithms with artificial intelligence (AI) techniques that can optimize allocation problems, transportation policy, etc.

The fixed-charge transportation problem (FCTP) is a practical interest in business and industry. The FCTP is an extended case of the first developed transportation model, previously introduced by Hitchcock [29]. Since Hitchcock [29] first developed the transportation model (TP), it has been extended from many numbers of researchers, such as Charnes and Copper [15], Dantzig [18], Arsham and Khan [5], Arsham [4], Adlakha and Kowalski [2], Brenner [13], and others.

In a FCTP, fixed cost is incurred for every route that is used in the solution, along with the variable cost that is proportional to the amount shipped. The objective is to find the combination of routes that minimizes the total variable and fixed costs while satisfying the supply and demand requirements of each origin and destination.

Consider the m potential distribution centers (DCs) and n customers with particular demands, and the transportation cost per unit from DC i to customer j is a trapezoidal fuzzy number, $\widetilde{c}_{ij}$, also, assumed for opening potential DC i, an opening cost is a trapezoidal fuzzy number, $\widetilde{f}_i$. In this problem, the objective is to find the following items, in order to minimize the total system cost:

- The DCs; the opened candidate places.
- Each customer receives its demand from which DC?

In order to, the mathematical model for this problem is:

$$\begin{aligned}
&\text{Min} \quad \widetilde{Z} = \sum_{i=1}^{m}\sum_{j=1}^{n} \widetilde{c}_{ij} \times x_{ij} \times b_j + \sum_{i=1}^{m} \widetilde{f}_i \times y_i,\\
&\text{S.t.}\\
&\sum_{i=1}^{m} x_{ij} = 1 \quad j = 1, 2, \ldots, n,\\
&x_{ij} \in \{0, 1\} \quad ,\\
&y_i = 0 \quad \text{if} \quad \sum_{j=1}^{n} x_{ij} = 0,\\
&y_i = 1 \quad \text{if} \quad \sum_{j=1}^{n} x_{ij} > 0.
\end{aligned}$$

In this model, b_j is the demand of costumer j.

2 Literature review

In this several decades, there have been many researchers who developed new models or methods to design the supply chain network that can give the least cost [27]. Supply chain network (SCN) design is a strategic issue, which aims at selecting the best combination of a set of facilities to achieve an efficient and effective management of the supply chain. It not only helps to achieve the SCN goals, but also it offers a great potential to reduce costs and to improve service quality. Furthermore, a main factor that influences on SCN is to decide the number of distribution centers (DCs). Geoffrion and Graves [24] worked on two-stage distribution problem for the first time. A new mathematical formulation to locate a number of production plants and warehouses and to design distribution network is presented by Pirkul and Jayaraman in which the total operating cost can be minimized. Their approach was based on Lagrangian relaxation to solve the problem. Hindi et al. [28] studied a two-stage distribution-planning problem. They supposed a single DC to serve all customers. They developed new mathematical model as well as a branch and bound algorithm to solve the problem.

Jawahar and Balaji [30] proposed a genetic algorithm (GA) to solve a two-stage FCTP, by considering unit transportation cost, fixed cost associated with each route, and uncapacitated DCs. Balaji and Jawahar [6] presented a simulated annealing (SA) based algorithm to solve the two-stage FCTP which is considered by Jawahar and Balaji [30]. Borisovsky et al. [12] considered the supply management problem with lower-bounded demands (SMPLDs), which plans the shipments from a set of suppliers to a set of customers in order to minimize the total cost, by considering the given lower and upper bounds on shipment sizes, lower-bounded consumption and linear costs (including fixed cost and variable cost) for opened deliveries; they proposed two variants of GA for SMPLD.

Antony Arokia Durai Raj et al. [3] proposed genetic algorithms to solve a two-stage transportation problem with two different scenarios. The first scenario considers the per-unit transportation cost and the fixed cost associated with a route, coupled with unlimited capacity at every DC. The second scenario considers the opening cost of a distribution center, per-unit transportation cost from a given plant to a given DC and the per-unit transportation cost from the DC to a customer. Costa et al. [17] presented a GA to minimize the total logistic cost resulting from the transportation of goods and the location and opening of the facilities in a single product three-stage supply chain network.

Prüfer number encoding is a useful type of tree encoding and solution representation method in the related works [22, 34, 35]. This method is one of the efficient ways and most used methods to represent various network problems. The use of the Prüfer number representation for solving various network problems was introduced by Gen and Cheng [20]. They employed the Prüfer number as an efficient way, which uniquely represent all possible trees in a network graph [21]. They mentioned that the use of the Prüfer number is more suitable for encoding a tree, especially for some research areas like; production/distribution problem [23, 25, 52], some extended transportation problems [51], and minimum spanning problems [21, 50].

Zhou et al. [59] also used the Prüfer number representation for allocation of customers to multiple distribution centers in the supply chain network using genetic algorithm. They formulated the balanced allocation problem as a balanced star-spanning forest problem. A hybrid genetic algorithm for

production/distribution problem is proposed by Syarif and Gen [51]. They represented the solution on each stage by Prüfer number and utilized a hybrid approach to enhance the performance of their proposed hybrid GA.

In the recent works, Hajiaghaei-Keshteli [25] considered the allocation of customer to potential DCs, and addressed the model by GA and artificial immune algorithm. His proposed model selects some potential places as distribution centers in order to supply demands of all customers. He used Prüfer number representation for the presented algorithms. In addition, Molla-Alizadeh-Zavardehi et al. [39] proposed artificial immune and genetic algorithms with a Prüfer number representation to solve a capacitated two-stage FCTP, by considering unit transportation cost, fixed cost associated with each route, fixed cost for opening potential distribution centers (DCs), and capacitated DCs or warehouses.

In addition, in a recent work, Hajiaghaei-Keshteli et al. [26] used Prüfer number representation for a type of transportation problem and developed some novel mutation and crossover operators. They also improved the spanning tree-based genetic algorithm by presenting a pioneer method to design a chromosome that does not need a repairing procedure for feasibility. Xie and Jia [54] proposed a genetic algorithm to solve a nonlinear FCTP, by considering transportation cost depending on the quadratic of the shipping units shown as a nonlinear term, and also fixed cost associated with each route.

Generally, it is often difficult to estimate the actual penalties (e.g., transportation cost, delivery time, quantity of goods delivered, under-used capacity, etc.), demands, availabilities, the capacities of different modes of transport between origins and destinations. Depending upon different aspects, these parameters fluctuate due to uncertainty in judgment, lack of evidence, insufficient information, etc. Hence, the typical models, in which all parameters are crisp numbers, fail in many practical applications. In recent years, the problems involving uncertainty has become the subject of extensive research.

The purpose of introducing entropy in transportation problem is to get better customer service. Bit [9], Bit et al. [10, 11], and Li and Lai [36] presented the fuzzy compromise programming approach to multiobjective transportation problem. Samanta and Roy [48] proposed an algorithm for solving multiobjective entropy transportation problem under fuzzy environment. Omar and Samir [44] and Chanas and Kuchta [14] discussed the solution algorithm for solving the transportation problem in fuzzy environment. The entropy optimization in transportation models as well as other models also is discussed in the book of Kapur and Kesavan [31]. Ojha et al. [43] discussed a solid transportation problem with entropy in fuzzy environment. For a further review on transportation and location problems in SCM, we refer to Melo et al. [38] and Ko et al. [33].

Problems involving uncertainty has become the subject of extensive research in the last decade. The typical models, in which all parameters are crisp numbers, fail in many practical applications. For most of the real-world processes, some parameters (e.g., costs) are not precisely known a priori. There are several approaches to modeling the uncertainty in optimization. The natural one is to apply the theory of fuzzy.

In this paper, two stages of supply chain network, distribution centers and customers, are considered. The customers have particular demands and potential places are candidate to be selected as distribution centers. All the potential DCs can send their products to any of the customers. There are two types of costs: opening cost, assumed for opening a potential DC plus shipping cost per unit from DC to the customers. Previous papers are based upon the condition that all types of costs are known exactly. However, the costs are often imprecise in many real-world applications and the imprecision is critical for the decisions made in supply chain. Hence, contrary to previous works, the proposed model selects some potential places as distribution centers in order to supply demands of all the customers, considering fuzzy costs.

As can be viewed in the previous works, genetic algorithm has been successfully employed to such a problem. The DE is a new metaheuristic, which has great abilities to reach good approximations of optimal solutions to numerous NP-hard problems. In this paper, at first, differential evolution (DE) algorithm is used for the first time in this research area. Then, a genetic algorithm is presented to compare with the presented DE. In addition, some novel crossover and mutation operators are proposed in DE. In both presented algorithms, Prüfer number representation is employed.

Five sections follow this introduction. The next section briefly introduced some knowledge of fuzzy costs. Section 3 describes the problem's details and elaborates the mathematical formulation of our model. The proposed algorithms are explained in Section 4 and 5. Section 6 describes the Taguchi experimental design and compares the computational results. Finally, in Section 7, conclusions are provided and some areas of further research are then presented.

3 Fuzzy model

Zadeh [58] introduced the theory of the fuzzy set with the membership function, and then, it has been well developed and employed in a wide variety of real problems. In the classic transportation problems, it is usually assumed that the aspects of the problem in hand are certain. Most existing models neglect the presence of uncertainty within a transportation

environment. In many real-world distribution problems, however, uncertainty and vagueness in required time or cost often do exist that make the models more complex. This uncertainty might come about because of delivery problems (e.g., transportation delay, traffic jam). The more transportation time, the larger the transportation time becomes. To describe the uncertainty in the distribution management, fuzzy logic is appropriate.

In order to measure a fuzzy event, the term fuzzy variable was introduced by Kaufman [32]. In this section, we briefly introduce some basic concepts and results about fuzzy measure theory initiated by Bellman and Zadeh [7]. Below, we give definitions and notations taken from Bezdek [8].

Definition 3.1 If X is a collection of objects denoted generically by x, then a fuzzy set in X is a set of ordered pairs: $\widetilde{A} = \left\{x, \widetilde{A}(x) \middle| x \in X\right\}$, where $\widetilde{A}(x)$ is called the membership function, which associates with each $x \in X$ a number in [0,1] indicating to what degree x is a number.

Definition 3.2 The α level set of $\widetilde{A}$ is the set $\widetilde{A}_\alpha = \left\{x \middle| \widetilde{A}(x) \geq \alpha\right\}$ where $\alpha \in [0, 1]$. The lower and upper bounds of any α level set $\widetilde{A}_\alpha$ are represented by finite number $\inf_{x \in \widetilde{A}_\alpha}$ and $\sup_{x \in \widetilde{A}_\alpha}$.

Definition 3.3 A fuzzy set A is convex if

$$\widetilde{A}(\lambda x + (1-\lambda)y) \geq \min\left\{\widetilde{A(x)}, \widetilde{A}(y)\right\} \quad \forall x, y \in X, \lambda \in [0, 1]$$

Definition 3.4 A convex fuzzy set $\widetilde{A}$ on $\mathbb{R}$ is a fuzzy number if the following conditions hold:

(a) Its membership function is piecewise continuous function.

(b) There exist three intervals $[a,b]$, $[b,c]$ and $[c,d]$ such that A is increasing on $[a,b]$, equal to 1 on $[b,c]$, decreasing on $[c,d]$ and equal to 0 elsewhere.

Definition 3.5 The support of a fuzzy set $\widetilde{A}$ is a set $\widetilde{A}$ is a set of elements in X for which $\widetilde{A}(x)$ is positive, that is,

$$\operatorname{supp}\widetilde{A} = \left\{x \in X \middle| \widetilde{A}(x) > 0\right\}$$

Definition 3.6 Let $A = (a^l, a^u, a, \beta)$ denote the trapezoidal fuzzy number, where $[a^l, \alpha, a^u + \beta]$ is the support of $\widetilde{A}$ and $[a^l, a^u]$ its core.

Remark 3.1 In this paper, we denote the set of all fuzzy numbers by $F(\mathbb{R})$.

We next define arithmetic on trapezoidal fuzzy numbers.

Let $\widetilde{a} = (a^l, a^u, \alpha, \beta)$ and $\widetilde{b} = (b^l, b^u, \gamma, \theta)$ be two trapezoidal fuzzy numbers. Define,

$$x > 0 \ , \quad x \in \mathbb{R} : x\widetilde{a} = \left(xa^l, xa^u, x\alpha, x\beta\right),$$
$$x < 0 \ , \quad x \in \mathbb{R} : x\widetilde{a} = \left(xa^u, xa^l, -x\beta, -xa\right),$$
$$\widetilde{a} + \widetilde{b} = \left(a^l + b^l, a^u + b^u, \alpha + \gamma, \beta + \theta\right),$$
$$\widetilde{a} - \widetilde{b} = \left(a^l - b^u, a^u - b^l, \alpha + \theta, \beta + \gamma\right).$$

3.1 Ranking function

A convenient method for comparing of the fuzzy numbers is by use of ranking function [37]. We define a ranking function $\mathscr{R}$: $F(\mathbb{R}) \to \mathbb{R}$, which maps each fuzzy number into the real line. Now, suppose that $\widetilde{a}$ and $\widetilde{b}$ be two trapezoidal fuzzy numbers. Therefore, we define,

$$\widetilde{a} \underset{\mathfrak{R}}{\geq} \widetilde{b} \text{ If and only if } \mathfrak{R}\left(\widetilde{a}\right) \geq \mathfrak{R}\left(\widetilde{b}\right) \tag{1}$$

$$\widetilde{a} \underset{\mathfrak{R}}{>} \widetilde{b} \text{ If and only if } \mathfrak{R}\left(\widetilde{a}\right) > R\left(\widetilde{b}\right) \tag{2}$$

$$\widetilde{a} \underset{\mathfrak{R}}{=} \widetilde{b} \text{ If and only if } \mathfrak{R}\left(\widetilde{a}\right) = \mathfrak{R}\left(\widetilde{b}\right) \tag{3}$$

Also we write

$$\widetilde{a} \underset{\mathfrak{R}}{\leq} \widetilde{b} \text{ If and only if } \widetilde{b} \underset{\mathfrak{R}}{\geq} \widetilde{a}.$$

Since there are many ranking function for comparing fuzzy numbers, we only apply linear

$$\mathfrak{R}\left(k\widetilde{a} + \widetilde{b}\right) = k\mathfrak{R}\left(\widetilde{a}\right) + \mathfrak{R}\left(\widetilde{b}\right),$$

For any $\widetilde{a}$ and $\widetilde{b}$ belonging to $F(\mathbb{R})$ and any $k \in F(\mathbb{R})$.

Here, we introduce a linear ranking function adopted by Maleki [37]. For a trapezoidal fuzzy number $\widetilde{a} = \left(a^l, a^u, \alpha, \beta\right)$, we use ranking function as follows:

$$\mathfrak{R}\left(\widetilde{a}\right) = \int_0^1 \left(\inf \widetilde{a}_\alpha + \sup \widetilde{a}_\alpha\right) da,$$

This reduces to

$$\mathfrak{R}\left(\widetilde{a}\right) = a^l + a^u + \frac{1}{2}\left(\beta - \alpha\right).$$

Then, for trapezoidal fuzzy numbers $\widetilde{a} = \left(a^l, a^u, \alpha, \beta\right)$ and $\widetilde{b} = \left(b^l, b^u, \gamma, \theta\right)$, we have

Table 1 The example parameters and costs

Customers		1	2	3	4	5	6	7
b_j		4,000	5,000	1,700	2,800	5,200	8,400	6,000
DCs	$\widetilde{f}_i$	$\widetilde{c}_{ij}$						
1	(350,550,150,250)	(1,3,2,5)	(2,7,1,4)	(2,5,1,4)	(3,8,1,5)	(5,6,2,4)	(3,18,1,5)	(4,5,2,4)
2	(250,400,150,200)	(4,6,2,3)	(6,8,5,1)	(4,10,1,5)	(8,9,5,1)	(3,7,1,5)	(6,10,1,3)	(2,9,1,2)
3	(700,900,500,100)	(3,7,2,2)	(3,9,1,1)	(6,11,1,3)	(5,7,3,2)	(3,7,2,5)	(3,15,2,2)	(7,8,4,4)
4	(200,450,100,250)	(3,5,1,1)	(5,7,3,6)	(8,13,2,4)	(4,5,1,1)	(5,9,3,1)	(4,5,3,4)	(3,5,2,1)

$\widetilde{a} \geq \widetilde{b}$ if and only if $a^l + a^u + \frac{1}{2}(\beta - \alpha) \geq b^l + b^u + \frac{1}{2}(\theta - \gamma)$.

For an illustration more about of the above model, we solve an example here. Suppose there are four potential DCs and seven costumers. The transportation costs and the opening cost are given in Table 1.

Suppose customers 2 and 4 are served their demands from DC 1, customers 1, 3, and 7 are served from DC 2 and other customers received their demands from DC 4, as shown in Fig. 1. Thus, we suppose that just potential plants 1, 2, and 4 are considered as DCs. Thus,

- The opening cost of DCs $\widetilde{f} = \widetilde{f}_1 + \widetilde{f}_2 + \widetilde{f}_3 + \widetilde{f}_4 = (350,\ 550,\ 150,\ 250) + (250,\ 400,\ 150,\ 200) + (700,\ 900,\ 500,\ 100) + (200,\ 450,\ 100,\ 250) = (1,500,\ 2,300,\ 900,\ 800)$, $\Re(\widetilde{f}) = \Re(1500,\ 2,300,\ 900,\ 800) = 3,750$.

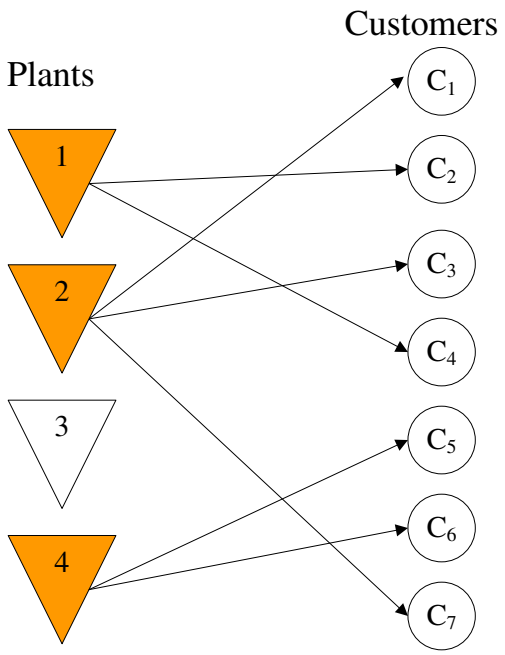

Fig. 1 Transportation graph for the example

- The transportation cost from DCs to customers is equal to:

$$\begin{aligned}
\widetilde{B} &= b_1 \times \widetilde{c}_{21} + b_2 \times \widetilde{c}_{12} + b_3 \times \widetilde{c}_{23} + b_4 \times \widetilde{c}_{14} + b_5 \times \widetilde{c}_{45} \\
&\quad + b_6 \times \widetilde{c}_{46} + b_7 \times \widetilde{c}_{27} \\
&= 4,000 \times (4,\ 6,\ 2,\ 3) + 5,000 \times (2,\ 7,\ 1,\ 4) \\
&\quad + 1,700 \times (4,\ 10,\ 1,\ 5) + 2,800 \times (3,\ 8,\ 1,\ 5) \\
&\quad + 5,200 \times (5,\ 9,\ 3,\ 1) + 8,400 \times (4,\ 5,\ 3,\ 4) \\
&\quad + 6,000 \times (2,\ 9,\ 1,\ 2) \\
&= (16,000,\ 24,000,\ 8,000,\ 12,000) \\
&\quad + (10,000,\ 35,000,\ 5,000,\ 20,000) \\
&\quad + (6,800,\ 17,000,\ 1,700,\ 8,500) \\
&\quad + (8,400, 22,400, 2,800, 14,000) \\
&\quad + (26,000,\ 46,800,\ 15,600,\ 5,200) \\
&\quad + (33,600, 42,000, 25,200, 33,600) \\
&\quad + (12,000,\ 54\ 000,\ 6,000,\ 12,000) \\
&= (112,800,\ 241,200,\ 64,300,\ 105,300), \\
\Re\left(\widetilde{B}\right) &= \Re(112,800,\ 241,200,\ 64,300,\ 105,300) \\
&= 374,500.
\end{aligned}$$

- The objective function value is:

$$\begin{aligned}
\widetilde{Z} &= (1,500, 2,300,\ 900,\ 800) \\
&\quad + (112,800,\ 241,200,\ 64,300,\ 105,300) \\
&= (114,300,\ 243,500,\ 65,200,\ 106,100), \\
\Re\left(\widetilde{Z}\right) &= 378,250 = 3,750 + 374,500.
\end{aligned}$$

4 The proposed GA

GAs are intelligent stochastic optimization techniques based on the mechanism of natural selection and genetics. GA is one of the most used metaheuristic approaches for tackling optimization problems. It is based on the idea of "survival of the fittest," which repeats evaluation, selection, crossover, and mutation after initialization until a stopping criterion is satisfied. It also has been shown as a robust optimization technique to solve many real world problems [20, 21].

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. They evaluate the individuals in the population and then generate new solutions for the next generation. New chromosomes are produced by reproduction, crossover, or mutation. In other words, these three genetic operators take the initial population and generate successive populations that improve over time. Chromosomes are then appraised according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated.

The new population is then evaluated again and the whole process is repeated. The overall procedure of proposed genetic algorithm is shown in Fig. 2.

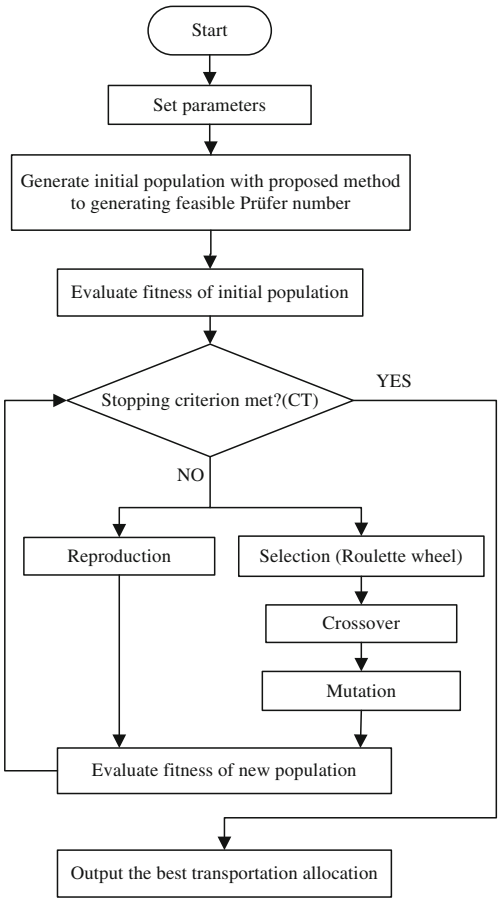

Fig. 2 The proposed GA flowchart. *CT* CPU time

4.1 Representation

The key to find a good solution using a GA lies in developing a good chromosome representation of solutions to the problem. A good GA chromosome design should reduce or eliminate redundant chromosomes from the population. Solution representation should be easy to decode to reduce the cost of the algorithm. The problem discussed in this paper is a network problem, which one of its feasible representation is a tree topology. Thus, tree-based representation would also appropriate for this problem. The use of the Prüfer number representation for solving various network problems was used by Gen and Cheng [20], Hajiaghaei-Keshteli [25], Molla-Alizadeh-Zavardehi et al. [39], and Hajiaghaei-Keshteli et al. [26]. They utilized the Prüfer number as it is capable of equally and uniquely representing all possible trees in a network graph.

Prüfer number encoding procedure:

Repeat the following steps until no edges are left in network.

Step 1 Let node j be the smallest labeled customer node in network.

Step 2 Set k to be the first digit in the permutation, if node k is incident to node j. Herein, we build the permutation by appending digits to the right so that the permutation can be built and read from left to right.

Step 3 Remove node j and the edge (j,k).

It is also possible to generate a unique network from a Prüfer number via the following decoding procedure:

Prüfer number decoding procedure:

Repeat the following steps until no digits are left in P.

Step 1 Let P be the original Prüfer number and P' the set of all nodes not included in P.

Step 2 Let j be the node with the smallest label in P' and let k be the leftmost digit of P. Add the edge from j to k into the network.

Step 3 Remove j from P' and k from P.

For instance, the Prüfer number for the network graph explained in example in Section 3, is generated and shown below:

P = | 2 | 1 | 2 | 1 | 4 | 4 | 2 |

4.2 Initialization

According to the most of the evolutionary algorithms, we use a random procedure to generate an initial set of solutions. Considering the decoding procedure, as explained in Section 4.1, and having m DC nodes and n customer nodes, we should represent a permutation of n digits in length in

which all digits are between 1 and m (number of DCs) inclusive and each digit is generated with the probability of $1/m$.

4.3 Selection mechanism

The aim is to minimize of total cost, and better solutions are those results in lower objective function. The greater fitness value means the better chromosome, so we consider the following function to calculate each fitness value:

$$\text{Fitness value} = \frac{1}{\text{Objective function}}$$

Roulette–Wheel selection mechanism is employed to show that a solution with higher fitness value has more chance to be selected for the next generation or operation.

4.4 Genetic operators

4.4.1 Reproduction

Chromosomes with higher fitness values are more desirable than the other, so one of the strategies in GAs is to keep the p_r% of the chromosomes with the greater fitness values. Hence, they are copied to the next generation.

4.4.2 Crossover

The genetic crossover operators generate new sequence by operating on two chromosomes at a time and combining both chromosomes features. The goal is to create better offsprings after combining the parents. As we assigned p_r% of the chromosomes of generation to reproduction, the $(1-p_r)$% remaining chromosomes are generated through crossover operator.

Many different general and specific crossover operators have been proposed for such a network problem (Gen and Cheng [20], Hajiaghaei-Keshteli [25], Molla-Alizadeh-Zavardehi et al. [39], and Hajiaghaei-Keshteli et al. [26]). In this paper, we employ the following operators:

- *One-point crossover*: one point is randomly selected for dividing both parents to two parts, and then, the opposite parts of parents are combined together.
- *Two-point crossover:* at first, two crossover points are randomly selected. The digits between randomly selected points are inherited from one parent to the offspring. The other symbols are placed in the order they appeared in the other parent. After changing the roles of parents, the same procedure is applied to produce the second offspring. This operator is proposed by Murata and Ishibuchi [40].
- *Uniform crossover:* a random crossover mask is generated and then relative genes between parents in accordance with the mask are exchanged.

4.4.3 Mutation

For each selected chromosome, we present following five mutation operators:

- *Swap mutation*: swapping the content of two random adjacent genes.
- *Big swap mutation*: two genes are randomly selected and swapped.
- *Inversion mutation*: two positions are selected, and the sub string is inverted between them.
- *Displacement mutation*: a substring is selected and inserted in a position.
- *Perturbation mutation*: a gene is randomly selected and replaced by randomly generated another root–node.

5 The proposed DE algorithm

The DE is a population-based and stochastic global evolution algorithm, created by Storn and Price [49], whose main objective is functions optimization. It is one strategy based on evolutionary algorithms with some specific characteristics. The prime idea of DE is to adapt the search during the evolutionary process [55]. The theoretical framework of DE is very simple and DE is computationally inexpensive in terms of memory requirements and CPU times. Thus, nowadays, DE has gained much attention and wide application in a variety of fields [46]. Due to its simplicity, easy implementation, fast convergence, and robustness, DE is effective for solving various optimization problems with nonsmooth and nonconvex characteristics. The programming and operation of DE are also quite easy because it requires the settings of only three control parameters: population size, scaling factor, and crossover constant rate (CR) in crossover operator. These advantages facilitate the wide usage of DE.

DE starts with a number of populations of NP candidate solutions, so-called individuals. The DE's main strategy is to generate new individuals by calculating vector differences between other randomly selected individuals of the population. The subsequent generations in DE are denoted by $G=0, 1,\ldots,G_{\max}$. It is usual to denote each individual as a D-dimensional vector $X_{i,G}=X^1_{i,G},\ldots,\ X^D_{i,G}\}$, $i=1,2,\ldots,\text{NP}$, called a target vector. The key idea behind DE is a scheme for generating trial vectors. The main operation is founded on the differences of randomly sampled pairs of solutions in the population. The main difference between traditional evolutionary algorithms and DE is that in traditional evolutionary algorithms, mutation results in small perturbations to the genes of an individual, while in DE, the mutation is an arithmetic combination of individuals [19]. The main steps of the DE are shown in Fig. 3.

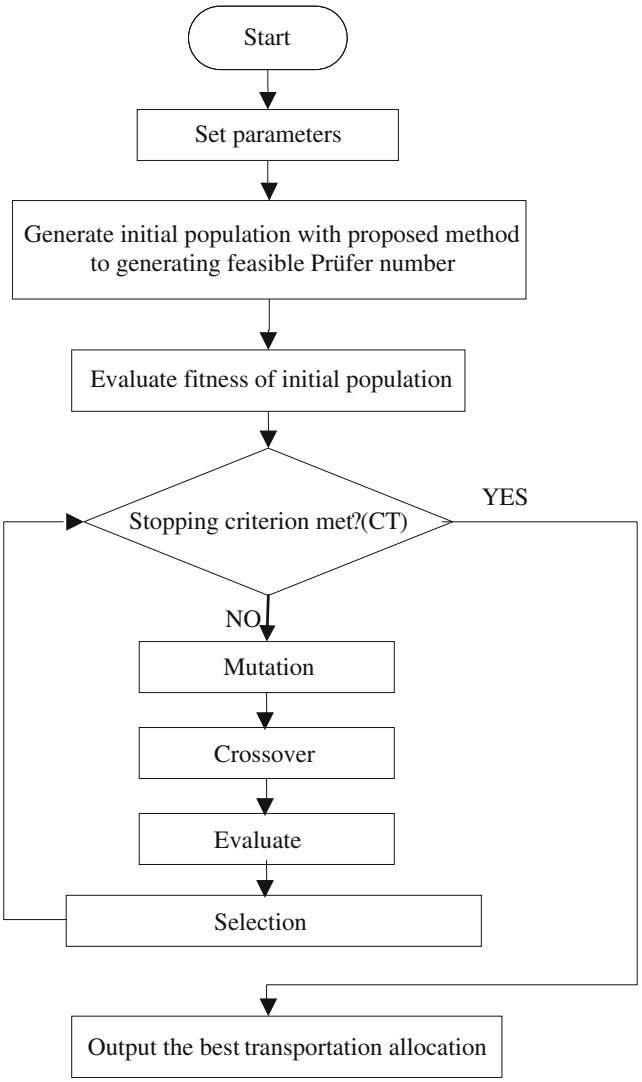

Fig. 3 The proposed DE Algorithm

5.1 Representation

Since DE works in continues space, and we want to show a network by a vector as a solution, a procedure is developed to represent the feasible solutions in this paper. By explaining an example, we show that how we can create a feasible vector.

Remember the example explained in Section 3. In that example, we had four DCs and the digits in the Prüfer number could be one of the following numbers: 1, 2, 3, 4. Since plant 2 is not selected as a DC, number 2 does not exist in the Prüfer number. Here, we want to depict that how this Prüfer number can be obtained and represented.

For four potential DCs and seven customers, as mentioned above, the Prüfer number should have seven integer digits between 1 and 4. At first, we create seven random numbers that can be real numbers in continues space. Then, these digits should be modified in a way that all seven random digits change to seven integer digits between 1 and 4. Because of shortening the explanation of modification procedure, we depict the example shown in Fig. 4, which consider all possible states that can be occurred. As shown, at first seven random real digits are considered, then digits <1, and more than 4 are changed to 1 and 4, respectively. Other real digits are altered to first integer digit that is equal or more than the random digit.

5.2 Initial population

Like other evolutionary algorithms, DE works with a population of individuals (candidate solutions) and the number of population never changes during the optimization process. Normally, the initial population is randomly generated, and the population will be improved by the algorithm iteratively, through the mutation, crossover, and selection operators. As will be explained in the types and operation of mutation operators in the next section, some of digits may go less than zero or more than the number of DCs. In this occasion, we must use the modification procedure explained in Section 5.1 and Fig. 4.

5.3 Mutation operator

According to the DE, after initialization, it employs the mutation operator. The mutation in DE is a distinct innovation. It is based on the difference of different individuals (solutions), to produce a mutant vector $V_{i,G}$ with respect to each individual $X_{i,G}$, in the current population. This main operation is founded on the differences of randomly sampled pairs of solutions in the population. For each target vector $X_{i,G}$, $i=1, 2,\ldots,$ NP, a mutant vector $V_{i,G}$ can be made by the following mutation operators. In all types, the scale factor F is a positive control parameter for scaling the difference vector. The following operators mostly used in the related papers:

Mutation type 1, Storn and Price [49]:

$$V_{i,G} = X_{r_1,G} + F\left(X_{r_2,G} - X_{r_3,G}\right),$$

Mutation type 2, Qin and Suganthan [47]:

$$V_{i,G} = X_{\text{best},G} + F\left(X_{r_1,G} - X_{r_2,G}\right),$$

Mutation type 3, Qin and Suganthan [47]:

$$V_{i,G} = \mathrm{X}_{i,G} + F\left(X_{\text{best},G} - X_{i,G}\right) + F\left(\mathrm{X}_{r_1,G} - \mathrm{X}_{r_2,G}\right),$$

Random real digits:	1.73	-0.5	1.56	0.25	3.66	4.79	2
Modification	1.73	1	1.56	0.25	3.66	4	2
Representation	2	1	2	1	4	4	2

Fig. 4 Representation an example for DE

Mutation type 4, Storn and Price [49] and Abbass et al. [1]:

$$V_{i,G} = X_{\text{best},G} + F\left(X_{r_1,G} - X_{r_2,G}\right) + F\left(X_{r_3,G} - X_{r_4,G}\right),$$

Mutation type 5, Qin and Suganthan [47]:

$$V_{i,G} = X_{r_1,G} + F\left(X_{r_2,G} - X_{r_3,G}\right) + F\left(X_{r_4,G} - X_{r_5,G}\right),$$

Mutation type 6, Qin and Suganthan [47]:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best},G} - X_{r_2,G}\right) + F\left(X_{r_3,G} - X_{r_4,G}\right),$$

Mutation type 7, Qin and Suganthan [47]:

$$V_{i,G} = X_{i,G} + F\left(X_{r_1,G} - X_{r_2,G}\right) + F\left(X_{r_3,G} - X_{r_4,G}\right),$$

Mutation type 8, Nearchou [41]:

$$V_{i,G} = kX_{best,G} + (1-k)X_{r_1,G} + F\left(X_{r_2,G} + X_{r_3,G} - X_{r_4,G} - X_{r_5,G}\right),$$

Mutation type 9, Nearchou [42]:

$$V_{i,G} = tX_{r_1,G} + (1-t)X_{r_2,G} + F\left(X_{r_3,G} - X_{r_4,G}\right),$$

The indices r_1,r_2,r_3,r_4,r_5 are mutually exclusive integers randomly generated within the range [1, NP], The scale factor F is a positive control parameter for scaling the difference vector, $t \in (0,1)$ is chosen randomly. X_{best} is the solution with the best performance among individuals of the population at generation G. k is a coefficient for convex combination between the best element $X_{best,G}$ of the current population and randomly selected element $X_{r1,G}$.

Since the main difference between GA and DE algorithms is the operation of mutation operator, we develop new nine mutation operators for the first time in this paper. These new nine mutation operators plus above nine mutation operators (18 operators) will be used in parameter tuning to calibrate the operator. The new mutation operators are as follows:

Mutation type 10:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best}} - X_{i,G}\right),$$

Mutation type 11:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best},G} - X_{r_2,G}\right),$$

Mutation type 12:

$$V_{i,G} = X_{r_1,G} + F\left(X_{i,G} - X_{r_1}\right),$$

Mutation type 13:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best},G} - X_{i,G}\right) + F\left(X_{r_2,G} - X_{r_3,G}\right),$$

Mutation type 14:

$$V_{i,G} = X_{r_1,G} + F\left(X_{i,G} - X_{r_1,G}\right) + F\left(X_{r_2,G} - X_{r_3,G}\right),$$

Mutation type 15:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best},G} - X_{r_2,G}\right) + F\left(X_{\text{best},G} - X_{i,G}\right),$$

Mutation type 16:

$$V_{i,G} = X_{r_1,G} + F\left(X_{i,G} - X_{r_1,G}\right) + F\left(X_{\text{best},G} - X_{r_1,G}\right),$$

Mutation type 17:

$$V_{i,G} = X_{r_1,G} + F\left(X_{\text{best},G} + X_{r_2,G}\right) + F\left(X_{\text{best},G} - X_{r_3,G}\right),$$

Mutation type 18:

$$V_{i,G} = X_{r_1,G} + F\left(X_{i,G} - X_{r_2,G}\right) + F\left(X_{i,G} - X_{r_3,G}\right),$$

5.4 Crossover operator

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced after the mutation operation. Crossover operation is employed to generate a temporary or trial vector by replacing certain parameters of the target vector by the corresponding parameters of a randomly generated donor vector. To get each individual's trial vector, $U_{i,G+1}$, crossover operation is performed between each individual and its corresponding mutant vector. We propose four types of crossover operators for DE. To the best of our knowledge, crossovers type 2 and 3, have not been employed in DE yet. We also develop crossover type 4 for the first time in this paper. The proposed crossover operators are as follows:

- Crossover type 1, Storn and Price [49]:

$$U_{i,j,G+1} = \begin{cases} \mathbf{v}_{i,j,G+1} & \text{if rand}(j) \le \text{CR or } j = \text{randn}(i), \\ \mathbf{x}_{i,j,G+1} & \text{if rand}(j) > \text{CR and } j \ne \text{randn}(i), \end{cases}$$

where rand(j) is the jth evaluation of a random number uniformly distributed in the range of [0, 1], and randn(i) is a randomly chosen index from the set $\{1, 2, \ldots, N\}$. CR$\in$[0, 1] is a crossover constant rate that controls the diversity of the population. The more the value of CR, the less the influence of the parent will be.

- Crossover type 2, one-point crossover:
  In this operator, one point is randomly selected for dividing a vector and its mutant to two parts, and then, the two random opposite parts of them are combined together to generate a new vector.
- Crossover type 3, two-point crossover:
  At first, two crossover points are randomly selected in a vector. The digits between randomly selected points in the vector are inherited from the mutant of the vector and the other digits are placed in the order they appeared in the original vector.

- Crossover type 4, new crossover:

  As one can see in the last three types of crossover operators, there is no control on the rate of changes in the original vector. For example, in two point crossover, the two points may be selected after the first digit and before the last digit, respectively, and hence, the new vector inherits most of its digits from the mutant vector. Because of these uncontrollable types of operators, we propose a method to manage the changes will occur in the new vector by ourselves. In crossover type 1, we have, CR∈[0, 1], which is a constant crossover rate that controls the diversity of the population. We use this term and multiple it with the n, number of customers or number of digits in vector. This product gives us a number between 1 and n. the more the CR, the more the product will be. Therefore, we can control the number of digits that must change in new vector. By the example shown in Fig. 5, the procedure of this type of crossover is explained clearly.

In this example, we have 15 customers, eight plants, and the CR supposed to be two numbers 0.3 and 0.7, to consider both high and low CR, and its effect on the changes in the produced vectors by the crossover. The products of these parameters will be 4.5 and 10.5. We round these numbers to 5 and 11. Thus, the numbers of digits that must be inserted from the mutant vector to new vectors are 5 and 11, respectively. For the former, the crossover point can be selected between 1 and 10, and for the latter, the point can be selected between 1 and 4. The 5 and 11 digits after the points must be inherited from mutant vector, respectively, to create two new vectors for two constant crossover rates.

5.5 Selection operator

To generate the new individual for the next generation, selection operation is performed between each individual and its corresponding trial vector by the following greedy selection criterion:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} \text{ if } f\left(U_{i,G+1}\right) < f\left(X_{i,G}\right), \\ X_{i,G} \qquad\qquad \text{otherwise}, \end{cases}$$

where f is the objective function, and $X_{i,G+1}$ is the individual of the new population.

6 Experimental design

6.1 Taguchi parameter design

Because of the dependency of the metaheuristic algorithms on the correct selection of parameters and operators, we are to study the behavior of the different parameters and operators of the proposed algorithms. In order to calibrate the parameters and operators of algorithms, there are several ways to statistically design experimental investigation. One of the present methods that often used in the most researches is the full factorial design method. But, because it examines all possible combinations of factors, it does not cost-effective way when the number of factors increases meaningfully. The more the number of required experiments for tuning, the more the time and cost will be spent. As it would be explained clearly later, for the GA, there are 28 test problems, four 3-level factors, and one 5-level factor in our case that each of which should be run three times. Therefore, the total number of running the problem in GA is $28\times3^4\times5^1\times3$, which is equal to 34,020. In the DE, there are 28 test problems, three 3-level factors, one 18-level factor, and one 4-level factor in our case that each of which should be run three times. Hence, the total number of running the problem in DE is $28\times3^3\times18^1\times4^1\times3$, which is equal to 163,296.

In the related works, to be economic, several experimental designs have been proposed to decrease the number of experiments. Among several experimental design techniques, the Taguchi experimental design method has been successfully employed for a systematic approach for optimization [45, 56, 57]. In Taguchi method, the orthogonal arrays are used to analyze a large number of decision variables with a small number of experiments. Taguchi separates the factors into two main groups: controllable and noise factors. Controllable factors will be placed in inner orthogonal array and noise factors in the outer orthogonal array. Due to unpractical and often impossible omission of the noise factors, the Taguchi tends to both minimize the impact of noise and also find the best level of the influential controllable factors on the basis of robustness. Moreover, Taguchi determines the relative importance of each factor with respect to its main impacts on the performance of the algorithm.

Taguchi has created a transformation of the repetition data to another value, which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio, which explains why this type of parameter design is called robust design [45]. Here, the term “signal” denotes the desirable value (mean response variable), and “noise” denotes the

Original vector	2.4	5.1	1.1	1.8	8.7	4.3	-2.5	2	7.3	2.4	4.5	4.9	3.1	2.2	0.8
Mutant vector	0.2	1.7	0.1	20.4	-9.5	-0.5	1.6	16.3	3.6	6.79	2.9	3.1	7.8	9.9	4.9
New vector if *CR*=0.3	2.4	5.1	1.1	1.8	8.7	4.3	-2.5	2	3.6	6.79	2.9	3.1	7.8	2.2	0.8
New vector if *CR*=0.7	2.4	5.1	1.1	20.4	-9.5	-0.5	1.6	16.3	3.6	6.79	2.9	3.1	7.8	9.9	0.8

Fig. 5 Examples of operating new crossover

undesirable value (standard deviation). Thus, the S/N ratio indicates the amount of variation present in the response variable. The aim is to maximize the signal-to-noise ratio. In the Taguchi method, the S/N ratio of the minimization objectives is as such [39]:

$$\mathrm{S/N\,ratio} = -10\log_{10}(\mathrm{objective\,function})^2$$

In this paper, we applied the Taguchi method in parameter setting to achieve better robustness of the proposed algorithms. The control factors of presented GA are population size, crossover percentage, mutation probability, type of crossover, and type of mutation, and for the proposed DE, the factors are population size, crossover CR, scale factor (F), type of mutation, and type of crossover. Levels of these factors are illustrated in Table 2.

For the GA, to select the appropriate orthogonal array, it is necessary to calculate the total degree of freedom. The proper array should contain a degree of freedom for the total mean, two degrees of freedom for each factor with three levels ($2\times4=8$), and four degrees of freedom for the only factor with five levels. Thus, the sum of the required degrees of freedom is $1+2\times4+4=11$. Therefore, the appropriate array must have at least 11 rows.

Table 3 The modified orthogonal array L_{18} for GA

Trial	A	B	C	D	E
1	1	1	1	1	1
2	1	1	2	2	2
3	1	2	1	3	3
4	1	2	3	1	4
5	1	3	2	3	5
6	1	3	3	2	5
7	2	1	1	3	5
8	2	1	3	1	5
9	2	2	2	2	1
10	2	2	3	3	2
11	2	3	1	2	4
12	2	3	2	1	3
13	3	1	2	3	4
14	3	1	3	2	3
15	3	2	1	2	5
16	3	2	2	1	5
17	3	3	1	1	2
18	3	3	3	3	1

The selected orthogonal array should be able to accommodate the factor level combinations in the experiment. From

Table 2 Factors and their levels

Factors	GA symbols	DE symbols	GA levels	DE levels
Population size	A	A	A(1)—40	A(1)—85
			A(2)—45	A(2)—90
			A(3)—50	A(3)—95
Crossover percentage	B	–	B(1)—55 %	–
			B(2)—65 %	–
			B(3)—75 %	–
Scale factor (F)	–	B	–	B(1)—0.7
			–	B(2)—0.8
			–	B(3)—0.9
Mutation probability	C	–	C(1)—0.25	–
			C(2)—0.3	–
			C(3)—0.35	–
Crossover constant rate (CR)	–	C	–	C(1)—0.4
			–	C(2)—0.45
			–	C(3)—0.5
Type of crossover	D	D	D(1)—one-point crossover	D(1)—crossover type 1
			D(2)—two-point crossover	D(2)—crossover type 2
			D(3)—uniform crossover	D(3)—crossover type 3
				D(4)—crossover type 4
Type of mutation	E	E	E(1)—swap E(2)—big swap	E(1) to E(18): mutation types 1–18
			E(3)—Inversion	
			E(4)—Displacement	
			E(5)—perturbation mutation	

Table 4 The modified orthogonal array L_{80} for DE

Trial	A	B	C	D	E	Trial	A	B	C	D	E	Trial	A	B	C	D	E	Trial	A	B	C	D	E
1	1	1	1	1	1	21	2	1	2	1	6	41	3	1	1	3	14	61	3	1	1	1	18
2	1	1	1	4	2	22	2	1	2	1	7	42	3	1	2	2	15	62	3	1	1	3	17
3	1	1	2	3	3	23	2	1	3	1	10	43	3	1	3	2	11	63	3	1	2	4	18
4	1	1	3	4	4	24	2	1	3	2	8	44	3	1	3	3	12	64	3	1	3	2	17
5	1	1	3	3	5	25	2	1	3	4	9	45	3	1	3	4	13	65	3	1	3	2	16
6	1	2	1	2	6	26	2	2	1	4	3	46	3	2	1	3	18	66	3	2	1	1	15
7	1	2	1	2	7	27	2	2	2	2	1	47	3	2	2	2	18	67	3	2	2	4	14
8	1	2	3	1	8	28	2	2	2	3	2	48	3	2	2	4	17	68	3	2	3	3	13
9	1	2	3	3	9	29	2	2	3	4	5	49	3	2	3	1	16	69	3	2	3	4	12
10	1	2	3	2	10	30	2	2	3	3	4	50	3	2	3	1	17	70	3	2	3	1	11
11	1	3	1	4	11	31	2	3	1	4	17	51	3	3	1	2	4	71	3	3	1	3	10
12	1	3	2	1	12	32	2	3	1	4	16	52	3	3	2	1	5	72	3	3	2	2	9
13	1	3	2	2	13	33	2	3	3	1	17	53	3	3	3	2	2	73	3	3	2	4	8
14	1	3	3	1	14	34	2	3	3	3	18	54	3	3	3	3	1	74	3	3	3	3	7
15	1	3	3	4	15	35	2	3	3	2	18	55	3	3	3	1	3	75	3	3	3	3	6
16	1	3	2	3	16	36	2	3	1	1	13	56	3	3	1	1	9	76	3	3	1	2	5
17	1	3	2	3	17	37	2	3	1	2	12	57	3	3	1	3	8	77	3	3	2	1	4
18	1	3	3	1	18	38	2	3	2	3	11	58	3	3	2	4	10	78	3	3	3	2	3
19	1	3	3	2	17	39	2	3	3	3	15	59	3	3	3	4	7	79	3	3	3	1	2
20	1	3	3	4	18	40	2	3	3	2	14	60	3	3	3	4	6	80	3	3	3	4	1

standard table of orthogonal arrays, the L_{18} is selected as the fittest orthogonal array design, which fulfills our all minimum requirements. However, this orthogonal array still entails some modifications to adapt itself to our experimental design. The modified orthogonal array L_{18} is presented in Table 3, where control factors are assigned to the columns of the orthogonal array and the corresponding integers in these columns indicate the actual levels of these factors. The experiments on the parameters of the GA are based on the L_{18} orthogonal array; therefore, 18 different combinations of control factors (trials) are considered.

In the DE, two degrees of freedom for each factor with three levels (2×3=6), 17 degrees of freedom for the only factor with 18 levels and three degrees of freedom for the only factor with four levels. Thus, the sum of the required degrees of freedom is 1+2×3+17+3=27. Therefore, the appropriate array must have at least 27 rows. Considering this, L_{80} (4^{10}, 20^{1}) is an appropriate array that satisfies these conditions. As there are three factors with three levels and a factor with 18 levels, and this scheme offers factors with four and 20 levels, we should adjust this array to the problem by means of adjustment techniques. Using the dummy level technique, we convert three 4-level column into three 3-level column and a 20-level column into an 18-level column. To assign the three-level factor to the four-level column from the orthogonal array L80 (4^{10}, 20^{1}), one of these levels are required to be replicated twice. Fourth level is chosen to be replicated twice. Similarly to assign the 18-level factor to the 20-level column, two levels are required to be

Table 5 Test problems characteristics

Problem size	Total Demand	Problem type	Range of variable costs				Range of fixed costs			
			a^l	a^l–b^l	α	β	a^l	a^l–b^l	α	β
10×10	10,000	A	$U(3, 7)$	$U(0, 1)$	$U(0.25, 1)$	$U(0.25, 1)$	$U(1{,}000, 4{,}000)$	$U(0, 500)$	$U(100, 500)$	$U(100, 500)$
10×20	15,000	B	$U(3, 7)$	$U(0, 1)$	$U(0.25, 1)$	$U(0.25, 1)$	$U(2{,}000, 8{,}000)$	$U(0, 1{,}000)$	$U(200, 1{,}000)$	$U(200, 1{,}000)$
15×15	15,000	C	$U(3, 7)$	$U(0, 1)$	$U(0.25, 1)$	$U(0.25, 1)$	$U(4{,}000, 16{,}000)$	$U(0, 2{,}000)$	$U(400, 2{,}000)$	$U(400, 2{,}000)$
10×30	15,000	D	$U(3, 7)$	$U(0, 1)$	$U(0.25, 1)$	$U(0.25, 1)$	$U(8{,}000, 32{,}000)$	$U(0, 4{,}000)$	$U(800, 4{,}000)$	$U(800, 4{,}000)$
50×50	50,000									
30×100	30,000									
50×200	50,000									

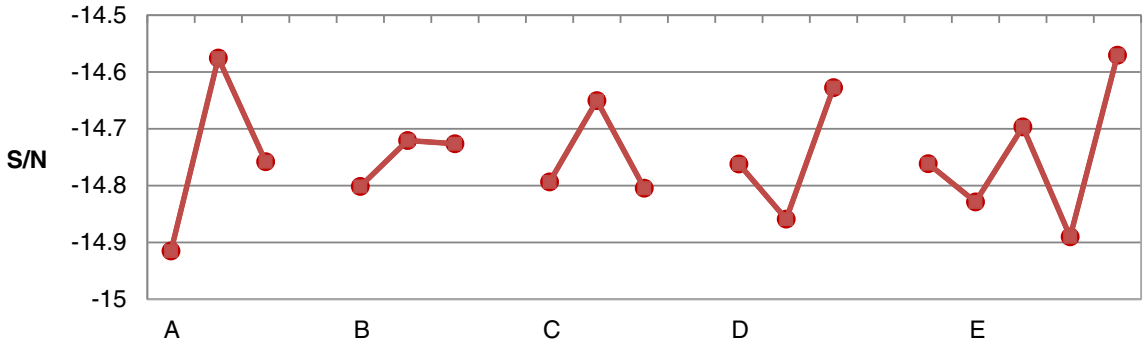

Fig. 6 Mean S/N ratio plot for each level of the factors in GA

replicated twice. Nineteenth and 20th levels are chosen to be replicated twice. Additionally, since there are three 3-level factors and one 4-level factor in our work, according to Taguchi experimental design procedure, we can keep six columns empty. It is essential to notice that, after applying these techniques, the obtained array remains orthogonal. Furthermore, the accuracy of these levels that are replicated twice is twice the accuracy of the other level. The modified orthogonal array L_{80} is presented in Table 4.

6.2 Data generation

In order to present the efficiency of the proposed algorithms for solving the problem, a plan is utilized to generate test data. Following Hajiaghaei-Keshteli [25], the data required for a problem consists of the number of DCs and customers, total demand, and range of variable costs and fixed costs. For running the algorithms, 28 problem sets were generated at random in which seven size of problem are implemented for experimental study. The problem size is determined by the number of DCs and customers. The lower and upper bounds of variable costs are 2 and 9, such that a^l, a^u–a^l, α and β are made from a uniform distribution of $U(3, 7)$, $U(0, 1)$, $U(0.25, 1)$, and $U(0.25, 1)$. Within each problem size, four problem types A, B, C, and D are considered. For each problem size, problem types are different in range of fuzzy fixed cost numbers, which increases according to the alphabetic order of the problem types. Variable costs are uniformly generated in the small interval, while the lower and upper bounds of fixed costs are generated in larger interval. The problem sizes, types, DCs/customers, and fixed costs ranges are shown in Table 5.

6.3 Parameter tuning

Twenty-eight test problems, with different sizes and specifications, are generated and solved to evaluate the performance of the presented algorithms. The experiments on the GA and DE were based on the L_{18} and L_{80} orthogonal arrays; therefore, 18 different combinations of control factors were considered. To yield more reliable information and due to having stochastic nature of the algorithms, we tackle each test problem three times. The algorithms are coded and run via C++ language. After obtaining the results of the test problems in different trial, results of each trial are transformed into S/N ratio. The S/N ratios of trials are averaged in each level, and its value is plotted against each control factor in Fig. 6. In GA, better robustness of the GA is happened when parameters of factors A, B, C, D, and E are obviously 2, 2, 2, 3, and 5, respectively, as depicted in Fig. 6. Besides, for DE, as illustrated in Fig. 7, all the best parameters are defined as 2, 2, 2, 4, and 10, respectively, according to their alphabetical order.

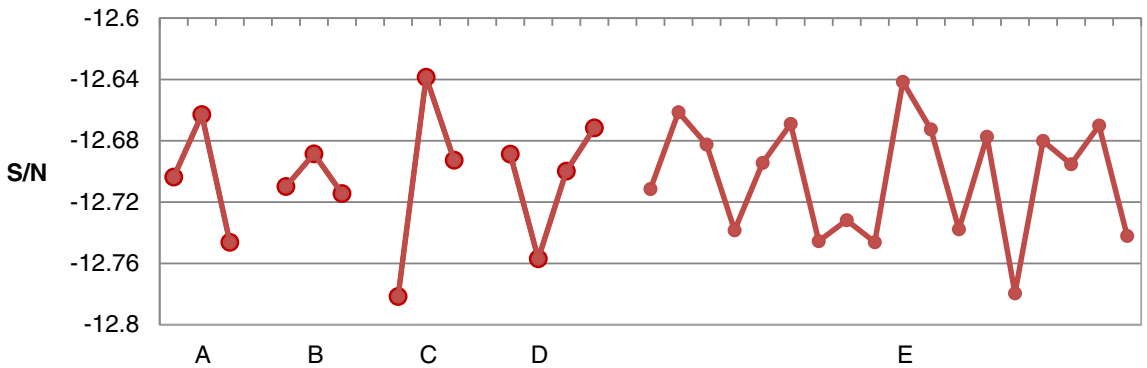

Fig. 7 Mean S/N ratio plot for each level of the factors in DE

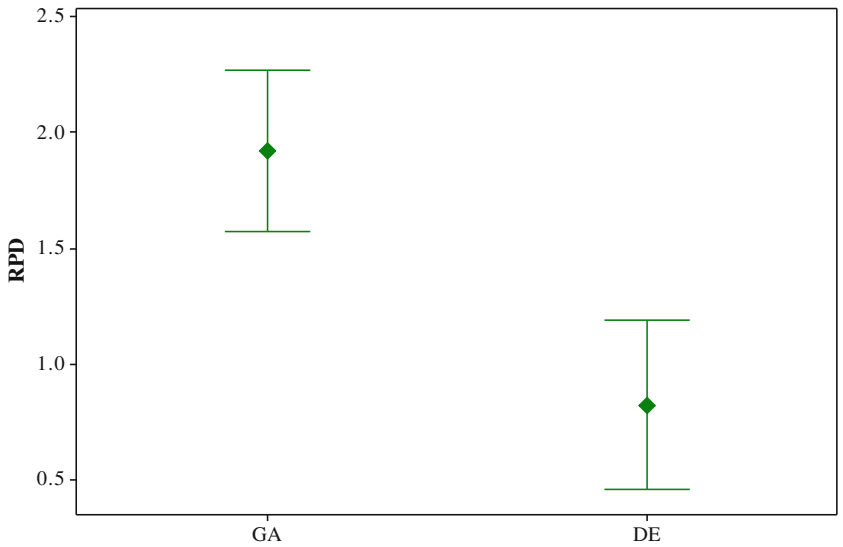

Fig. 8 Means plot and LSD intervals for the algorithms

6.4 Experimental results

We set searching time to be identical for both algorithms which is equal to $0.25 \times n \times m$ milliseconds. Hence, this criterion is affected by both n and m. The more the number of DCs or number of customers, the more the rise of searching time increases. Twenty instances for each of the seven problem sizes, i.e., totally 140 instances are generated which are different from the ones used for calibration to avoid bias in the results. Each instance is run three times. Because the scale of objective functions in each instance is different, they could not be used directly. To solve this problem, the relative percentage deviation (RPD) is used for each instance. The RPD is obtained by the following formula:

$$\mathrm{RPD} = \frac{\mathrm{Alg}_{\mathrm{sol}} - \mathrm{Min}_{\mathrm{sol}}}{\mathrm{Min}_{\mathrm{sol}}} \times 100$$

where $\mathrm{Alg}_{\mathrm{sol}}$ and $\mathrm{Min}_{\mathrm{sol}}$ are the obtained objective value and minimum objective value found from both proposed algorithms for each instance, respectively. Considering 20 instances for each of the seven problem sizes, for both algorithms, the instances have been run three times, and hence, using the RPD, we deal with 60 data for each algorithm. The averages of these data for each algorithm and each instance are shown in Fig. 9.

In order to verify the statistical validity of the results, we have performed an analysis of variance (ANOVA) to accurately analyze the results.

The point that can be concluded from the results is that there is a clear statistically meaningful difference between performances of the algorithms. The means plot and LSD intervals (at the 95 % confidence level) for two algorithms are shown in Fig. 8. Since, we are to appraise the robustness of the algorithms in different circumstances, the effects of the problem sizes on the performance of both algorithms are analyzed. The reciprocal between the capability of the algorithms and the size of problems is illustrated in Fig. 9. As it is shown, DE exhibits robust performance; meanwhile, the problem size increases. It

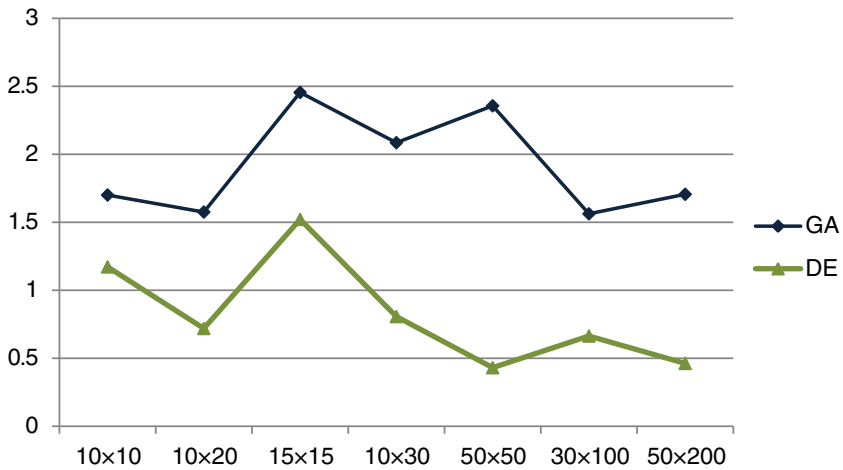

Fig. 9 Means plot for the interaction between each algorithm and problem size

also shows remarkable performance improvements of DE in large size problems versus GA.

7 Conclusion and future works

The proposed model minimizes the total cost with selecting some potential places as distribution centers in order to supply demands of all customers in two stages of supply chain network and fuzzy environment. In order to solve the given problem, two algorithms (GA and DE) are presented. Various operators were employed in the algorithms, in which some of them are newly presented in this paper. In order to tune the parameters of the proposed algorithms, the Taguchi parameter design method was used. Applying this method decreases the original massive experiment to only 18 levels (due to Taguchi standard L18 orthogonal array) for the GA and 80 levels for the DE. Computational results showed the superiority of the newly proposed operators in comparison with the existing ones. They also demonstrate the efficiency of DE to solve the problem and superior performance over GA in all problem sizes. There still exist rich opportunities for researchers to further study in this area.

The future work is to extend our approach to the case of generalized trapezoidal fuzzy costs for solving real-life distribution problems. Another work is to represents all the parameters (cost, availability, and demand) by fuzzy numbers. In addition, single-echelon single commodity, and multiechelon multicommodities are the areas where the algorithms can be employed. Another direction is to work on other metaheuristics, such as ant colony optimization, scatter search and artificial immune algorithms.

Acknowledgment This study was partially supported by Islamic Azad University, Masjed Soleyman Branch. The authors are grateful for this financial support.

References

1. Abbass HA, Sarker R, Newton C (2001) PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the IEEE congress on evolutionary computation, vol 2, pp 971–978
2. Adlakha V, Kowalski K (2001) A heuristic method for more-for-less in distribution related problems. Int J Math Educ Sci Technol 26:61–71
3. Antony Arokia Durai Raj K, Rajendran C (2011) A genetic algorithm for solving the fixed-charge transportation model: two-stage problem. Comput Oper Res 39(9):2016–2032
4. Arsham H (1992) Postoptimality analyses of the transportation problem. J Oper Res Soc 43:121–159
5. Arsham H, Khan AB (1989) A simplex type algorithm for general transportation problems: an alternative to stepping-stone. J Oper Res Soc 40:581–670
6. Balaji N, Jawahar N (2010) A simulated annealing algorithm for a two-stage fixed charge distribution problem of a supply chain. Int J Oper Res 7:192–215
7. Bellman R, Zadeh L (1970) Decision-making in a fuzzy environment. Manag Sci 17(4):141–164
8. Bezdek JC (1993) Fuzzy models—what are they, and why? IEEE Trans Fuzzy Sys 1:1–9
9. Bit AK (2005) Fuzzy programming with hyperbolic membership functions for multi-objective capacitated solid transportation problem. J Fuzzy Math 13:373–385
10. Bit AK, Biswal MP, Alam SS (1993) Fuzzy programming approach to multi-objective solid transportation problem. Fuzzy Sets Syst 57: 183–194
11. Bit AK, Biswal MP, Alam SS (1993) An additive fuzzy programming model for multi-objective transportation problem. Fuzzy Sets Syst 57:313–319
12. Borisovsky P, Dolgui A, Eremeev A (2009) Genetic algorithms for a supply management problem: MIP-recombination vs greedy decoder. Eur J Oper Res 195:770–779
13. Brenner U (2008) A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. Oper Res Lett 36:408–413
14. Chanas S, Kuchta D (1996) A concept of the optimal solution of the transportation problem with fuzzy cost coefficients. Fuzzy Sets Syst 82:299–305
15. Charnes A, Copper WW (1954) The stepping-stone method for explaining linear programming calculation in transportation problem. Manag Sci 1:49–69
16. Chen IJ, Paulraj A (2004) Understanding supply chain management: critical research and theoretical framework. Int J Prod Res 42:131–163
17. Costa A, Celano G, Fichera S, Trovato E (2010) A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms. Comput Ind Eng 59:986–999
18. Dantzig GB (1963) Linear programming and extensions. Princeton University Press, Princeton
19. Feoktistov V, Janaqi S (2004) Generalization of the strategies in differential evolution. In the Proceedings of the 18th International parallel and distributed processing symposium (IPDPS'04) 165–170
20. Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York
21. Gen M, Cheng R (2000) Genetic algorithms and engineering optimization. Wiley, New York
22. Gen M, Li YZ (1998) Solving multi-objective transportation problem by spanning tree-based genetic algorithm. In: Parmee I (ed) Adaptive computing in design and manufacture. Springer, Berlin, pp 95–108
23. Gen M, Syarif A (2003) Hybrid genetic algorithm for production/distribution system in supply chain. Int J Smart Eng Syst Des 5:289–298
24. Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by benders decomposition. Manag Sci 20:822–844
25. Hajiaghaei-Keshteli M (2011) The allocation of customers to potential distribution centers in supply chain network; GA and AIA approaches. Int J Appl Soft Comput 11:2069–2078
26. Hajiaghaei-Keshteli M, Molla-Alizadeh-Zavardehi S, Tavakkoli-moghaddam R (2010) Addressing a nonlinear fixed-charge transportation problem using a spanning tree-based genetic algorithm. Int J Comput Ind Eng 59:259–271
27. Hajiaghaei-Keshteli M, Sajadifar SM (2010) Deriving the cost function for a class of three-echelon inventory system with N-retailers and one-for-one ordering policy. Int J Adv Manuf Technol 50:343–351
28. Hindi KS, Basta T, Pienkosz K (1998) Efficient solution of a multi-commodity, two-stage distribution problem with constraints on assignment of customers to distribution centers. Int Transp Oper Res 5(6):519–527
29. Hitchcock FL (1941) The distribution of a product from several sources to numerous localities. J Math Phys 20:224–230

30. Jawahar N, Balaji AN (2009) A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge. Eur J Oper Res 194:496–537
31. Kapur JN, Kesavan HK (1992) Entropy optimization principles with applications. Academic, San Diego
32. Kaufman A (1975) Introduction to the theory of fuzzy subsets. Academic, New York
33. Ko M, Tiwari A, Mehnen J (2010) A review of soft computing applications in supply chain management. Appl Soft Comput 10: 661–674
34. Li YZ, Gen M (1997) Spanning tree-based genetic algorithm for bicriteria transportation problem with fuzzy coefficients. Aust J Intell Inf Process Syst 4:220–229
35. Li YZ, Gen M, Ida K (1998) Improved genetic algorithm for solving multi objective solid transportation problem with fuzzy number. Japan J Fuzzy Theory Syst 4(3):220–229
36. Li L, Lai KK (2000) A fuzzy approach to the multi-objective transportation problem. Comput Oper Res 27:43–57
37. Maleki HR (2002) Ranking function and their application to fuzzy linear programming. Far East J Math Sci (FJMS) 4:283–301
38. Melo MT, Nickel S, Saldanha-da-Gama F (2009) Facility location and supply chain management—a review. Eur J Oper Res 196:401–412
39. Molla-Alizadeh-Zavardehi S, Hajiaghaei-Keshteli M, Tavakkoli-moghaddam R (2011) Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Prüfer number representation. Expert Syst Appl 38:10462–10474
40. Murata T, Ishibuchi H (1994) Performance evaluation of genetic algorithms for flowshop scheduling problems. International Conference on Evolutionary Computation, pp 812–817
41. Nearchou AC (2007) Balancing large assembly lines by a new heuristic based on differential evolution method. Int J Adv Manuf Technol 34:1016–1029
42. Nearchou AC (2008) Multi-objective balancing of assembly lines by population heuristics. Int J Prod Res 46:2275–2297
43. Ojha D, Mondal SK, Maiti M (2009) An entropy based solid transportation problem for general fuzzy costs and time with fuzzy equality. Math Comput Model 50:166–178
44. Omar MS, Samir AA (2003) A parametric study on transportation problem under fuzzy environment. J Fuzzy Math 11:115–124
45. Phadke MS (1989) Quality engineering using robust design. Prentice-Hall, Englewood Cliffs
46. Qian B, Wang L, Huang D, Wang X (2008) Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. Int J Adv Manuf Technol 35:1014–1027
47. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. Proc IEEE Congr Evol Comput 2:1785–1791
48. Samanta B, Roy TK (2005) Multi-objective entropy transportation model with trapezoidal fuzzy number penalties, sources and destination. J Transp Eng 131:419–428
49. Storn R, Price K (1997) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. J Glob Optim 11:341–359
50. Syarif A, Gen M (2003) Solving exclusionary side constrained transportation problem by using a hybrid spanning tree-based genetic algorithm. Int J Intell Manuf 14:389–399
51. Syarif A, Gen M (2003) Double spanning tree-based genetic algorithm for two stage transportation problem. Int J Knowl-Based Eng Syst 7(4):214–221
52. Syarif A, Yun YS, Gen M (2002) Study on multi-stage logistics chain network: a spanning tree based genetic algorithm. Comput Ind Eng 43:299–314
53. Thomas DJ, Griffin PM (1996) Coordinated supply chain management. Eur J Oper Res 94:1–15
54. Xie F, Jia R (2012) Nonlinear fixed charge transportation problem by minimum cost flow-based genetic algorithm. Comput Ind Eng 63: 763–778
55. Yang SH, Natarajan U, Sekar M, Palani S (2010) Prediction of surface roughness in turning operations by computer vision using neural network trained by differential evolution algorithm. Int J Adv Manuf Technol 51:965–971
56. Yildiz A (2012) Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. Int J Adv Manuf Technol 64:55-61
57. Yildiz AR, Solanki KN (2013) Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach. Int J Adv Manuf Technol 59(1–4):367–376
58. Zadeh LA (1965) Fuzzy sets. Inf Control 8:338–353
59. Zhou G, Min H, Gen M (2002) The balanced allocation of customers to multiple distribution centers in the supply chain networks, a genetic algorithm approach. Comput Ind Eng 43: 251–261