Int. J. Mach. Learn. & Cyber. (2013) 4:65–74
DOI 10.1007/s13042-012-0073-0

ORIGINAL ARTICLE

Capacitated two-stage facility location problem with fuzzy costs and demands

Shuming Wang · Junzo Watada

Received: 28 May 2011 / Accepted: 18 January 2012 / Published online: 15 February 2012

Abstract In this study, we develop a two-stage capacitated facility location model with fuzzy costs and demands. The proposed model is a task of 0–1 integer two-stage fuzzy programming problem. In order to solve the problem, we first apply an approximation approach to estimate the objective function (with fuzzy random parameters) and prove the convergence of the approach. Then, we design a hybrid algorithm which integrates the approximation approach, neural network and particle swarm optimization, to solve the proposed facility location problem. Finally, a numerical example is provided to test the hybrid algorithm.

Keywords Location · Two-stage fuzzy programming · Fuzzy variable · Neural network · Particle swarm optimization

1 Introduction

Since the original study by Cooper [9], facility location problems (FLP) as a crucial and generic engineering optimization model have being attracting an increasing number of people. The key issue of FLP is to find the optimal sizes to open facilities among a given set of potential sites to meet the objective of profit maximization or cost minimization.

S. Wang (✉) · J. Watada
Graduate School of Information, Production and Systems, Waseda University, 2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka 808-0135, Japan
e-mail: smwangips@gmail.com

J. Watada
e-mail: junzow@osb.att.ne.jp

Regarding FLPs with deterministic parameters, capacitated FLP in which the capacities of facilities are limited, was originally discussed by Murtagh and Niwattisyawong [30] which is considered as one of studies of the most importance in the filed of FLP, and later on, more and more studies along this direction (capacitated FLP) have been reported in the literature [1, 4, 14]. Since it was proved by Megiddo and Supowit [29] that FLP is NP-hard, a series of analytical methods and programming techniques as well as heuristic algorithms have been developed to solve the FLPs. For instance, Love [27] discussed one-dimensional facility location–allocation problem using dynamic programming. Gong et al. [15] designed a hybrid evolutionary method for solving obstacle location–allocation problem. Lozano et al. [28] discussed the application of Kohonen maps to solve a class of location–allocation problems. Ernst and Krishnamoorthy [14] combined the simulated annealing and random descent method.

In real applications, many parameters of FLP, such as demands of clients, the costs of operating the facilities, may be of uncertainties, e.g., randomness and fuzziness. These distinct uncertainties yields the stochastic FLP and fuzzy FLP, respectively. For stochastic scenarios, readers may refer to [5, 24–26]. Due to the development of fuzzy theory [12, 13, 21, 31, 40, 41], a number of pieces of research brought this tool into the FLP. Darzentas [11] discussed various facility location problems by fuzzy logic methods. Bhattacharya et al. [2] considered facilities located under multiple fuzzy criteria, and proposed a fuzzy goal programming approach to deal with the problem. Ishii et al. [16] developed a location model considering the satisfaction degree with respect to the distance from the facility for each customer and preference of the site in an urban area. In the above-mentioned studies, the parameters of FLP are deterministic, and fuzzy theory are just used to

solve classical mathematical programming effectively, the problems were static ones in nature. Apart from those researches, Zhou and Liu [42] modeled three types of capacitated location–allocation problem with fuzzy demands according to different decision criteria. Wen and Iwamura [39] presented an α-cost FLP with fuzzy demands under Hurwicz criterion. In these two papers, the costs for establishing and operating the facilities and the size of the facilities are all assumed fixed, and since the potential region where the location are to be chosen are assumed to be continuous, the proposed FLP in [39, 42] are both continuous type of fuzzy programming problems.

This paper addresses a more realistic fuzzy capacitated FLP in which both the demands of clients and the variable costs of facilities can be fuzzy, and the decision is a 0-1 integer vector which consists of the optimal location and size of the new facilities. What is more, when process the decision-making in FLP, we assume that the decisions are made in two stages so as to maximize expected total profit. The first-stage decision, location decision, is made before the values of fuzzy parameters are realized, and the second-stage decisions, distribution pattern can be taken after the realization of the fuzzy parameters are observed. The optimal value of the problem depends on the realization of the fuzzy parameters and the first-stage decisions. It is a dynamic process.

The rest of the paper is organized as follows. Section 2 recalls some basic concepts of fuzzy variable. In Sect. 3, the model is formulated. Section 4 discusses an approximation method to the expected objective value of second-stage programming, and proves the convergence of the approximation. Then, a hybrid algorithm is designed in which the approximation approach, neural network (NN) and PSO are fused to solve the proposed FLP. A numerical example is provided in Sect. 5 to illustrate the effectiveness of the hybrid algorithm. Section 6 draws the conclusions.

2 Fuzzy variable

Given a universe Γ, let Pos be a possibility measure defined on the power set $\mathcal{P}(\Gamma)$ of Γ. Then, triplet $(\Gamma, \mathcal{P}(\Gamma), \mathrm{Pos})$ is called a *possibility space*. A function $\xi = (\xi_1, \xi_2, \ldots, \xi_n) : \Gamma \to \Re^n$ is said to be an n-ary fuzzy vector. As $n = 1$, ξ is called a fuzzy variable. The function

$$\begin{aligned}\mu_\xi(\mathbf{t}) &= \mathrm{Pos}\{\gamma \in \Gamma | \xi(\gamma) = \mathbf{t}\} \\ &= \min_{1 \le i \le n} \mathrm{Pos}\{\gamma \in \Gamma | \xi_i(\gamma) = t_i\}\end{aligned} \tag{1}$$

for any $\mathbf{t} = (t_1, \ldots, t_n) \in \Re^n$ is said to be the possibility distribution of ξ, or the joint possibility distribution of $\xi_i, i = 1, 2, \ldots, n$.

For fuzzy variable ξ with possibility distribution μ_ξ, the possibility, necessity and credibility of event $\{\xi \le r\}$ can be given respectively by

$$\begin{aligned}\mathrm{Pos}\{\xi \le r\} &= \sup_{t \le r} \mu_\xi(t), \\ \mathrm{Nec}\{\xi \le r\} &= 1 - \sup_{t > r} \mu_\xi(t), \\ \mathrm{Cr}\{\xi \le r\} &= \frac{1}{2}[\mathrm{Pos}\{\xi \le r\} + \mathrm{Nec}\{\xi \le r\}].\end{aligned} \tag{2}$$

Based on credibility measure, the expected value of a fuzzy variable is defined below [21]:

Definition 1 Let ξ be a fuzzy variable defined on a possibility space $(\Gamma, \mathcal{P}(\Gamma), \mathrm{Pos})$. The expected value of ξ is defined as

$$E[\xi] = \int_0^\infty \mathrm{Cr}\{\gamma \mid \xi(\gamma) \ge r\}\mathrm{d}r - \int_{-\infty}^0 \mathrm{Cr}\{\gamma \mid \xi(\gamma) \le r\}\mathrm{d}r \tag{3}$$

provided that one of the two integrals is finite.

Example 1 Let ξ be a triangular fuzzy variable (2, 3, 4). Calculate the expected value $E[\xi]$.

Recall the possibility distribution of triangular fuzzy variable $\xi = (2, 3, 4)$ is

$$\mu_\xi(t) = \begin{cases} t - 2, & \text{if } 2 \le t < 3 \\ 4 - t, & \text{if } 3 \le t < 4 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

From (2), for any $r \ge 0$, we can compute

$$\begin{aligned}\mathrm{Cr}\{\xi \ge r\} &= \frac{1}{2}\left(\sup_{t \ge r} \mu_\xi(t) + 1 - \sup_{t < r} \mu_\xi(t)\right) \\ &= \begin{cases} 1, & \text{if } r \le 2 \\ (4 - r)/2, & \text{if } 2 < r \le 4 \\ 0, & \text{otherwise.} \end{cases}\end{aligned}$$

For any $r < 0$, we have

$$\mathrm{Cr}\{\xi \le r\} = \frac{1}{2}\left(\sup_{t \le r} \mu_\xi(t) + 1 - \sup_{t > r} \mu_\xi(t)\right) \equiv 0.$$

It follows from Definition 1 that

$$E[\xi] = \int_0^\infty \mathrm{Cr}\{\xi \ge r\}\mathrm{d}r = 2 + \int_2^4 \frac{4 - r}{2}\mathrm{d}r = 3.$$

Particularly, for a discrete fuzzy variable ξ with the following possibility distribution:

$$\mu_\xi(t) = \begin{cases} \mu_1 & \text{if } t = a_1 \\ \mu_2 & \text{if } t = a_2 \\ \vdots & \\ \mu_n & \text{if } t = a_n \\ 0 & \text{otherwise.} \end{cases}$$

Without any loss of generality, we assume that $a_1 \le a_2 \le \cdots \le a_n$, i.e., a_i is the ith smallest outcome

value of ξ. Then the expected value defined by (3) reduces to the following form [21]:

$$E[\xi] = \sum_{i=1}^{n} p_i a_i, \tag{5}$$

where the weights $p_i's$ are determined by

$$p_i = \frac{1}{2}\left(\max_{j=1}^{i} \mu_j - \max_{j=0}^{i-1} \mu_j\right) + \frac{1}{2}\left(\max_{j=i}^{n} \mu_j - \max_{j=i+1}^{n+1} \mu_j\right) \tag{6}$$

$(\mu_0 = 0, \mu_{n+1} = 0)$ for $i = 1, \ldots, n$, and satisfy the following constraints

$$p_i \geq 0, \quad \text{and} \quad \sum_{i=1}^{n} p_i = \max_{i=1}^{n} \mu_i = 1.$$

Example 2 Let ξ be a fuzzy variable with the following possibility distribution

$$\mu_\xi(t) = \begin{cases} 0.7 & \text{if } t = 1 \\ 1 & \text{if } t = 3 \\ 0.8 & \text{if } t = 5 \\ 0 & \text{otherwise.} \end{cases}$$

Calculate the expected value of ξ.

By (6), we have

$$p_1 = \frac{1}{2}(0.7 - 0) + \frac{1}{2}(\max\{0.7, 1, 0.8\} - \max\{1, 0.8\}) = 0.35,$$
$$p_2 = \frac{1}{2}(\max\{0.7, 1\} - 0.7) + \frac{1}{2}(\max\{1, 0.8\} - 0.8) = 0.25,$$
$$p_3 = \frac{1}{2}(\max\{0.7, 1, 0.8\} - \max\{0.7, 1\}) + \frac{1}{2}(0.8 - 0) = 0.40.$$

It follows from (5) that $E[\xi] = 1 \times 0.35 + 3 \times 0.25 + 5 \times 0.40 = 3.10$.

3 Model formulation

A FLP with fuzzy costs and demands may be described as follows: assume that there are m clients having uncertain demand which is a fuzzy vector for some given commodity. The firm can open some new facilities in potential sites $i = 1, 2 \ldots, n$, the cost of each facility consists of fix opening and operating cost and variable operating cost, the later is a fuzzy variable. Each client can be supplied from an open facility where the commodity is made available. No more than 100% of a client's demand can be served, but the possibility exists that not all demand is served. The total supply from one facility to all clients can not exceed the capacity of the facility. The distribution pattern i.e., the quantities distributed form facilities to clients, is not fixed, it is adapted to the realization of fuzzy event with respected to the demand and variable operating cost. The problem of the firm is to choose the best locations for facilities to open to maximize profit or minimize costs.

In order to model the problem, we give the following notations:

$i = 1, 2, \ldots, n$: the index of facilities;
$j = 1, 2, \ldots, m$: the index of clients;
d_j: fuzzy demand of client j for a given commodity;
r_j: the unit price charged to clint j;
x_i: decision variable which is a binary variable equal to one if facility i is open and zero otherwise;
s_i: the capacity of facility i;
c_i: the fixed cost for opening and operating facility i;
v_i: the unit variable operating cost of facility i, which is a fuzzy variable;
y_{ij}: the quantity supplied to client j from facility i.
t_{ij} unit transportation cost from i to j.

All variable operating cost $v_i, i = 1, 2, \ldots, n$ and the demands d_j, $j = 1, 2, \ldots, m$ are fuzzy variables defined on a possibility space $(\Gamma, \mathcal{P}(\Gamma), \text{Pos})$, and for any $\gamma \in \Gamma$, $v_i(\gamma)$ and $d_j(\gamma)$ are the realizations of v_i and d_j, respectively, for each i and j.

Under the above assumptions and notations, taking the objective as maximization of expected profit, we can formulate a capacitated fuzzy FLP as follows:

$$\begin{aligned} \max \quad & -\sum_{i=1}^{n} c_i x_i + E\left[\max \sum_{i=1}^{n}\sum_{j=1}^{m} (r_j - v_i(\gamma) - t_{ij}) y_{ij}\right] \\ \text{subject to} \quad & x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n, \\ & \sum_{i=1}^{n} y_{ij} \leq d_j(\gamma), \quad j = 1, 2, \ldots, m, \\ & \sum_{j=1}^{m} y_{ij} \leq s_i x_i, \quad i = 1, 2, \ldots, n, \\ & y_{ij} \geq 0, \quad j = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m. \end{aligned} \tag{7}$$

In order to clarify the dynamic process of the problem more detailedly, we suppose that the decision variables of the model are divided into two categories. The location decision vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the first-stage decision which must be taken before the outcome of fuzzy event γ is revealed, here the outcome of fuzzy event refers to the realizations of fuzzy demands and fuzzy operating cost

$$\xi(\gamma) = (d_1(\gamma), \ldots, d_m(\gamma), v_1(\gamma), \ldots, v_n(\gamma)). \tag{8}$$

In the second stage, the demands of all clients are known. As a consequence, the second-stage decision variables y_{ij} for $i = i, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, which represent the distribution pattern, can be adjusted to the realization of the fuzzy event γ.

Following this scheme, we present a two-stage fuzzy FLP, in which there are two optimization problems to be

solved. By assuming $\boldsymbol{x}$ and γ to be fixed, the *second-stage problem* can be formulated as follows:

$$\begin{aligned} \max \sum_{i=1}^{n} & \sum_{j=1}^{m}(r_j - v_i(\gamma) - t_{ij})y_{ij} \\ \text{subject to} & \sum_{i=1}^{n} y_{ij} \le d_j(\gamma), \quad j = 1,2,\ldots,m, \\ & \sum_{j=1}^{m} y_{ij} \le s_i x_i, \quad i = 1,2,\ldots,n, \\ & y_{ij} \ge 0, \quad i = 1,2,\ldots,n, \quad j = 1,2,\ldots,m. \end{aligned} \tag{9}$$

Let $Q(\boldsymbol{x}, \xi(\gamma))$ be the optimal value of problem (9) at fixed $\boldsymbol{x}$ and $\xi(\gamma)$, it is usually called second-stage value function (abbreviated as SSVF) in two-stage fuzzy programming theory [22]. Furthermore, if we define expected second-stage value function (abbreviated as ESSVF)

$$\mathcal{Q}_E(\boldsymbol{x}) = E[Q(\boldsymbol{x}, \xi)], \tag{10}$$

where $E[\cdot]$ is the expected value operator with respect to fuzzy vector ξ, then it represents the expected total revenue obtained from the severed clients, given the first-stage decision $\boldsymbol{x}$.

Based on the above notations, the *first-stage* of the FLP can be formulated as follows:

$$\begin{aligned} \max \quad & -\sum_{i=1}^{n} c_i x_i + \mathcal{Q}_E(\boldsymbol{x}) \\ \text{subject to} \quad & x_i \in \{0,1\}, i = 1,2,\ldots,n. \end{aligned} \tag{11}$$

Combining the problems (9) and (11) yields the two-stage fuzzy FLP. It is equivalent to the problem (7).

Particularly, if fuzzy vector ξ in problem (9)–(11) is a discrete one that takes the following values

$$\begin{aligned} \widehat{\xi}^1 &= \left(\widehat{d}_1^1, \ldots, \widehat{d}_m^1, \widehat{v}_1^1, \ldots, \widehat{v}_n^1\right) && \text{with possibility } \mu_1 > 0, \\ \widehat{\xi}^2 &= \left(\widehat{d}_1^2, \ldots, \widehat{d}_m^2, \widehat{v}_1^2, \ldots, \widehat{v}_n^2\right) && \text{with possibility } \mu_2 > 0, \\ & \cdots && \cdots \\ \widehat{\xi}^N &= \left(\widehat{d}_1^N, \ldots, \widehat{d}_m^N, \widehat{v}_1^N, \ldots, \widehat{v}_n^N\right) && \text{with possibility } \mu_N > 0, \end{aligned}$$

and $\max_{k=1}^{N} \mu_k = 1$. Without any loss of generality, we assume that for any fixed $\boldsymbol{x}$ the SSVF $Q(\boldsymbol{x}, \xi(\gamma))$ satisfies the condition $Q(\boldsymbol{x}, \widehat{\xi}^1) \le Q(\boldsymbol{x}, \widehat{\xi}^2) \le \cdots \le Q(\boldsymbol{x}, \widehat{\xi}^N)$, then the value of the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$ at $\boldsymbol{x}$ is computed by the formula

$$\mathcal{Q}_E(\boldsymbol{x}) = \sum_{k=1}^{N} p_k Q(\boldsymbol{x}, \widehat{\xi}^k), \tag{12}$$

where

$$\begin{aligned} Q(\boldsymbol{x}, \widehat{\xi}^k) = \max & \sum_{i=1}^{n}\sum_{j=1}^{m}(r_j - \widehat{v}_i^k - t_{ij})y_{ij} \\ \text{subject to} & \sum_{i=1}^{n} y_{ij} \le \widehat{d}_j^k, \quad j = 1,2,\ldots,m, \\ & \sum_{j=1}^{m} y_{ij} \le s_i x_i, \quad i = 1,2,\ldots,n, \\ & y_{ij} \ge 0, \quad i = 1,2,\ldots,n, \quad j = 1,2,\ldots,m, \end{aligned} \tag{13}$$

and the corresponding weights p_i's are given by formula (6). The following example is provided to depict the process of solving the problem (9)–(11) in a simple discrete fuzzy vector scenario.

Example 3 Let $n = 2$, $m = 1$, $(r_1, r_2) = (5, 4)$, $(s_1, s_2) = (2, 2)$, $(t_1, t_2) = (3, 2)$, $(c_1, c_2) = (0.1, 0.3)$. If fuzzy operating cost v_1 takes the values 1 and 2 with possibilities 0.5 and 1, respectively, and v_2 takes the values 3 and 4 with possibilities 0.25 and 1, respectively, and fuzzy demand $d \equiv 3$, then problem (9)–(11) can be built as

$$\begin{aligned} &\max -0.1x_1 - 0.3x_2 + E[Q(\boldsymbol{x}, \xi)] \\ &\text{subject to } x_i \in \{0,1\}, i = 1,2. \end{aligned} \tag{14}$$

where

$$\begin{aligned} Q(\boldsymbol{x}, \xi(\gamma)) = \max \; & 2y_1 + 2y_2 - v_1(\gamma)y_1 - v_2(\gamma)y_2 \\ \text{subject to} \; & y_1 + y_2 \le 3, \\ & 0 \le y_1 \le 2x_1, \\ & 0 \le y_2 \le 2x_2. \end{aligned} \tag{15}$$

By the assumptions, we know fuzzy vector $\xi = (d, v_1, v_2)$ is discrete, and

$$\begin{aligned} \widehat{\xi}^1 &= (3,1,3) \quad \text{with possibility } \mu_1 = \min\{0.5, 0.25\} = 0.25, \\ \widehat{\xi}^2 &= (3,1,4) \quad \text{with possibility } \mu_2 = \min\{0.5, 1\} = 0.5, \\ \widehat{\xi}^3 &= (3,2,3) \quad \text{with possibility } \mu_3 = \min\{1, 0.25\} = 0.25, \\ \widehat{\xi}^4 &= (3,2,4) \quad \text{with possibility } \mu_4 = \min\{1, 1\} = 1. \end{aligned}$$

We note that the first-stage decision variable $\boldsymbol{x}$ takes four values in all: (0, 0), (1, 1), (1, 0) and (0,1). And, fuzzy vector ξ have four realizations (3, 1, 3), (3, 1, 4), (3, 2, 3) and (3, 2, 4) for each value of $\boldsymbol{x}$. The process of the solution of this problem can be divided roughly into three steps: Firstly, calculate the SSVF $Q(\boldsymbol{x}, \hat{\xi})$ by solving a second-stage programming (15) for each value of $\boldsymbol{x}$ and realization $\hat{\xi}$ of ξ. Secondly, compute the ESSVF $\mathcal{Q}_E(\boldsymbol{x}) = E[Q(\boldsymbol{x}, \xi)]$ for each $\boldsymbol{x}$ through formula (12). Thirdly, find the optimal solution by solving the first-stage programming (14). The detailed solution process is given below.

In the case of $\boldsymbol{x} = (0, 0)$, straightforwardly, we have the optimal second-stage solution $y_1^* = y_2^* = 0$, and the second-stage value $Q(0, 0, \widehat{\xi}^k) = 0$ for $k = 1, 2, 3, 4$. Hence, $E[Q(0, 0, \xi)] = 0$, and the objective value of the model at $\boldsymbol{x} = (0,0)$ is 0.

For the case that $\boldsymbol{x} = (1, 1)$, we can compute respectively that the second-stage value $Q(1, 1, \widehat{\xi}^k), k = 1, 2, 3, 4$ as follows. For $\widehat{\xi}^1 = (3, 1, 3)$ with possibility $\mu_1 = 0.25$, we have the second-stage programming is

$$\begin{aligned} Q(1,1,\widehat{\xi}^1) = \max \quad & y_1 - y_2 \\ \text{subject to:} \quad & y_1 + y_2 \le 3, \\ & 0 \le y_i \le 2, i = 1,2. \end{aligned}$$

Clearly, the optimal second-stage solution is $y_1^* = 2, y_2^* = 0$. Hence, $Q(1,1,\widehat{\xi}^1) = 2$ with possibility $\mu_1 = 0.25$. By the same reasoning, we can calculate

$$\begin{aligned} &Q(1,1,\widehat{\xi}^2) = 2 \quad \text{with possibility } \mu_2 = 0.5,\\ &Q(1,1,\widehat{\xi}^3) = 0 \quad \text{with possibility } \mu_2 = 0.25,\\ &Q(1,1,\widehat{\xi}^4) = 0 \quad \text{with possibility } \mu_2 = 1. \end{aligned}$$

That is

$$0 = Q(1,1,\widehat{\xi}^4) = Q(1,1,\widehat{\xi}^3) < Q(1,1,\widehat{\xi}^2) = Q(1,1,\widehat{\xi}^1) = 2.$$

Furthermore,

$$\begin{aligned} p_1 &= 0.5 \times (1-1) + 0.5 \times 0.25 = 0.125,\\ p_2 &= 0.5 \times (1-1) + 0.5 \times (0.5 - 0.25\}) = 0.125,\\ p_3 &= 0.5 \times (1-1) + 0.5 \times (0.5 - 0.5) = 0,\\ p_4 &= 0.5 \times 1 + 0.5 \times (\max\{1, 0.25, 0.5, 0.25\}\\ &\quad - \max\{0.25, 0.5, 0.25\}) = 0.75. \end{aligned}$$

As a consequence,

$$E[Q(1,1,\xi)] = 0 \times 0.75 + 0 \times 0 + 0.125 \times 2 + 0.125 \times 2 = 0.5,$$

and the objective value at $\boldsymbol{x} = (1,1)$ of the model is $-0.1 - 0.3 + 0.5 = 0.1$.

If $\boldsymbol{x} = (0, 1)$, we can calculate $Q(0,1,\widehat{\xi}^k) = 0$ for $k = 1, 2, 3, 4$. Thus, $E[Q(0, 1, \xi)] = 0$ and the objective value of the model at $\boldsymbol{x} = (0,1)$ is -0.3.

If $\boldsymbol{x} = (1, 0)$, we can calculate

$$\begin{aligned} &Q(1,0,\widehat{\xi}^1) = 2 \quad \text{with possibility} \mu_2 = 0.25,\\ &Q(1,0,\widehat{\xi}^2) = 2 \quad \text{with possibility} \mu_2 = 0.5,\\ &Q(1,0,\widehat{\xi}^3) = 0 \quad \text{with possibility} \mu_2 = 0.25,\\ &Q(1,0,\widehat{\xi}^4) = 0 \quad \text{with possibility} \mu_2 = 1, \end{aligned}$$

hence, we only need to compute

$$p_1 = 0.125, \quad p_2 = 0.125.$$

Furthermore,

$$E[Q(1,0,\xi)] = 0.125 \times 2 + 0.125 \times 2 = 0.5,$$

and the objective value of the model at $\boldsymbol{x} = (1,0)$ is 0.4.

Comparing the objective values for all the values of the first-stage decision $\boldsymbol{x}$, we obtain the optimal value of this model is 0.4 with the optimal solution $\boldsymbol{x}^* = (x_1^*, x_2^*) = (1, 0)$.

Observing the problem (9)–(11), we can see the fact that the complexity of the problem is directly related to the number of location decisions and the possible realizations of fuzzy variables. In other words, the problem size rapidly increases with the number of x_i and the realizations of ξ. In fact, as illustrated in Example 3, for each first-stage decision $\boldsymbol{x}$, there are different realizations $\xi(\gamma), \gamma \in \Gamma$, and for each pair $(\boldsymbol{x}, \xi(\gamma))$ we have to solve the second-stage programming (9), which is a linear programming. Therefore, it is very difficult to obtain the analytical expression of the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$. Furthermore, since the fuzzy operating cost v_i and fuzzy demand d_j involved in problem (9)–(11) are usually continuous which are defined through possibility distributions with infinite support, the model is inherently an infinite-dimensional optimization problem that can not be solved directly and exactly. As a consequence, algorithms designed to solve such a problem must rely on intelligent computing and some approximation scheme. Solution procedure will be discussed in the next section.

4 Solution procedure

In this section, using an scheme proposed by Liu [23], we shall approximate fuzzy variables with infinite supports by finitely supported ones, which enable us to solve the infinite-dimensional optimization model by solving a finite-dimensional problem. Furthermore, in order to accelerate the solution process, we employ an neural network (NN) based on the approximation scheme to simulate the ESSVF. Finally, to avoid getting stuck at a local optimal solution, we suggest a heuristic algorithm, which incorporates approximation approach, NN and PSO, to solve the problem (9)–(11).

4.1 Approximation approach

From the discussion of Sect. 3, we know that in order to solve the problem (9)–(11), it is required to evaluate the ESSVF

$$\mathcal{Q}_E(\boldsymbol{x}) = E[Q(\boldsymbol{x}, \xi)], \tag{16}$$

where ξ is the fuzzy vector described in (8).

For any given decision $\boldsymbol{x}$, we compute $\mathcal{Q}_E(\boldsymbol{x})$ by the following approach.

Assume that $\xi = (d_1, \ldots, d_m, v_1, \ldots, v_n)$ is a continuous fuzzy vector whose support is

$$\Xi = \prod_{j=1}^{m} [a_j^L, a_j^U] \times \prod_{i=1}^{n} [b_i^L, b_i^U] \tag{17}$$

where $[a_j^L, a_j^U]$ is the support of d_j and $[b_i^L, b_i^U]$ is that of v_i. Then we employ the method proposed in ([23]) to approximate the possibility distribution of ξ by a sequence of possibility distributions of discrete fuzzy vectors $\{\zeta_m\}$.

For each integer l, we define the discrete fuzzy vector $\zeta_l = (\zeta_{l,1}, \ldots, \zeta_{l,m}, \zeta_{l,m+1}, \ldots, \zeta_{l,m+n})$ as follows:

For each $j \in \{1, 2, \ldots, m\}$ and $i \in \{1, 2, \ldots, n\}$, define fuzzy variables $\zeta_{l,j} = g_{l,j}(d_j)$ and $\zeta_{l,m+i} = g_{l,m+i}(v_i)$, for $l = 1, 2, \ldots$, where $g_{l,j}(d_j)$'s and $g_{l,m+i}(v_i)$'s are given respectively as follows,

$$g_{l,j}(u_j) = \sup\left\{\frac{k}{l} \mid k \in Z,\ \text{s.t. } \frac{k}{l} < u_j\right\}, \quad u_j \in [a_j^L, a_j^U]$$

and

$$g_{l,m+i}(u_i) = \sup\left\{\frac{k}{l} \mid k \in Z,\ \text{s.t. } \frac{k}{l} < u_i\right\}, \quad u_i \in [b_i^L, b_i^U]$$

Z is the set of integers.

According to the above definitions, for each j, as d_j takes its values in the interval $[a_j^L, a_j^U]$, the fuzzy variables $\zeta_{l,j}$ takes values $\frac{k}{l}$ for $k = [l \cdot a_j^L], [l \cdot a_j^L] + 1, \ldots, [l \cdot a_j^U]$ with $[\cdot]$ the integer part of $\cdot$. Moreover, for each k, fuzzy variable $\zeta_{l,j}$ only takes the value $\frac{k}{l}$ as d_j takes its values in $[\frac{k}{l}, \frac{k+1}{l})$. Consequently, for each $\gamma \in \Gamma$, we have

$$\left|\zeta_{l,j}(\gamma) - d_j(\gamma)\right| < \frac{1}{l}, \quad j = 1, 2, \ldots, m.$$

By the same analysis, we can obtain

$$\left|\zeta_{l,m+i}(\gamma) - v_i(\gamma)\right| < \frac{1}{l}, \quad i = 1, 2, \ldots, n.$$

Note that ζ_l and ξ are $m+n$ary fuzzy vectors, and $\zeta_{l,j}^d, \zeta_{l,i}^v$ and d_j, v_i are their components, respectively, we have

$$\|\zeta_l(\gamma) - \xi(\gamma)\| = \sqrt{\sum_{j=1}^{m}(\zeta_{l,j}(\gamma) - d_j(\gamma))^2 + \sum_{i=1}^{n}(\zeta_{l,m+i}(\gamma) - v_i(\gamma))^2} \le \frac{\sqrt{m+n}}{l} \tag{18}$$

for all $\gamma \in \Gamma$. That implies the sequence $\{\zeta_l\}$ of fuzzy vectors converges to infinite-supported fuzzy vector ξ uniformly. The sequence $\{\zeta_l\}$ of finite-supported fuzzy vectors is referred to as *discretization* of the fuzzy vector ξ.

In the following, we provide an example to show the discretization process described above.

Example 4 Consider the triangular fuzzy variable $\xi = (2, 3, 4)$ in Example 1, define a sequence of discrete fuzzy variables $\zeta_l = g_l(\xi)$, where the function g_l is defined by

$$g_l(u) = \sup\left\{\frac{k}{l} | k \in Z, s.t. \frac{k}{l} < u\right\}, u \in [2, 4].$$

Determine the possibility distributions of the discrete fuzzy variables ζ_l for $l = 1, 2, \ldots$.

Let $l = 1$. Then ζ_1 takes only two values: 2 and 3. As ξ takes its values in [2, 3), ζ_1 takes the value 2 only, and ζ_1 takes the value 3 as ξ takes its values in [3,4). Hence,

$$\mu_1(2) = \text{Pos}\{\zeta_1 = 2\} = \text{Pos}\{2 \le \xi < 3\} = 1,$$

$$\mu_1(3) = \text{Pos}\{\zeta_1 = 3\} = \text{Pos}\{3 \le \xi < 4\} = 1.$$

That is, discrete fuzzy variable ζ_1 takes values 2 and 3 with possibility 1 each.

Let $l = 2$. Then ζ_2 takes four values, 2, $\frac{5}{2}$, 3 and $\frac{7}{2}$, as ξ takes it values in the intervals $[2, \frac{5}{2}), [\frac{5}{2}, 3), [3, \frac{7}{3})$ and $[\frac{7}{3}, 4)$, respectively. Therefore

$$\mu_2(2) = \text{Pos}\{\zeta_2 = 2\} = \text{Pos}\left\{2 \le \xi < \frac{5}{2}\right\} = \frac{1}{2},$$

$$\mu_2\left(\frac{5}{2}\right) = \text{Pos}\left\{\zeta_2 = \frac{5}{2}\right\} = \text{Pos}\left\{\frac{5}{2} \le \xi < 3\right\} = 1.$$

$$\mu_2(3) = \text{Pos}\{\zeta_2 = 3\} = \text{Pos}\left\{3 \le \xi < \frac{7}{2}\right\} = 1,$$

$$\mu_2\left(\frac{7}{2}\right) = \text{Pos}\left\{\zeta_2 = \frac{7}{2}\right\} = \text{Pos}\left\{\frac{7}{2} \le \xi < 4\right\} = \frac{1}{2}.$$

That is, ζ_2 takes values 2, $\frac{5}{2}$, 3 and $\frac{7}{2}$ with possibilities $\frac{1}{2}$, 1, 1 and $\frac{1}{2}$, respectively.

Generally, ζ_l takes $2l$ values $\frac{i}{l}, i = 2l, 2l+1, \ldots, 4l-1$. The possibility that ζ_l takes value $\frac{i}{l}$ is

$$\mu_l\left(\frac{i}{l}\right) = \text{Pos}\left\{\frac{i}{l} \le \xi < \frac{i+1}{l}\right\} = \begin{cases} \frac{i+1}{l} - 2, & \text{if } 2l \le i \le 3l-1 \\ 4 - \frac{i}{l}, & \text{if } 3l \le i \le 4l-1 \\ 0, & \text{otherwise.} \end{cases}$$

We now replace the possibility distribution of ξ by that of its discretization ζ_l, and approximate the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$ by $E[Q(\boldsymbol{x}, \zeta_l)]$ for each given decision $\boldsymbol{x}$. For each given integer l, the fuzzy vector ζ_l takes finite number of values, which are denoted by

$$\widehat{\zeta}_l^k = (\widehat{\zeta}_{l,1}^k, \ldots, \widehat{\zeta}_{l,m}^k, \widehat{\zeta}_{l,m+1}^k, \ldots, \widehat{\zeta}_{l,m+n}^k), \quad k = 1, 2, \ldots, K.$$

Then, we denote

$$v_k = \min\{\mu_{l,1}(\widehat{\zeta}_{l,1}^k), \ldots, \mu_{l,m}(\widehat{\zeta}_{l,m}^k), \mu_{l,m+1}(\widehat{\zeta}_{l,m+1}^k), \ldots, \mu_{l,m+n}(\widehat{\zeta}_{l,m+n}^k)\}$$

for $k = 1, 2, \ldots, K$, where $\mu_{l,i}$ are the possibility distribution of $\zeta_{l,i}$ for $i = 1, 2, \ldots, m, m+1, \ldots, m+n$, respectively. For each k, we solve the second-stage linear programming problem (9) through simplex algorithm [8, 10] and obtain the optimal value $Q(\boldsymbol{x}, \widehat{\zeta}_l^k)$, whose possibility is v_k. Rearrange the subscript k of v_k and $Q(\boldsymbol{x}, \widehat{\zeta}_l^k)$ such that

$$Q(\boldsymbol{x}, \widehat{\zeta}_l^1) \le Q(\boldsymbol{x}, \widehat{\zeta}_l^2) \le \cdots \le Q(\boldsymbol{x}, \widehat{\zeta}_l^K).$$

Calculate the wights $p_k, k = 1, 2, \ldots, K$ by formula (6). After that, the expected value $E[Q(\boldsymbol{x}, \zeta_l)]$ is computed by the formula

$$\mathcal{Q} = \sum_{k=1}^{K} p_k Q(\boldsymbol{x}, \widehat{\zeta}_l^k). \tag{19}$$

The following Theorem 1 will show that $E[Q(\boldsymbol{x}, \zeta_l)]$ converges to $\mathcal{Q}_E(\boldsymbol{x})$, as $l \to \infty$. As a consequence, the original ESSVF $\mathcal{Q}_E(\boldsymbol{x})$ can be approximated by $E[Q(\boldsymbol{x}, \zeta_l)]$ through the formula (19), provided l is sufficiently large. The process to compute the ESSVF is summarized as

Algorithm 1: Approximation approach

Step 1. Generate K points $\widehat{\zeta}_l^k = \left(\widehat{\zeta}_{l,1}^k, \ldots, \widehat{\zeta}_{l,m+n}^k\right)$ uniformly from the support Ξ of ξ for $k = 1, 2, \ldots, K$.

Step 2. Solve the second-stage linear programming (9) via simplex algorithm, and obtain the optimal value $Q(\boldsymbol{x}, \widehat{\zeta}_l^k)$ for $k = 1, 2, \ldots, K$.

Step 3. compute the possibility of $Q(\boldsymbol{x}, \widehat{\zeta}_l^k)$ by setting $v_k = \min\{\mu_{l,1}(\widehat{\zeta}_{l,1}^k), \ldots, \mu_{l,m+n}(\widehat{\zeta}_{l,m+n}^k)\}$ for $k = 1, 2, \ldots, K$.

Step 4. Rearrange the subscript k of v_k and $Q(\boldsymbol{x}, \widehat{\zeta}_l^k)$ such that $Q(\boldsymbol{x}, \widehat{\zeta}_l^1) \le Q(\boldsymbol{x}, \widehat{\zeta}_l^2) \le \cdots \le Q(\boldsymbol{x}, \widehat{\zeta}_l^K)$.

Step 5. Calculate the wights $p_k, k = 1, 2, \ldots, K$ by formula (6).

Step 6. Return the value of $\mathcal{Q}$ through the estimation formula (19).

The convergence of Algorithm 1 is ensured by the following theorem.

Theorem 1 *Consider the problem* (9)–(11). *Suppose the fuzzy cost and demand vector ξ is a continuous fuzzy vector with support* (17), *and ζ_l's are the discretization of the fuzzy vector ξ, then for any given feasible decision $\boldsymbol{x}$, we have*

$$\lim_{l \to \infty} E[Q(\boldsymbol{x}, \zeta_l)] = \mathcal{Q}_E(\boldsymbol{x}).$$

Proof Since the problem (9)–(11) is a two-stage fuzzy programming, and the second-stage programming (9) can be expressed in the form given below:

$$\begin{array}{ll} \max & V(\gamma)^T y \\ \textit{subject to} & Wy = H - T(\gamma)\boldsymbol{x} \\ & y \ge 0 \end{array}$$

where V is a matrix containing r_j, t_{ij} and fuzzy costs $v_i(\gamma)$ for $i = 1, 2, \ldots, n, T$ a matrix containing s_i and fuzzy demands $d_j(\gamma)$, H is the matrix of slack variables, and matrix W only consists of 0 and 1. Noting that matrix W is fixed, together with the fact that ξ is a continuous fuzzy vector with a compact support (17) and that ζ_l's are the discretization of ξ, by the properties of two-stage fuzzy programming with fixed recourse [22], we can obtain

$$\lim_{l \to \infty} E[Q(\boldsymbol{x}, \zeta_l)] = \mathcal{Q}_E(\boldsymbol{x}).$$

The proof is complete. □

4.2 Estimating the ESSVF by NN

So far, we have discussed the approximation approach for the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$. It is easy to see that the approach is a time-consuming process though it's convergence can be ensured by Theorem 1, since in each iteration of simulation, we have to solve the linear programming (9) many times in the second stage. To speed up the solution process, in this paper, we employ the fast BP algorithm [18] to train a feedforward NN to estimate $\mathcal{Q}_E(\boldsymbol{x})$. The training data set can be generated by the approximation approach of $\mathcal{Q}_E(\boldsymbol{x})$ discussed in Sect. 4.1 Therefore, much time can be saved during the solution process since it is not necessary to evaluate $\mathcal{Q}_E(\boldsymbol{x})$ by the approximation approach. We use the NN with input layer, one hidden layer and output layer connected in a feedforward way, in which there are n neurons in input layer representing the input value of decision variables, p neurons in hidden layer, and 1 neuron in output layer representing the value of the ESSVF. In addition, the number p in the hidden layer can be determined by pruning algorithm [6]. Let $\{(\boldsymbol{x}_k, y_k) \mid k = 1, 2, \ldots, N\}$ be a set of input-output data generated by fuzzy simulations. The training process is to find the best weight vector $\boldsymbol{w}$ that minimizes the following error function

$$Err(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{N} |F(\boldsymbol{x}_k, \boldsymbol{w}) - y_k|^2, \tag{20}$$

where $F(\boldsymbol{x}_k, \mathbf{w})$ is the output function of the NN. As for more details on NN and its applications, the reader can refer to [3, 7, 35, 36, 37].

4.3 A hybrid algorithm

Particle swarm optimization (PSO) developed by Kennedy and Eberhart [19] is an evolutionary computation technique which uses collaboration among a population of simple search agents (called particles) to find optima in function spaces. In PSO, a potential solution to a problem is represented as a particle i having current position x_{id} and the direction v_{id} in which the particle will travel. Each particle i maintains a record of the position of its previous best performance in a vector p_{id}. The variable g is the index of the particle with best performance so far in the population. An iteration comprises evaluation of each particle, then stochastic adjustment of v_{id} in the direction of particle i's best previous position p_{id} and the best previous position p_{gd} of any particle in the population. The direction vector v_{id} is updated first and then added to x_{id} according to the formulas given below [32]:

$$\begin{aligned} v_{id} = {} & \omega * v_{id} + c_1 * \text{rand}() * (p_{id} - x_{id}) \\ & + c_2 * \text{rand}() * (p_{gd} - x_{id}), \end{aligned} \tag{21}$$

$$x_{id} = x_{id} + v_{id} \tag{22}$$

where i is the ith particle, c_1 and c_2 are random numbers independently generated in the interval [0,4] which

represent the weighting of the stochastic acceleration terms that pull each particle towards p_{id} and p_{gd} positions, respectively, rand() is a uniform random number in the interval [0,1], and ω is the inertia weight whose value decreases linearly as the number of iterations of the algorithm increases. PSO has demonstrated great success in providing good solutions to many complex optimization problems and has received more and more attention during the past decade. The reader who are interested in detailed discussion about PSO and its applications may refer to [17, 19], [20, 32–34, 38).

In order to use a PSO algorithm to solve the location problem (9)–(11) more effectively, we define the following function:

$$\varphi(t_i) = \begin{cases} 1, & \text{if } t_i > 0.5 \\ 0, & \text{if } t_i \le 0.5 \end{cases} \tag{23}$$

for $i = 1, 2, \ldots,$ and denote $\varphi(\mathbf{t}) = (\varphi(t_1), \ldots, \varphi(t_n))$, where $\mathbf{t} = (t_1, \ldots, t_n) \in [0, 1]^n$. Hence, the problem (9)–(11) can be solved by solving the following modified model:

$$\begin{aligned} \max \quad & -\sum_{i=1}^{n} c_i \varphi(t_i) + \mathcal{Q}_E(\varphi(\mathbf{t})) \\ \text{subject to} \quad & t_i \in [0, 1], i = 1, 2, \ldots, n, \end{aligned} \tag{24}$$

where $\mathcal{Q}_E(\varphi(\mathbf{t})) = E[Q(\varphi(\mathbf{t}), \xi)]$ and

$$\begin{aligned} Q(\varphi(\mathbf{t}), \xi(\gamma)) = \max \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} (r_j - v_i(\gamma) - t_{ij}) y_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{n} y_{ij} \le d_j(\gamma), \quad j = 1, 2, \ldots, m, \\ & \sum_{j=1}^{m} y_{ij} \le s_i \varphi(t_i), \quad i = 1, 2, \ldots, n, \\ & y_{ij} \ge 0, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m. \end{aligned} \tag{25}$$

Clearly, the original problem (9)–(11) and modified problem (24, (25) correspond the same optimal value, and the optimal solution of problem (9)–(11) can be presented by that of problem (24, (25) using (23):

$$\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*) = (\varphi(t_1^*), \ldots, \varphi(t_n^*)). \tag{26}$$

In the following, we incorporate approximation approach, NN and PSO to produce a hybrid algorithm for solving the FPR problems. First, we generate a training data set for an ESSVF $\mathcal{Q}_E(\boldsymbol{x})$ by approximation approach. Then, using the generated input-output data, we train an NN by fast BP algorithm to estimate $\mathcal{Q}_E(\boldsymbol{x})$. We repeat this BP algorithm until the error for all vectors in the training set is reduced to an acceptable value or perform the specified number of epochs of training. After that, we use new data (which are not learned by the NN) to test the trained NN. If the test results are satisfactory, then we stop the training process; otherwise, we continue to train the NN. After the NN is well-trained, it is embedded into a PSO. During the solution process, the output values of the trained NN are used to represent the approximate values of $\mathcal{Q}_E(\boldsymbol{x})$. This process of the hybrid algorithm for solving the problem (9)–(11) is summarized as follows.

Algorithm 2 *A Hybrid Algorithm*

Step 1. Generate a set of input-output data for an ESSVF $\mathcal{Q}_E(\boldsymbol{x})$ through Algorithm 1;

Step 2 Train an NN to approximate $\mathcal{Q}_E(\boldsymbol{x})$ by the generated data;

Step 3. Initialize a population of particles (potential solutions) at random, set p_{id} equal to x_{id} for each particle i, and find p_{gd} for the population;

Step 4. Update all the particles by formulas (21) and (22);

Step 5. Calculate the objective values for all particles by the trained NN, and evaluate each particle according to the objective value;

Step 6. Update the p_{id} for each particle, and the p_{gd} for the population, respectively;

Step 7. Repeat Step 4 to Step 6 for a given number of generations;

Step 8. Return the particle p_{gd} as the optimal solution of modified problem (24, (25), and obtain the optimal solution of original problem (9)–(11) through formula (26).

5 A numerical example

Suppose a firm intends to open new facilities in six potential sites, whose fixed costs and variable operating costs are given in Table 1, and there are five clients whose demands are triangular fuzzy variables given below

$$d_1 = (12, 13, 14), d_2 = (10, 12, 14), d_3 = (14, 15, 16), d_4 = (12, 14, 16), d_5 = (9, 10, 12).$$

Example 5 Consider the following two-stage fuzzy facility location problem

Table 1 Capacities and fixed costs of six facilities

Facility site i	Capacity s_i	Fixed cost c_i	Variable cost v_i
1	10	3	(4, 5, 6)
2	14	1	(1, 2, 3)
3	8	2	(2, 3, 4)
4	9	1	(3, 4, 6)
5	12	2	(3, 4, 5)
6	10	3	(4, 5, 6)

$$\begin{array}{ll} \max & -3x_1 - x_2 - 2x_3 - x_4 - 2x_5 - 3x_6 + \mathcal{Q}_E(\boldsymbol{x}) \\ \textit{subject to} & x_1, x_2, \ldots, x_6 \in \{0, 1\}, \end{array} \tag{27}$$

where $\mathcal{Q}_E(\boldsymbol{x}) = E[Q(\boldsymbol{x}, \xi)]$ and

$$\begin{array}{ll} Q(\boldsymbol{x}, \xi(\gamma)) = \max & \sum_{i=1}^{6}\sum_{j=1}^{5}(r_j - v_i(\gamma) - t_{ij})y_{ij} \\ \textit{subject to} & \sum_{i=1}^{n} y_{ij} \le d_j(\gamma), j = 1, 2, \ldots, 5, \\ & \sum_{j=1}^{m} y_{ij} \le s_i x_i, i = 1, 2, \ldots, 6, \\ & y_{ij} \ge 0, i = 1, 2, \ldots, 6, j = 1, 2, \ldots, 5, \end{array} \tag{28}$$

and the values of $r_j - t_{ij}$, for $i = 1, 2, \ldots, 6, j = 1, 2, \ldots, 5$ are given by matrix

$$M = \left(r_j - t_{ij}\right)_{6\times 5} = \begin{pmatrix} 3 & 4 & 3 & 6 & 5 \\ 4 & 3 & 2 & 5 & 6 \\ 4 & 5 & 3 & 5 & 4 \\ 4 & 6 & 5 & 4 & 6 \\ 3 & 4 & 6 & 5 & 6 \\ 4 & 5 & 6 & 5 & 2 \end{pmatrix}.$$

To solve this problem, for any feasible solution $\boldsymbol{x}$, we generate 3,000 sample points via approximation approach to evaluate the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$. That is, we are first required to solve the second-stage programming (28) 3,000 times and obtain the SSVF $Q(\boldsymbol{x}, \xi(\gamma_i))$ for $i = 1, 2, \ldots, 3,000$. Then the value of $\mathcal{Q}_E(\boldsymbol{x})$ at $\boldsymbol{x}$ can be computed by formula (19).

Furthermore, we generate 2,000 input-output data by using the method mentioned above to train an NN to estimate the ESSVF $\mathcal{Q}_E(\boldsymbol{x})$. After the NN is well trained, it is embedded into a PSO algorithm to produce a hybrid algorithm (Algorithm 2) to search for the optimal solution. As have described in Sect. 4.3, for problem (27), (28), we replace all the $x_i \in \{0, 1\}$ by $\varphi(t_i) \in [0, 1]$, for $i = 1, 2, \ldots, 6$, where $\varphi(\cdot)$ is given by formula (23). Hence, $\mathbf{t} = (t_1, t_2, \ldots, t_6)$ is considered as the particle in the PSO, and the optimal solution of the problem is given by $\boldsymbol{x}^* = (\varphi(t_1^*), \varphi(t_2^*), \ldots, \varphi(t_6^*))$.

In this paper, parameters of PSO c_1 and c_2 are set equal to 2, the population size $P_{size} = 4$, the maximum generation $G_{max} = 60$, and using the method of (Shi and Eberhart, 1998a) that inertia weight decreases linearly from about 0.9 to 0.4 during a run, ω is given by

$$w = 0.5 * (G_{max} - G_n)/G_{max} + 0.4,$$

where G_n the number of current generation. A run of the hybrid algorithm (Algorithm 2) shows the optimal location decision is

$$\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_6^*) = (1, 1, 1, 1, 1, 0)$$

Table 2 Comparison solutions of Example 5.1

P_{size}	c_1	c_2	Gen	Optimal solution	Objectivevalue
4	2.0	2.0	5	(0, 1, 0, 1, 1, 0)	45.844616
4	2.0	2.0	10	(0, 1, 1, 1, 1, 1)	49.420930
4	2.0	2.0	16	(0, 1, 1, 1, 1, 0)	52.207808
4	2.0	2.0	20	(1, 1, 1, 1, 1, 0)	53.798394
4	2.0	2.0	40	(1, 1, 1, 1, 1, 0)	53.798394
4	2.0	2.0	60	(1, 1, 1, 1, 1, 0)	53.798394

whose objective value is 53.798394. In Table 2, we can compare the results at different generations of particles. It follows from Table 2 that the algorithm converges at the generation of 20. Furthermore, in order to test the effectiveness of the algorithm, we reset the $P_{size} = 100$ to achieve an ergodic search, and obtain the global optimal solution which is right $\boldsymbol{x}^* = (1, 1, 1, 1, 0)$ with objective value 53.798394. That is, the hybrid algorithm searches out the global optimal solution at the generation of 20 with population size 4, which implies the high effectiveness of the algorithm.

6 Concluding remarks

This paper puts forth a novel and realistic capacitated two-stage FLP with fuzzy costs and demands. Since the problem is a 0–1 integer two-stage fuzzy programming, and the possibility distributions of fuzzy costs and demands own infinite support, the proposed FLP can not be solved directly and exactly. Therefore, in order to tackle this complicated model, first, an approximation method of the location problem is employed, and the convergence of the method is proved. After that, a hybrid algorithm which integrates the approximation approach, neural network and particle swarm optimization is designed to solve the problem. Finally, a numerical example is provided to test the effectiveness of the hybrid algorithm. With the proposed model, the approximately optimal location can be determined when the location problems are with imprecise cost and demand parameters, or they are directly provided by the experts.

There is much room for further development based on this study, for instance, a more complex situation could be more interesting by considering a hybrid continuous-discrete location region, in that case, the solution should be totally different and much more difficult.

Acknowledgments The work was supported partially by "Ambient SoC Global COE Program of Waseda University" of Ministry of Education, Culture, Sports, Science and Technology, Japan, and by the Research Fellowships of the Japan Society for the Promotion of Science (JSPS) for Young Scientists.

References

1. Badri MA (1999) Combining the analytic hierarchy process and goal programming for global facility location–allocation problem. Int J Prod Econ 62(3):237–248
2. Bhattacharya U, Rao JR, Tiwari RN (1992) Fuzzy multi-criteria facility location problem. Fuzzy Sets Syst 51(3):277–287
3. Boehm O, Hardoon DR, Manevitz LM (2011) Classifying cognitive states of brain activity via one-class neural networks with feature selection by genetic algorithms. Int J Mach Learn Cybern 2(3):125–134
4. Bongartz I, Calamai PH, Conn AR (1994) A projection method for lp norm location–allocation problems. Math Program 66(1–3): 283–312
5. Carrizosa E, Conde E, Munoz-Marquez M, Puerto J (1995) The generalized Weber problem with expected distances. Rairo-Recherche Operationnelle Oper Res 29(1):35–57
6. Castellano G, Fanelli AM, Pelillo M (1997) An iterative pruning algorithm for feedforward neural networks. IEEE Trans Neural Netw 8:519–537
7. Chen C-J (2011) Structural vibration suppression by using neural classifier with genetic algorithm. Int J Mach Learn Cybern. doi: 10.1007/s13042-011-0053-9
8. Chvatal V (1983) Linear Programming. W. H. Freeman and Company, New York
9. Cooper L (1963) Location–allocation problems. Oper Res 11(3): 331–344
10. Dantzig GB (1963) Linear Programming and Extensions. Princeton University Press, Princeton, New Jersey
11. Darzentas J (1987) A discrete location model with fuzzy accessibility measures. Fuzzy Sets Syst 23(1):149–154
12. Dubois D, Prade H (1980) Fuzzy sets and systems: theory and applications. Academic Press, New York
13. Dubois D, Prade H (1988) Possibility theory. Plenum Press, New York
14. Ernst AT, Krishnamoorthy M (1999) Solution algorithms for the capacitated single allocation hub location problem. Ann Oper Res 86(1–4):141–159
15. Gong D, Gen M, Xu W, Yamazaki G (1995) Hybrid evolutionary method for obstacle location–allocation problem. Comput Ind Eng 29(1–4):525–530
16. Ishii H, Lee YL, Yeh KY (2007) Fuzzy facility location problem with preference of candidate sites. Fuzzy Sets Syst 158(17): 1922–1930
17. Jarboui B, Damak N, Siarry P, Rebai A (2008) A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl Math Comput 195(1):299–308
18. Karayiannis NB, Venetsanopoulos AN (1992) Fast learning algorithms for neural networks. IEEE Trans Circuits Syst II Analog Digital Signal Process 39:453–474
19. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks, vol IV, pp 1942–1948
20. Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann Publishers, San Francisco
21. Liu B, Liu YK (2002) Expected value of fuzzy variable and fuzzy expected value models. IEEE Trans Fuzzy Syst 10:445–450
22. Liu YK (2005) Fuzzy programming with recourse. Int J Uncertain Fuzziness Knowl Based Syst 13(4):381–413
23. Liu YK (2006) Convergent results about the use of fuzzy simulation in fuzzy optimization problems. IEEE Trans Fuzzy Syst 14(2):295–304
24. Logendran R, Terrell MP (1988) Uncapacitated plant location–allocation problems with price sensitive stochastic demands. Comput Oper Res 15(2):189–198
25. Laporte G, Louveaux FV, Hamme LV (1994) Exact solution to a location problem with stochastic demands. Transport Sci 28(2): 95–103
26. Louveaux FV, Peeters D (1992) A dual-based procedure for stochastic facility location. Oper Res 40(3):564–573
27. Love RF (1976) One-dimensional facility location–allocation using dynamic programming. Manag Sci 24(5):224–229
28. Lozano S, Guerrero F, Onieva L, Larraneta J (1998) Kohonen maps for solving a class of location–allocation problems. Eur J Oper Res 108(1):106–117
29. Megiddo N, Supowit KJ (1984) On the complexity of some common geometric location problems. SIAM J Comput 13(1):182–196
30. Murtagh BA, Niwattisyawong SR (1982) Efficient method for the muti-depot location em dash allocation problem. J Oper Res Soc 33(7):629–634
31. Nahmias S (1978) Fuzzy variable. Fuzzy Sets Syst 1(2):97–101
32. Shi Y, Eberhart RC (1998a) A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE international conference on evolutionary computation, pp 69–73
33. Shi Y, Eberhart RC (1998b) Parameter selection in particle swarm optimization. In: Proceedings of 7th annual conference on evolutionary programming, pp 591–600
34. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177(3):1930–1947
35. Tong DL, Mintram R (2010) Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. Int J Mach Learn Cybern 1(1–4):75–87
36. Wang S, Liu Y, Dai X (2007) On the continuity and absolute continuity of credibility functions. J Uncertain Syst 1(3):185–200
37. Wang X-Z, Li C-G (2008) A definition of partial derivative of random functions and its application to RBFNN sensitivity analysis. Neurocomputing 71(7–9):1515–1526
38. Wang X-Z, He Y-L, Dong L-C, Zhao H-Y (2011) Particle swarm optimization for determining fuzzy measures from data. Inform Sci 181(19):4230–4252
39. Wen M, Iwamura K (2008) Fuzzy facility location–allocation problem under the Hurwicz criterion. Eur J Oper Res 184(2): 627–635
40. Zadeh LA (1965) Fuzzy sets. Inform Contr 8(3):338–353
41. Zadeh LA (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst 1(1):3–28
42. Zhou J, Liu B (2007) Modeling capacitated location–allocation problem with fuzzy demands. Comput Ind Eng 53(3):454–468