Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Sciences 158 (2004) 3–13

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Tabu search for fuzzy optimization and applications

Chunguang Li *, Xiaofeng Liao, Juebang Yu

Institute of Electronic Systems, College of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, PR China

Received 10 June 2003; accepted 23 July 2003

Abstract

In this paper, we present a tabu search technique to approximately solve fuzzy optimization problems. We demonstrate the performance of the proposed method by applying it to an elementary fuzzy optimization problem. Other applications of the method to fuzzy linear programming, fuzzy regression and the training of fuzzy neural networks are also presented.

Keywords: Tabu search; Fuzzy optimization; Fuzzy linear programming; Fuzzy linear regression; Fuzzy neural network

1. Introduction

Fuzzy optimization problems play a very important role in many fuzzy systems. In the last past years, several kinds of fuzzy optimization problems have appeared in literatures [2–5,7,12,13] and obviously with them different approaches of resolution have been proposed too. Among the many methods presented in literatures, we mention the fuzzy genetic algorithm (FGA), proposed by Buckley and Hayashi [5], which can produce good approximate

* Corresponding author. Tel.: +86-28-83202508.
E-mail address: cgli@uestc.edu.cn (C. Li).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2003.07.015

solution in solving fuzzy optimization problems. But it should be pointed out that the FGA is too computational expensive.

We introduce here a tabu search (TS) method, a stochastic global optimization method originally developed by Glover [8,9] for very large combinatorial optimization tasks and extend to continuous-valued functions in [6], for the fuzzy optimization problems. For notational convenience, we denote it as FTS in the rest of this paper.

Our work has been inspired by the fact that TS is very general and conceptually much simpler than simulated annealing (SA) or genetic algorithm (GA). Besides, it has been shown that TS is superior to SA and GA both in the time required to obtain a solution and in the solution quality in solving many optimization problems [6,8–11].

The rest of this paper is organized as follows. In next section, we briefly describe the basic TS technique. And in Section 3, we will present our FTS algorithm. The FTS will be applied in Section 4 to generate good approximate solutions to an elementary fuzzy optimization problem. Simulation results show that it outperforms the FGA in solving this problem. More substantial applications are given in Section 5, where we discuss FTS solutions to fuzzy linear programming (FLP), fuzzy linear regression (FLR) and the training of fuzzy neural networks (FNNs). Finally conclusions are summarized in Section 6.

Let us now introduce some basic notations to be used in this paper. We place a bar over a symbol if it represent a fuzzy set. So, $\overline{A}, \overline{B}, \overline{C}, \ldots, \overline{X}, \overline{Y}, \overline{Z}$ all represent fuzzy sets. All our fuzzy sets will be fuzzy subsets of the real numbers. If $\overline{A}$ is a fuzzy set, then $\overline{A}(x)$ denotes its membership function evaluated at x. A triangular fuzzy number $\overline{N}$ is defined by three numbers $a < b < c$ where (1) $\overline{N}(x) = 0$ for $x \leqslant a$ and $x \geqslant c$, and $\overline{N}(b) = 1$; and (2) the graph of $\overline{N}(x) = y$ is a straight line segment from $(a, 0)$ to $(b, 1)$ on $[a, b]$ and a straight line segment from $(b, 1)$ to $(c, 0)$ on $[b, c]$. We write $\overline{N} = (a, b, c)$ for triangular fuzzy numbers. We use the standard arithmetic of fuzzy sets based on the extension principle.

2. General description of tabu search

The TS is a metaheuristic global optimization method originally developed by Glover [8,9] for large combinatorial optimization tasks, and extended to continuous-valued function in [6]. It is different from the well-known hill-climbing local search techniques in the sense that it does not become trapped in local solutions, i.e. the TS allows moves out of current solution that makes the objective function worse in hope that it will achieve a better solution. The TS technique has been shown in literatures [1,6,8–11] that it is superior to SA and

GA both in time required to obtain a solution and in the solution quality in solving some combinatorial optimization problems.

The TS requires the following basic elements to be defined [1]:

- *Configuration* is a solution or an assignment of values to variables.
- A *move* is a transition ($s' \to s''$) from one trial solution (s') to another (s'').
- *Set of candidate moves* (*neighborhood*, or *trial solutions*) is the set of all possible moves out of a current configuration.
- *Tabu restrictions:* there are certain conditions imposed on moves which make some of them forbidden, these forbidden moves are known as *tabu*. And are maintained in short-term memory on a list, called *tabu list*. After a specified duration (*tabu list size*), they are removed from the list and are free to be visited again.
- *Aspiration criteria:* these are rules that override tabu restrictions, i.e. if a certain move is forbidden by tabu restriction, then the aspiration criterion, when satisfied, can make this move allowable.

Given the above basic elements, the TS schemes can be described as follows. Start with a certain configuration, evaluate the objective function for that configuration, then follow a certain set of candidate moves. If the best of the moves is not tabu or if the best is tabu, but satisfies the aspiration criterion, then pick that move and consider it to be the new current configuration; otherwise, pick the best move that is not tabu and consider it to be the new current configuration. Repeat the procedure for a certain number of iterations. On termination, the best solution obtained so far is the solution obtained by the TS approach. A recent trend in the TS field is the use of dynamic tabu lists [10,11], which in their most common form use variable list sizes, as a way of managing short-term memory. In this paper, we use a dynamic tabu list size strategy during the course of the search.

3. Tabu search for fuzzy optimization

A general function F with fuzzy input and output can be expressed as

$$\overline{Y} = F(\overline{X}) \tag{1}$$

where $\overline{X}$ is the input and it is a fuzzy subset of some interval $[0, M]$, $M > 0$. $\overline{Y}$ is the output from F given $\overline{X}$. In this section we assume that there is only one independent variable and we will generalize it to more independent variables case in Section 5. Also, we can work with any interval for $\overline{X}$ and we specified the interval $[0, M]$ only for notational convenience. As described below, the solving of fuzzy optimization is not a trivial work. The fuzzy optimization problems are much more complex than their crisp counterpart. Moreover, the

optimal solutions of the fuzzy optimization problem cannot be obtained by differentiating the objective function with respect to the fuzzy variables. We provide a TS method for solving the fuzzy optimization in this paper.

We wish to find $\overline{X}$ in $[0, M]$ to "maximize" $\overline{Y}$. However, we cannot maximize $\overline{Y}$ since it is a fuzzy set. We use the centroid of $\overline{Y}$ to measure the largeness of $\overline{Y}$, which is the same as that in [5]. Now we wish to find $\overline{X}$ in $[0, M]$ to maximize the centroid θ

$$\theta = \text{centroid}(\overline{Y}) \tag{2}$$

θ will be the objective function in our FTS. Next, we do the same discretization work for the fuzzy sets $\overline{X}$ as in [5]. Let N be a positive integer and choose

$$z_i = i \cdot M/N, \quad 0 \leqslant i \leqslant N \tag{3}$$

Thus we discretize $[0, M]$ into $N + 1$ points z_i, with $z_0 = 0$ and $z_N = M$. We will now use a vector $\overline{X}$, comprised by the $N + 1$ points, to represent fuzzy variable as shown below:

$$\overline{X} = (x_0, x_1, \ldots, x_N) \tag{4}$$

where

$$x_i = \overline{X}(z_i), \quad 1 \leqslant i \leqslant N \tag{5}$$

We input the discrete version of $\overline{X}$ to F and obtain θ of $F(x_0, x_1, \ldots, x_N)$. So, F maps $[0, 1]^{N+1}$ into the real numbers. Now we want to find a vector $\overline{X}$ in $[0, 1]^{N+1}$ to maximize θ. FTS will be designed to find this vector in $[0, 1]^{N+1}$ in this paper.

There are some important things require to be described before we state our FTS.

- Let $\overline{X}_c, \overline{X}_t, \overline{X}_b$ denote the *current*, *trial*, and *best solution* and θ_c, θ_t, θ_b denote the corresponding *current*, *trial* and *best objective value*, respectively.
- Given a current solution $\overline{X}_c$, one can *generate a trial solution* using several strategies. We use here the following neighborhood structure as described in [6]. The solution space $S = [0, 1]^{N+1}$ is partitioned into disjunct cells by division of the coordinate interval along the $x_0, x_1, \ldots, x_N$ axes into $p_0, p_1, \ldots, p_N$ parts. The problem-specific empirical partition parameter $P = (p_0, p_1, \ldots, p_N)$ determines a unique partition of S into cells. Thus specifies the "address" of each cell. At each iteration step, we draw n_s sample point from a uniform distribution over n_c chosen cells. These points are the neighbors of the current solution $\overline{X}_c$ (trial solutions). The number of the trial solutions is $n_c n_s$. Thus, the "address" of a cell can be expressed as the following array A.

n_0	n_1	n_2	…	n_N

where $1 \leqslant n_i \leqslant p_i$, $1 \leqslant i \leqslant N$ denotes the n_ith interval of the ith axis. We use the following strategy to choose cell at each iteration: given $\overline{X}_c$, which belongs to cell A_c, and a probability threshold P, for $i = 0, 1, 2, \ldots, N$, draw a random number $r \sim u(0, 1)$, if $r < P$, then $A_t(i) = A_c(i)$; otherwise, draw randomly an integer l from the following set $\{l : l = 1, 2, \ldots, p_i, l \neq A_c(i)\}$, and let $A_t(i) = l$, where A_c and A_t are the address of the current and trial solutions.

- The *probability threshold* (P) controls the shake-up that is performed on a certain solution to choose a neighbor. The higher the value of P, the less shake-up is allowed and consequently the closer the neighbor to the current solution and vice versa.
- The cell, from which a move is picked at a certain iteration is *tabu* and is maintained on tabu list. After a specified duration (*tabu list size*, or *tabu period*), it is removed from the list and is free to be visited again. We use dynamic tabu list size strategy in this paper.
- The *dynamic tabu list strategy* presented in literature [10] is adopted in this paper. There are three list sizes: small (S), medium (M) and large (L). They are set equal to N, $1.5N$ and $2N$, respectively. The value of tabu list size is initially set to S and then systematically changed every N iterations following the sequence {S, M, S, L, M, L}. The sequence is repeated as many times as necessary until the end of the search.
- The *number of trial solutions* (NTS $= n_c n_s$) controls the number of trial moves to be generated from a current one to decide on the next move. Clearly, the larger the value of NTS, the better, simply because one has more choices to select from. However, this is done at the expense of more computational effort. Therefore, if one were to decide a priori on a certain amount of computational effort to be made by the algorithm, then larger values of NTS at each iteration means less number of iterations are made and vice versa.
- Let the *aspiration condition* be $\theta(F(\overline{X}_t)) > \theta_b$, i.e. if a move to $\overline{X}_t$ is forbidden by tabu restriction, then the aspiration condition, $\theta(F(\overline{X}_t)) > \theta_b$, when satisfied, can make this move allowable.

Now, we are ready to state our FTS

(1) Initialization: Let $\overline{X}_c$ be an arbitrary initial solution generate randomly from $[0, 1]^{N+1}$ and θ_c be the corresponding objective function value computed using Eq. (1) and (2), let $\overline{X}_b = \overline{X}_c$, $\theta_b = \theta_c$. Select values for the following parameters: P (probability threshold), NTS $= n_c n_s$ (number of trial solutions), N_{max} (the maximal number of iterations allowed without improvement), NNI (number of non-improved iterations) $= 0$. Go to step 2.

(2) Using $\overline{X}_c$ to generate NTS trial solutions $\overline{X}_t^1, \overline{X}_t^2, \ldots, \overline{X}_t^{NTS}$, and evaluate their corresponding objective function value $\theta_t^1, \theta_t^2, \ldots, \theta_t^{NTS}$, go to step 3.

(3) Ordering $\theta_t^1, \theta_t^2, \ldots, \theta_t^{NTS}$ in an descending order, and denote them by $\theta_t^{[1]}, \theta_t^{[2]}, \ldots, \theta_t^{[NTS]}$. If $\theta_t^{[1]}$ is not tabu, or if it is tabu but $\theta_t^{[1]} > \theta_b$, then let $\overline{X}_c = \overline{X}_t^{[1]}$, $\theta_c = \theta_t^{[1]}$ and go to step 4; otherwise let $\overline{X}_c = \overline{X}_t^{[L]}$, $\theta_c = \theta_t^{[L]}$, where $\theta_t^{[L]}$ is the maximum objective function of $\overline{X}_t^{[2]}, \ldots, \overline{X}_t^{[NTS]}$ that is not tabu and go to step 4. If all $\overline{X}_t^{[2]}, \ldots, \overline{X}_t^{[NTS]}$ are tabu, let NNI = NNI + 1 and go to step 2.

(4) Let A_c keep tabu status for a number of iterations that corresponding to the current size of tabu list. If $\theta_c > \theta_b$, let $\overline{X}_b = \overline{X}_c$, $\theta_b = \theta_c$ and NNI = 0; if $\theta_c \leqslant \theta_b$, let NNI = NNI + 1, if NNI $< N_{max}$, go to step 2, otherwise stop.

The above algorithm is designed for solving fuzzy optimization task when the independent variable $\overline{X}$ is a general fuzzy subset of $[0, M]$. When the independent variable $\overline{X}$ is a special type of fuzzy set (triangular, trapezoidal, Gaussian, ... fuzzy numbers), TS can also be applied to produce (approximate) solution. Assume $\overline{X}$ be a triangular fuzzy number, or $\overline{X} = (a, b, c)$, $0 \leqslant a < b < c \leqslant M$. We can also partition the solution space $[0, M]^3$ into cells. And do the same search as described above except that in each iterations we should sort the three numbers of the new produced trial solution in an ascending order to make it be a fuzzy triangular number.

In next section, we will use our algorithm to calculate an approximate solution to an elementary fuzzy optimization problem.

4. An example

We will demonstrate the performance of the proposed method by applying it to an elementary fuzzy optimization example in this section. The corresponding crisp optimization problem of it is simple, but the fuzzy version is very complex.

We wish to solve the fuzzy optimization problem [5]:

$$\max \overline{Y} = \overline{X}(1 - \overline{X}) \tag{6}$$

for $\overline{X}$ a fuzzy subset of $[0, 1]$. We choose the centroid θ of $\overline{Y}$ as a measure of the largeness of $\overline{Y}$. So we wish to find $\overline{X}$, a fuzzy subset of $[0, 1]$ to maximize θ.

Now let us assume that $\overline{X}$ is a discrete fuzzy subset of $[0, 1]$ as required in our FTS. Let $\overline{X}(z_i) = x_i$, for $z_i = i/N$, $i = 0, 1, \ldots, N$. We do not know what are the values of x_i to maximize θ. From the functional dependence of θ on the x_i as shown below, we know that we cannot differentiate θ with respect to the x_i to obtain an optimal solution [5].

Let $\overline{Y}(y_i) = e_i$, $0 \leqslant i \leqslant M$, then $\theta = \sum_{i=0}^{M} y_i e_i / \sum_{i=0}^{M} e_i$. For simplicity let $N = 10$ and let us compute e_4 corresponding to $y_4 = 0.04$. Due to fuzzy multiplication, e_4 is not a simple function of the x_i. We may obtain $y_4 = 0.04$ three ways: $z_1(1 - z_6)$, $z_2(1 - z_8)$, $z_4(1 - z_9)$ from $\overline{X}$ and $1 - \overline{X}$, respectively. Fuzzy

multiplication then implies the $e_4 = \max\{\min\{x_1, x_6\}, \min\{x_2, x_8\}, \min\{x_4, x_9\}\}$. Obviously, we cannot differentiate θ with respect to the x_i to obtain an optimal solution [5]. After computing every y_i, we know that $M = 42$ when $N = 10$.

Now, let us apply our FTS algorithm to the problem. The parameters for the algorithm are

(1) The discretization of $[0, 1]$ is $z_0 = 0$, $z_1 = 0.1, \ldots, z_{10} = 1.0$ ($N = 10$).
(2) The partition of the solution space is $P = (2, 2, 2, 2, 2, 2, 2, 2, 2, 2)$. That is, each axes are parted into two parts.
(3) The number of trial solutions is $(n_c, n_s) = (1, 10)$.
(4) The probability threshold is $P = 0.85$.
(5) The tabu list size $S = 10$, $M = 15$, $L = 20$.
(6) The maximum number of iterations allowed without improvement is $N_{\max} = 100$.

For comparison, we also simulate the FGA described in [5]. The FGA will be stopped when 100 generations have been done without improvement. In 10 simulations, the best solutions obtained by FGA and FTS are shown in Table 1. The corresponding objective function values are respectively 0.4536 and 0.4914, which are shown in Table 2. Table 2 also presents the averaged objective function value obtained by FGA and FTS and the averaged time required ratio in 10 simulations, respectively. From Tables 1 and 2, we know

Table 1
The best solutions $\overline{X}$ obtained by FGA and FTS

Z_i	$\overline{X}(z_i)$ obtained by FGA ($q = 0.003$)	$\overline{X}(z_i)$ obtained by FTS
0.0	0.9039	0.9989
0.1	0.0657	0.0023
0.2	0.0008	0.0007
0.3	0.0013	0.0101
0.4	0.0091	0.0005
0.5	0.0904	0.0011
0.6	0.0107	0.0009
0.7	0.0016	0.0003
0.8	0.0002	0.0042
0.9	0.0310	0.0010
1.0	0.9007	0.9895

Table 2
Comparisons between FGA and FTS in 10 simulations

	Best objective function value	Averaged objective function value	Average time required ratio
FGA	0.4536	0.4018	$\frac{7.9}{1}$
FTS	0.4914	0.4697	

that the FTS is superior to the FGA both in the time required to obtain a solution and in the solution quality obtained in solving this fuzzy optimization problem.

In fact, if we let $\overline{X} = (1,0,0,0,0,0,0,0,0,0,1)$, then $\overline{Y}$ is symmetric with respect to 0.5, we can get $\theta = 0.5$. It is the maximum value of θ. The above solution implies that the FTS can produce good approximate answers to the fuzzy optimization problem.

5. Applications

In this section, we briefly discuss three applications of FTS. They are three well-known fuzzy optimization problems.

5.1. Fuzzy linear programming

FLP has been widely studied in literatures [3–5,7], now we propose a FTS solution to the FLP problem.

Consider the FLP problem [5]:

$$\begin{aligned} \max \quad & \overline{Q} = \overline{C}_1\overline{X}_1 + \overline{C}_2 X_2 \\ \text{subject to} \quad & \overline{A}_{11}\overline{X}_1 + \overline{A}_{12}\overline{X}_2 \leqslant \overline{B}_1, \\ & \overline{A}_{21}\overline{X}_1 + \overline{A}_{22}\overline{X}_2 \leqslant \overline{B}_2, \\ & \overline{X}_1, \overline{X}_2 \geqslant 0 \end{aligned} \tag{7}$$

where the $\overline{C}_i$, $\overline{A}_{ij}$ and $\overline{B}_j$ are all triangular fuzzy numbers. Let

$$\overline{\Omega}_i = \overline{B}_i - (\overline{A}_{i1}\overline{X}_1 + \overline{A}_{i2}\overline{X}_2), \quad i = 1,2 \tag{8}$$

and then we solve

$$\max(\overline{Q} + \psi_1\overline{\Omega}_1 + \psi_2\overline{\Omega}_2) \tag{9}$$

for $\overline{X}_1, \overline{X}_2 \geqslant 0$. In (9), the $\psi_i > 0$ are penalty coefficients.

Next, we use the inequalities constraint $\overline{\Omega}_i \geqslant 0$ to estimate constants $M_i > 0$ so that $\overline{X}_i$ should be a fuzzy subset of $[0, M_i]$, $i = 1,2$. What this means is that if $\overline{X}_1$ ($\overline{X}_2$) is not a fuzzy subset of $[0, M_1]$ ($[0, M_2]$) then surely $\overline{\Omega}_1 \geqslant 0$ or $\overline{\Omega}_2 \geqslant 0$ is not satisfied [5].

Let $\theta = m(\overline{Q} + \psi_1\overline{\Omega}_1 + \psi_2\overline{\Omega}_2)$ be our measure of how big this fuzzy set is. We wish to find fuzzy sets $\overline{X}_i$ in $[0, M_i]$, $i = 1,2$, to maximize θ. We discretize the intervals $[0, M_i]$ to get

$$\overline{X}_1 = (x_{10}, x_{11}, \ldots, x_{1N_1}) \tag{10}$$

$$\overline{X}_2 = (x_{20}, x_{21}, \ldots, x_{2N_2}) \tag{11}$$

To employ the FTS algorithm, we concatenate $\overline{X}_1$ and $\overline{X}_2$ to obtain

$$\overline{S} = (x_{10}, x_{11}, \ldots, x_{1N_1}, x_{20}, x_{21}, \ldots, x_{2N_2}) \tag{12}$$

Treating $\overline{S}$ as an independent variable, we can employ the FTS algorithm to this FLP problem.

5.2. Fuzzy linear regression

Regression is concerned with discovering functional relationships between variables when we have some data on these variables. Fuzzy regression is then concerned with finding functional relationships between fuzzy variables.

We consider a system identification problem where we know some inputs $\overline{X}_i$ and outputs $\overline{Y}_i$, $1 \leqslant i \leqslant n$, from a function F but we do not know the exact structure of F. We have

$$\overline{Y}_i = F(\overline{X}_i), \quad 1 \leqslant i \leqslant n \tag{13}$$

We will try a linear relationship $\overline{Y} = \overline{A}\overline{X} + \overline{B}$ for unknown fuzzy sets $\overline{A}$ and $\overline{B}$. We wish to 'fit' the linear equation to the data. This is what is called FLR, which has been widely studied in literatures. In FLR, researchers usually assume that all the fuzzy sets are of the same type (triangular, trapezoidal, ... fuzzy numbers) and produce an analytical solution for $\overline{A}$ and $\overline{B}$ to minimize some error function. In this paper, we also want to minimize some error function but we allow $\overline{A}$ and $\overline{B}$ to be arbitrary fuzzy sets in certain intervals.

Let d be some distance measure used between fuzzy sets. That is d will measure how close $\overline{W}_i = \overline{A}\overline{X}_i + \overline{B}$ is to $\overline{Y}_i$. Then we can choose

$$E = \sum_{i=1}^{n} d(\overline{Y}_i, \overline{W}_i) \tag{14}$$

to be minimized. The FTS was constructed for a maximization problem. So, we make the change to maximize θ, where

$$\theta = M - E \tag{15}$$

for some sufficiently large $M > 0$ [5].

Initial analysis leads us to believe that $\overline{A}$ will be a non-negative fuzzy set in $[0, M_1]$, $M_1 > 0$ and $\overline{B}$ will be a fuzzy set in $[-M_2, M_2]$, $M_2 > 0$ [5]. Discretize the intervals $[0, M_1]$ and $[-M_2, M_2]$, and concatenate the discrete versions of them into one array and partition the axes into cells, we can apply the FTS algorithm to the FLR task.

5.3. Training fuzzy neural networks

FNNs have been investigated by many researchers. For a survey, please refer to Buckley and Hayashi's paper [14].

We consider a FNN with fuzzy set input signals and fuzzy weights as shown in Fig. 1.

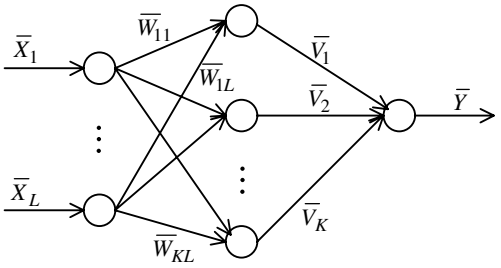

Fig. 1. A fuzzy neural network.

In this FNN, the inputs $\overline{X}_i$, the weights $\overline{W}_{ij}$, $\overline{V}_i$, and the output $\overline{Y}$ are all fuzzy variables. The input to the ith hidden neuron is

$$\overline{I}_i = \overline{W}_{i1}\overline{X}_1 + \cdots + \overline{W}_{iL}\overline{X}_L, \quad 1 \leqslant i \leqslant K \tag{16}$$

The output from the ith hidden neuron is

$$\overline{S}_i = f(\overline{I}_i), \quad 1 \leqslant i \leqslant K \tag{17}$$

for sigmoidal f, where the extension principle is used to obtain $\overline{S}_i$. It follows that the input to the output neuron is

$$\overline{I}_0 = \overline{V}_1\overline{S}_1 + \cdots + \overline{V}_K\overline{S}_K \tag{18}$$

and the output will be

$$\overline{Y} = f(\overline{I}_0) \tag{19}$$

where the extension principle is also used to obtain $\overline{Y}$.

Let the training set be $(\overline{X}_p, \overline{D}_p)$, $\overline{X}_p = (\overline{X}_{p1}, \ldots, \overline{X}_{pL})$ for inputs and $\overline{D}_p$ desired output, $1 \leqslant p \leqslant P$. Given $\overline{X}_p$, let the actual output be $\overline{Y}_p$. Thus the error measure can be defined as

$$\overline{E} = \sum_{p=1}^{P} (\overline{D}_p - \overline{Y}_p)^2 \tag{20}$$

which is to be "minimized". This a fuzzy optimization problem with $K(L+1)$ fuzzy variables $\overline{W}_{ij}$, $\overline{V}_i$, $i = 1, \ldots, K$, $j = 1, \ldots, L$.

Discretize the intervals of each fuzzy variables, and concatenate the discrete versions of them into one array and partition the axes into cells, we can apply the FTS algorithm to the training of FNN.

6. Conclusions and remarks

In this paper, we present a TS technique to approximately solve fuzzy optimization problems. We demonstrate the performance of the proposed method

by applying it to a fuzzy optimization problem. Computer simulation results show that the FTS is superior to the FGA both in the time required to obtain a solution and in the solution quality in solving the problems. Other applications of the method to FLP, fuzzy regression and the training of FNN are also presented. Thus this paper provides a novel and efficient method for the solving of fuzzy optimization problems.

The method presented in this paper can be further developed by introducing the concepts of the parallelized TS, long-term memory, diversification of search et al., to reduce the computing time. More efficient neighborhood structure may also be developed.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 60271019 and the Youth Science and Technology Foundation of UESTC under Grant YF020207.

References

[1] K.S. Al-sultan, A tabu search approach to the clustering problem, Pattern Recognit. 28 (1995) 1443–1451.
[2] J.J. Buchley, T. Feuring, Linear and non-linear fuzzy regression: evolutionary algorithm solutions, Fuzzy Sets Syst. 112 (2000) 381–394.
[3] J.J. Buckley, T. Feuring, Y. Hayashi, Neural net solutions to fuzzy linear programming, Fuzzy Sets Syst. 106 (1999) 99–111.
[4] J.J. Buckley, H. Hayashi, Applications of fuzzy chaos to fuzzy simulation, Fuzzy Sets Syst. 99 (1998) 151–157.
[5] J.J. Buckley, Y. Hayashi, Fuzzy genetic algorithm and applications, Fuzzy Sets Syst. 61 (1994) 129–1436.
[6] D. Cvijovic, J. Klinowski, Taboo search: an approach to the multiple minima problem, Science 267 (1995) 664–666.
[7] M. Delgado, J.L. Verdegay, M.A. Vila, A general model for fuzzy linear programming, Fuzzy Sets Syst. 29 (1989) 21–29.
[8] F. Glover, Tabu search, part I, ORSA J. Comp. 1 (1989) 190–206.
[9] F. Glover, Tabu search, part II, ORSA J. Comp. 2 (1990) 4–32.
[10] M. Laguna, F. Glover, Band width packing: a tabu search approach, Mgmt. Sci. 39 (1993) 492–500.
[11] C.Y. Lee, H.G. Kang, Cell planning with capacity expansion in mobile communications: A tabu search approach, IEEE Trans. Veh. Tech. 49 (2000) 1678–1691.
[12] E. Lee, R.J. Li, Fuzzy multiple objective programming and compromise programming with Pareto optimum, Fuzzy Sets Syst. 53 (1993) 275–288.
[13] H.R. Maleki, M. Tata, M. Mashinchi, Linear programming with fuzzy variables, Fuzzy Sets Syst. 109 (2000) 21–33.
[14] J.J. Buckley, Y. Hayashi, Fuzzy neural network: a survey, Fuzzy Sets Syst. 66 (1994) 1–13.