

## Simultaneous Perturbation Stochastic Approximation Algorithm Combined with Neural Network and Fuzzy Simulation\*

NING Yufu (宁玉富)<sup>1,2</sup>, TANG Wansheng (唐万生)<sup>1</sup>, GUO Changyou (郭长友)<sup>2</sup>

(1. Institute of Systems Engineering, Tianjin University, Tianjin 300072, China;

2. Department of Computer Science, Dezhou University, Dezhou 253023, China)

**Abstract:** In order to solve three kinds of fuzzy programming models, i.e. fuzzy expected value model, fuzzy chance-constrained programming model, and fuzzy dependent-chance programming model, a simultaneous perturbation stochastic approximation algorithm is proposed by integrating neural network with fuzzy simulation. At first, fuzzy simulation is used to generate a set of input-output data. Then a neural network is trained according to the set. Finally, the trained neural network is embedded in simultaneous perturbation stochastic approximation algorithm. Simultaneous perturbation stochastic approximation algorithm is used to search the optimal solution. Two numerical examples are presented to illustrate the effectiveness of the proposed algorithm.

**Keywords:** fuzzy variable; fuzzy programming; fuzzy simulation; neural network; approximation theory; perturbation techniques; computer simulation; simultaneous perturbation stochastic approximation algorithm

Many optimization problems contain fuzzy information. Possibility theory<sup>[1]</sup> has been well developed and applied to this kind of optimization problems<sup>[2–5]</sup>. Fuzzy programming is an important tool to handle the optimization problems, which usually includes three types of models: fuzzy expected value model<sup>[6]</sup>, fuzzy chance-constrained programming model<sup>[7,8]</sup>, and fuzzy dependent-chance programming model<sup>[9]</sup>. Fuzzy programming models can be converted into crisp equivalents in some special cases, and then solved by the classical algorithms for crisp programming models. However, it is difficult for complex fuzzy programming models. To solve fuzzy programming models, hybrid intelligent algorithms (HIAs) were designed<sup>[6–9]</sup>. The HIAs first generate a set of training input-output data for the functions with fuzzy variables by fuzzy simulation, then train a neural network (NN) to approximate the functions according to the generated training data, and finally embed the trained NN in genetic algorithm (GA). GA is used to search the optimal solution in the entire space. The HIAs are feasible in

solving fuzzy programming models. However, GA is time-consuming in finding the global optimum, especially in solving large-dimensional optimization problems. Therefore, it is necessary to seek out other optimization methods to solve fuzzy programming models.

Usually, one would always need the global optimal solution. However, in practice this solution is not often available and one must be satisfied with obtaining a local optimal solution. In a lot of practical optimization problems, a local optimal solution is a fully acceptable solution for the resources available (human time, money, material, computer time, etc.) to be spent on the optimization.

In fuzzy environments, it is difficult to directly obtain the value of objective function, especially the gradient of the function with respect to the decision vector. Stochastic approximation (SA) algorithm uses the (possibly noisy) measurements of objective function to approximate the gradient, which was first applied to finding extrema of functions<sup>[10,11]</sup>. However, the SA

Accepted date: 2007-11-05.

\*Supported by National Natural Science Foundation of China (No.70471049) and China Postdoctoral Science Foundation (No. 20060400704).

NING Yufu, born in 1967, male, Dr, associate Prof.

Correspondence to NING Yufu, E-mail: ning@tju.edu.cn.

algorithm has not been widely applied due to the fact that  $2p$  function measurements are needed in each iteration ( $p$  represents the number of parameters being optimized). Apparently, the time of run would increase sharply with the rise of  $p$ . To overcome the disadvantage, the simultaneous perturbation stochastic approximation (SPSA) algorithm was designed<sup>[12]</sup> based on a highly efficient and easily implemented simultaneous perturbation approximation to the gradient: this gradient approximation used only two objective function measurements in each iteration independent of  $p$ . In essence, the SPSA algorithm is a "hill-climbing" method, so it is a local search algorithm. The SPSA algorithm has been validated and developed<sup>[13,14]</sup>. More specifically, it is concluded<sup>[13]</sup> that SPSA is preferable to use by comparing the algorithm with other SA algorithms such as the standard finite-difference SA and the random-directions SA. The theoretical and numerical global convergence property of SPSA was examined, and two theorems on the global convergence of SPSA were established<sup>[14]</sup>. The numerical studies show obviously better performance of SPSA as a global optimizer than the popular simulated annealing algorithm and GA, see Ref. [15].

Inspired by the above-mentioned algorithms, this paper proposes an efficient optimization algorithm to solve fuzzy programming models.

## 1 Preliminaries

### 1.1 Fuzzy variables

A fuzzy variable is defined as a function from a possibility space  $(\Theta, P(\Theta), Pos)$  to the set of real numbers, where  $\Theta$  is a universe,  $P(\Theta)$  is the power set of  $\Theta$ , and  $Pos$  is a possibility measure defined on  $P(\Theta)$ . The necessity measure of  $A$  is defined by

$$Nec\{A\} = 1 - Pos\{A^c\} \tag{1}$$

**Definition 1** (Ref. [6]) Let  $(\Theta, P(\Theta), Pos)$  be a possibility space, and  $A$  a set in  $P(\Theta)$ . Then the credibility measure of  $A$  is defined by

$$Cr\{A\} = \frac{1}{2}(Pos\{A\} + Nec\{A\}) \tag{2}$$

**Definition 2** (Ref. [6]) Let  $\xi$  be a fuzzy variable on the possibility space  $(\Theta, P(\Theta), Pos)$ . Then the expected value  $E[\xi]$  is defined by

$$E[\xi] = \int_0^{+\infty} Cr\{\xi \geq r\} dr - \int_{-\infty}^0 Cr\{\xi \leq r\} dr \tag{3}$$

provided that at least one of the two integrals is finite.

**Definition 3** (Ref. [2]) Let  $\xi$  be a fuzzy variable on the possibility space  $(\Theta, P(\Theta), Pos)$ , and  $\alpha \in (0, 1]$ , then

$$\xi_{sup}(\alpha) = \sup\{r \mid Cr\{\xi \geq r\} \geq \alpha\} \tag{4}$$

is called the  $\alpha$ -optimistic value of  $\xi$ .

### 1.2 Fuzzy programming models

In fuzzy environments, there are usually three kinds of fuzzy programming models. In order to obtain the decision with the maximum expected return, the following single-object fuzzy expected value model is provided<sup>[6]</sup>:

$$\begin{aligned} & \max E[f(\mathbf{x}, \boldsymbol{\xi})] \\ & \text{s.t. } E[g_j(\mathbf{x}, \boldsymbol{\xi})] \leq 0, \quad j=1, \dots, p \end{aligned} \tag{5}$$

where  $\mathbf{x}$  is a decision vector;  $\boldsymbol{\xi}$  is a fuzzy vector;  $f(\mathbf{x}, \boldsymbol{\xi})$  is the objective function;  $g_j(\mathbf{x}, \boldsymbol{\xi})$  are the constraint functions for  $j=1, \dots, p$ .

When we want to maximize the optimistic return, we have the following single-objective fuzzy chance-constrained programming model<sup>[2]</sup>:

$$\begin{aligned} & \max \bar{f} \\ & \text{s.t. } \\ & \quad Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \leq \bar{f}\} \geq \beta \\ & \quad Cr\{g_j(\mathbf{x}, \boldsymbol{\xi}) \leq 0, j=1, \dots, p\} \geq \alpha \end{aligned} \tag{6}$$

where  $\alpha$  and  $\beta$  are predetermined confidence levels with  $0 < \alpha \leq 1$  and  $0 < \beta \leq 1$ .

A fuzzy dependent-chance programming theory is provided based on selecting the decision with the maximum credibility to meet the event. A typical formulation is given as follows<sup>[9]</sup>:

$$\begin{aligned} & \max Cr\{h_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0, k=1, \dots, q\} \\ & \text{s.t. } g_j(\mathbf{x}, \boldsymbol{\xi}) \leq 0, j=1, \dots, p \end{aligned} \tag{7}$$

where  $h_k(\mathbf{x}, \boldsymbol{\xi}) \leq 0, k=1, \dots, q$  are fuzzy events, and  $g_j(\mathbf{x}, \boldsymbol{\xi}) \leq 0, j=1, \dots, p$  are uncertain environments.

### 1.3 Fuzzy simulations

In solving the models mentioned above, the key is to calculate the values of the functions  $E[f(\mathbf{x}, \boldsymbol{\xi})]$ ,  $\max\{\bar{f} \mid Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \beta\}$  and  $Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \leq 0\}$  for every fixed  $\mathbf{x}$ . However, it is almost impossible to find

an analytical method to obtain these values in most cases. Therefore, it is necessary to employ fuzzy simulations to estimate these values.

The fuzzy simulation method for estimating  $E[f(\mathbf{x}, \boldsymbol{\xi})]$  for every fixed  $\mathbf{x}$  is described as follows<sup>[6]</sup>.

Step 1 Set  $e = 0$ .

Step 2 Determine the number of samples, written as  $M$ .

Step 3 Generate uniformly  $\theta_k$  from  $\Theta$  such that  $Pos\{\theta_k\} \geq \varepsilon$  for  $k=1, \dots, M$ , where  $\varepsilon$  is a sufficiently small positive number.

Step 4 Set  $v_k = Pos\{\theta_k\}$ .

Step 5 Set  $a = f(\mathbf{x}, \boldsymbol{\xi}(\theta_1)) \wedge \dots \wedge f(\mathbf{x}, \boldsymbol{\xi}(\theta_M))$ ,  $b = f(\mathbf{x}, \boldsymbol{\xi}(\theta_1)) \vee \dots \vee f(\mathbf{x}, \boldsymbol{\xi}(\theta_M))$ .

Step 6 Generate uniformly  $r$  from  $[a, b]$ .

Step 7 If  $r \geq 0$ , then  $e \leftarrow e + Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \geq r\}$  where

$$Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \geq r\} = \frac{1}{2} \left( \max_{1 \leq k \leq M} \{v_k \mid f(\mathbf{x}, \boldsymbol{\xi}(\theta_k)) \geq r\} + \min_{1 \leq k \leq M} \{1 - v_k \mid f(\mathbf{x}, \boldsymbol{\xi}(\theta_k)) < r\} \right) \quad (8)$$

Step 8 If  $r < 0$ , then  $e \leftarrow e - Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \leq r\}$  where

$$Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \leq r\} = \frac{1}{2} \left( \max_{1 \leq k \leq M} \{v_k \mid f(\mathbf{x}, \boldsymbol{\xi}(\theta_k)) \leq r\} + \min_{1 \leq k \leq M} \{1 - v_k \mid f(\mathbf{x}, \boldsymbol{\xi}(\theta_k)) > r\} \right) \quad (9)$$

Step 9 Repeat from Step 6 to Step 8 for  $M$  times.

Step 10 Compute

$$E[f(\mathbf{x}, \boldsymbol{\xi})] = a \vee 0 + b \wedge 0 + e \cdot (b - a) / M$$

Step 11 Return  $E[f(\mathbf{x}, \boldsymbol{\xi})]$ .

For more details about fuzzy simulations, see Refs. [2—9]. Three convergence theorems were proved<sup>[16]</sup> about the use of fuzzy simulation in computing the credibility of a fuzzy event, the optimistic value of a return function, and the expected value of a fuzzy variable.

## 2 SPSA algorithm combined with NN and fuzzy simulation

In this section, an SPSA algorithm combined with NN and fuzzy simulation is proposed to solve fuzzy

programming models. The procedure for maximizing the objective function is described as follows.

Step 1 Generation of training data by fuzzy simulation

Generate training input-output data for the uncertain functions with fuzzy variables like

$$U_1: \mathbf{x} \rightarrow E[f(\mathbf{x}, \boldsymbol{\xi})]$$

$$U_2: \mathbf{x} \rightarrow \max\{\bar{f} \mid Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \beta\}$$

$$U_3: \mathbf{x} \rightarrow Cr\{f(\mathbf{x}, \boldsymbol{\xi}) \leq 0\}$$

by fuzzy simulation, where  $\mathbf{x}$  is the decision vector with  $n$  decision variables,  $\boldsymbol{\xi}$  is the fuzzy vector, and  $\beta$  is the predetermined confidence level.

Step 2 Training of NN

Train an NN to approximate the uncertain functions by the generated training data.

Step 3 Parameters selection and initiation

Choose nonnegative parameters  $a, c, A, \alpha$  and  $\gamma$ , which will be used in the following steps. Then set  $k = 1$ , and choose initial evaluation  $\hat{\mathbf{x}}_1$  of  $\mathbf{x}$  such that  $\hat{\mathbf{x}}_1$  satisfies the constraints. In this step, the trained NN may be used to check the feasibility of  $\hat{\mathbf{x}}_1$ .

Step 4 Generation of simultaneous perturbation vector

Generate an  $n$ -dimensional random perturbation vector  $\mathbf{A}_k$  by Monte Carlo simulation, and the  $i$ th component of the  $\mathbf{A}_k$  vector is written as  $A_{ki}$  ( $i = 1, \dots, n$ ),

$$A_{ki} = \begin{cases} 1, & \text{with probability } \frac{1}{2} \\ -1, & \text{with probability } \frac{1}{2} \end{cases}$$

Step 5 Evaluations of objective values

Compute  $c_k = c / k^\gamma$ . If  $\hat{\mathbf{x}}_k + c_k \mathbf{A}_k$  and  $\hat{\mathbf{x}}_k - c_k \mathbf{A}_k$  satisfy the constraints, then their objective values may be obtained by the trained NN, written as  $\hat{y}(\hat{\mathbf{x}}_k + c_k \mathbf{A}_k)$  and  $\hat{y}(\hat{\mathbf{x}}_k - c_k \mathbf{A}_k)$ , respectively. Otherwise, return to Step 4.

Step 6 Gradient approximation

Calculate the following gradient approximation:

$$\hat{\mathbf{g}}_k(\hat{\mathbf{x}}_k) = \frac{\hat{y}(\hat{\mathbf{x}}_k - c_k \mathbf{A}_k) - \hat{y}(\hat{\mathbf{x}}_k + c_k \mathbf{A}_k)}{2c_k} \begin{bmatrix} A_{k1}^{-1} \\ A_{k2}^{-1} \\ \vdots \\ A_{kn}^{-1} \end{bmatrix}$$

Step 7 Update of evaluation of  $\mathbf{x}$

Compute  $a_k = a / (A + k)^\alpha$ , then set

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k - a_k \hat{\mathbf{g}}_k(\hat{\mathbf{x}}_k)$$

Step 8 Check of constraints

If  $\hat{\mathbf{x}}_{k+1}$  satisfies the constraints, go to the next step. Otherwise, return to Step 4. In this step, the trained NN may be used to check the feasibility of  $\hat{\mathbf{x}}_{k+1}$ .

Step 9 Iteration or termination

If the stopping criterion is met, go to the next step. Otherwise, set  $k = k + 1$ , and return to Step 4. Usually the stopping criterion is that there is little change in a few successive iterations or the maximum allowable number of iterations has been reached.

Step 10 Display of result

Display  $\hat{\mathbf{x}}_{k+1}$  as the optimal solution, and compute its objective value by the trained NN.

The flow diagram of the proposed algorithm is shown in Fig.1.

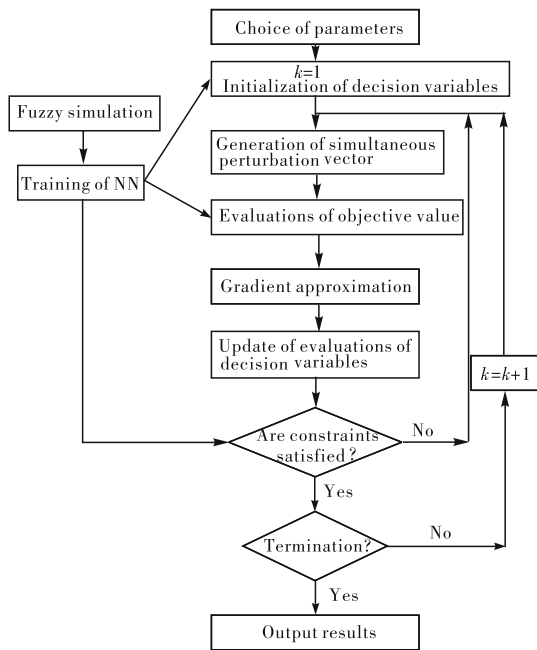


Fig. 1 Flow diagram of the proposed algorithm

**Remark 1** The purpose of training NN in the proposed algorithm is to improve the efficiency of the algorithm. During the iterations of SPSA, the values of objective functions and constraint functions are used. These values are obtained by NN rather than by fuzzy simulations. The method can greatly reduce the time spent on fuzzy simulations in the whole algorithm. For example, there is an optimization problem with an objective function and a constraint function with fuzzy variables. The stopping criterion of the algorithm is that

1 000 iterations have been reached. If an NN is trained with 2 000 training data, fuzzy simulation is called for 4 000 times. If an NN is not trained, the values of the objective function and the constraint function are obtained by calling fuzzy simulation for four times at each iteration. When the iteration is terminated, fuzzy simulation is called for 4 000 times. Apparently, the times of fuzzy simulation are equal in the two situations. However, the times for checking constraints are usually more than two at each iteration, and the stopping criterion of the algorithm is usually that more than 1 000 iterations have been reached. Therefore, if an NN is not embedded in the algorithm, the time spent on fuzzy simulation is much more.

**Remark 2** The choice of the gain sequences ( $a_k$  and  $c_k$ ) is crucial to the performance of the proposed algorithm. Practically effective (and theoretically valid) values for  $\alpha$  and  $\gamma$  are 0.602 and 0.101, respectively. The values of  $a$  and  $A$  can be chosen together to ensure effective practical performance of the algorithm. A guideline is to choose  $A$  such that it is much less than the maximum number of allowable iterations and choose  $a$  such that  $a/(A+1)^\alpha$  times the magnitude of elements in  $\hat{\mathbf{g}}_1(\hat{\mathbf{x}}_1)$  is approximately equal to the smallest of the desired change magnitudes among the elements of  $\mathbf{x}$  in the early iterations. With  $\alpha$  and  $\gamma$  as specified above, one typically finds that in a high-noise setting it is necessary to pick a smaller  $a$  and larger  $c$  than in a low-noise setting (see Ref. [15]).

### 3 Numerical examples

In this section, we employ the proposed algorithm to solve two fuzzy programming models.

**Example 1** Let us consider the following fuzzy expected value model<sup>[6]</sup>:

$$\min E[\sin(x_1 + \xi_1) + 2\sin(x_2 + \xi_2) + 3\sin(x_3 + \xi_3)]$$

s.t.

$$E[\sqrt{|\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3|}] \leq 5$$

$$2x_1 + 3x_2 + 4x_3 \leq 12$$

$$x_i > 0, \quad i = 1, 2, 3$$

(10)

where

$\xi_1$  is a triangular fuzzy variable (1,2,3);

$\xi_2$  is a fuzzy variable with membership function  $\mu_{\xi_2}(r) = \exp(-(r-4)^2)$ ;

$\xi_3$  is a trapezoidal fuzzy variable (2,3,4,8).

For convenience, we write

$$\mathbf{x} = (x_1, x_2, x_3)$$

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$$

$$f(\mathbf{x}, \boldsymbol{\xi}) = \sin(x_1 + \xi_1) + 2\sin(x_2 + \xi_2) + 3\sin(x_3 + \xi_3)$$

$$g(\mathbf{x}, \boldsymbol{\xi}) = \sqrt{|\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3|}$$

In order to solve the model, we first produce training input-output data for the uncertain functions

$$U_1 : \mathbf{x} \rightarrow E[\sin(x_1 + \xi_1) + 2\sin(x_2 + \xi_2) + 3\sin(x_3 + \xi_3)]$$

$$U_2 : \mathbf{x} \rightarrow E[\sqrt{|\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3|}]$$

by fuzzy simulation. Then we train an NN (3 input neurons, 5 hidden neurons, 2 output neurons) to approximate these uncertain functions. Then the trained NN is embedded into SPSA. The initial values of decision variables are set as follows:  $x_1 = 1.0$ ,  $x_2 = 1.0$ ,  $x_3 = 1.0$ . The parameters in SPSA are determined as follows:  $a = 0.16$ ,  $c = 0.20$ ,  $A = 100$ ,  $\alpha = 0.602$ ,  $\gamma = 0.101$ . After a run of the proposed algorithm, the optimal solution is

$$x_1 = 2.4851, x_2 = 0.5332, x_3 = 0.5388$$

with the objective value  $-3.3821$  (10 000 samples in fuzzy simulation, 2 000 training data in NN, and 1 000 iterations in SPSA).

The variation of objective value by different numbers of iterations in solving Eq. (10) is shown in Fig. 2, where  $N$  represents the number of iterations.

The variation of decision variables after different numbers of iterations is shown in Fig.3—Fig.5.

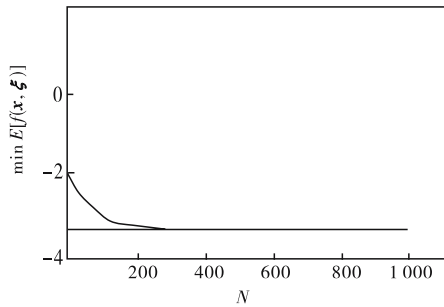


Fig. 2 Variation of objective value by different numbers of iterations in Example 1

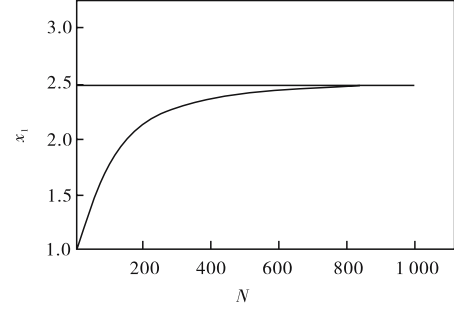


Fig. 3 Variation of  $x_1$  in Example 1

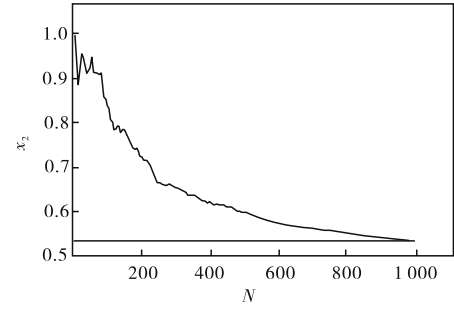


Fig. 4 Variation of  $x_2$  in Example 1

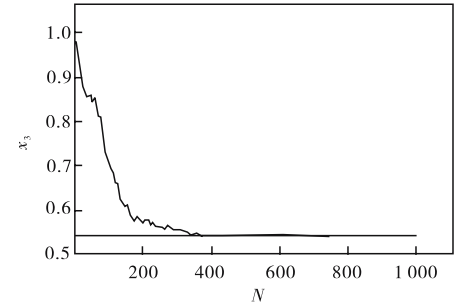


Fig. 5 Variation of  $x_3$  in Example 1

**Example 2** Let us consider the following fuzzy chance-constrained programming model<sup>[2]</sup>:

$$\begin{aligned} & \max \bar{f} \\ & \text{s.t.} \\ & Cr\{\sqrt{x_1 + \xi_1} + \sqrt{x_2 + \xi_2} + \sqrt{x_3 + \xi_3} \geq \bar{f}\} \geq 0.90 \\ & Cr\{\sqrt{(x_1 + \xi_1)^2 + (x_2 + \xi_2)^2 + (x_3 + \xi_3)^2} \leq 6\} \geq 0.80 \\ & x_i \geq 0, \quad i = 1, 2, 3 \end{aligned} \tag{11}$$

where  $\xi_1$ ,  $\xi_2$  and  $\xi_3$  are the triangular fuzzy variables (0, 1, 2), (1, 2, 3) and (2, 3, 4), respectively.

For convenience, we write

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, x_3) \\ \boldsymbol{\xi} &= (\xi_1, \xi_2, \xi_3) \\ f(\mathbf{x}, \boldsymbol{\xi}) &= \sqrt{x_1 + \xi_1} + \sqrt{x_2 + \xi_2} + \sqrt{x_3 + \xi_3} \\ g(\mathbf{x}, \boldsymbol{\xi}) &= \sqrt{(x_1 + \xi_1)^2 + (x_2 + \xi_2)^2 + (x_3 + \xi_3)^2} - 6 \end{aligned}$$

In order to solve the model, we first produce training input-output data for the uncertain functions,

$$\begin{aligned} U_1: \mathbf{x} \rightarrow \max \left\{ \bar{f} | Cr \left\{ \sqrt{x_1 + \xi_1} + \sqrt{x_2 + \xi_2} + \sqrt{x_3 + \xi_3} \geq \bar{f} \right\} \geq 0.90 \right\} \\ U_2: \mathbf{x} \rightarrow Cr \left\{ \sqrt{(x_1 + \xi_1)^2 + (x_2 + \xi_2)^2 + (x_3 + \xi_3)^2} \leq 6 \right\} \end{aligned}$$

by fuzzy simulation. Then we train an NN (3 input neurons, 6 hidden neurons, 2 output neurons) to approximate these uncertain functions. Then the trained NN is embedded into SPSA. We set the initial values of decision variables as follows:  $x_1 = 0.2, x_2 = 0.2, x_3 = 0.2$ . The parameters in SPSA are determined as follows:  $a = 0.16, c = 0.20, A = 100, \alpha = 0.602, \gamma = 0.101$ . After a run of the proposed algorithm, the optimal solution is

$$x_1 = 1.0422, x_2 = 0.7178, x_3 = 0.5063$$

with the objective value 4.2474 (10 000 samples in fuzzy simulation, 3 000 training data in NN, and 300 iterations in SPSA).

The variation of objective value by different numbers of iterations in solving Eq. (11) is shown in Fig.6.

The variation of decision variables after different numbers of iterations is shown in Fig.7—Fig.9.

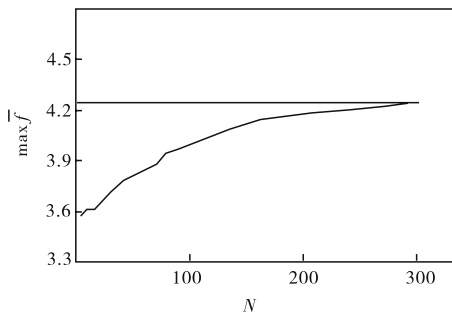


Fig. 6 Variation of objective value by different numbers of iterations in Example 2

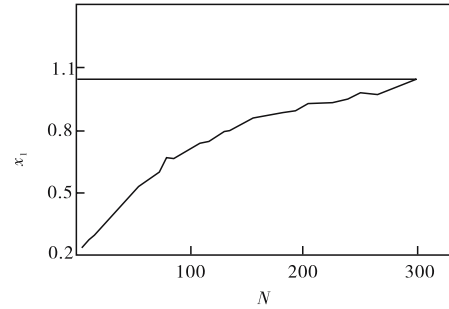


Fig. 7 Variation of  $x_1$  in Example 2

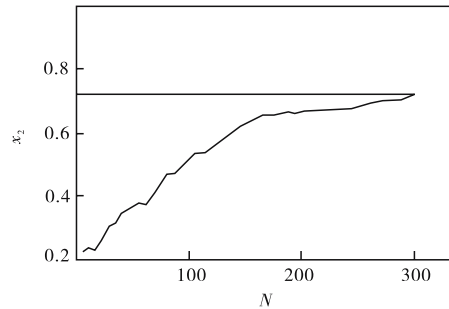


Fig. 8 Variation of  $x_2$  in Example 2

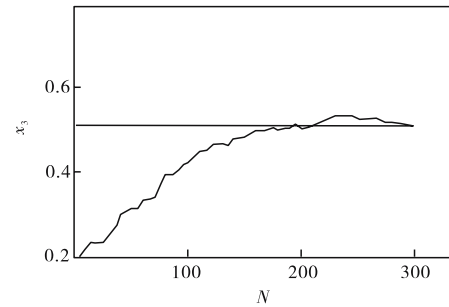


Fig. 9 Variation of  $x_3$  in Example 2

## 4 Conclusions

In this paper, a new algorithm is proposed to solve fuzzy programming models. The results of numerical examples show that the algorithm has good property of convergence.

## References

- [1] Zadeh L A. Fuzzy sets as a basis for a theory of possibility [J]. *Fuzzy Sets and Systems*, 1978, 8(1): 3-28.
- [2] Liu B. *Theory and Practice of Uncertain Programming* [M]. Physica-Verlag, Heidelberg, 2002.
- [3] Liu B. Toward fuzzy optimization without mathematical ambiguity [J]. *Fuzzy Optimization and Decision Making*, 2002, 1(1): 43-63.
- [4] Liu B. *Uncertainty Theory: An Introduction to Its Axiomatic Foundations* [M]. Springer-Verlag, Berlin,

- 2004.
- [ 5 ] Liu B. A survey of credibility theory [J]. *Fuzzy Optimization and Decision Making*, 2006, 5(4): 387-408.
  - [ 6 ] Liu B, Liu Y K. Expected value of fuzzy variable and fuzzy expected value models [J]. *IEEE Transactions on Fuzzy Systems*, 2002, 10(4): 445-450.
  - [ 7 ] Liu B, Iwamura K. Chance constrained programming with fuzzy parameters [J]. *Fuzzy Sets and Systems*, 1998, 94(2): 227-237.
  - [ 8 ] Liu B, Iwamura K. A note on chance constrained programming with fuzzy coefficients [J]. *Fuzzy Sets and Systems*, 1998, 100(1-3): 229-233.
  - [ 9 ] Liu B. Dependent-chance programming in fuzzy environments [J]. *Fuzzy Sets and Systems*, 2000, 109(1): 97-106.
  - [ 10 ] Kiefer J, Wofwitz J. Stochastic estimation of a regression function [J]. *Annals of Mathematical Statistics*, 1952, 23: 462-466.
  - [ 11 ] Blum J R. Multidimensional stochastic approximation methods [J]. *Annals of Mathematical Statistics*, 1954, 25: 737-744.
  - [ 12 ] Spall J C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation [J]. *IEEE Transactions on Automatic Control*, 1992, 37(3): 332-341.
  - [ 13 ] Chin D C. Comparative study of stochastic algorithms for system optimization based on gradient approximations [J]. *IEEE Transactions on Systems, Man, and Cybernetics-B*, 1997, 27(2): 244-249.
  - [ 14 ] Maryak J L, Chin D C. Global random optimization by simultaneous perturbation stochastic approximation[C]. In: *Proceedings of the American Control Conference*. Arlington, VA, 2001: 756-762.
  - [ 15 ] Spall J C. Implementation of the simultaneous perturbation algorithm for stochastic optimization [J]. *IEEE Transactions on Aerospace and Electronic Systems*, 1998, 34(3): 817-823.
  - [ 16 ] Liu Y K. Convergent results about the use of fuzzy simulation in fuzzy optimization problems [J]. *IEEE Transactions on Fuzzy Systems*, 2006, 14(2): 295-304.