Journal of Uncertain Systems
Vol.2, No.3, pp.165-176, 2008
Online at: www.jus.org.uk

Monkey Algorithm for Global Numerical Optimization

Ruiqing Zhao* Wansheng Tang
Institute of Systems Engineering, Tianjin University, Tianjin 300072, China

Received 7 July 2007; Accepted 25 September 2007

Abstract

In this paper, monkey algorithm (MA) is designed to solve global numerical optimization problems with continuous variables. The algorithm mainly consists of climb process, watch-jump process, and somersault process in which the climb process is employed to search the local optimal solution, the watch-jump process to look for other points whose objective values exceed those of the current solutions so as to accelerate the monkeys' search courses, and the somersault process to make the monkeys transfer to new search domains rapidly. The proposed algorithm is applied to effectively solve the benchmark problems of global optimization with 30, 1000 or even 10000 dimensions. The computational results show that the MA can find optimal or near-optimal solutions to the problems with a large dimensions and very large numbers of local optima.

Keywords: monkey algorithm, evolution algorithm, genetic algorithm, gradient algorithm, multivariate optimization

1 Introduction

In almost all real-world optimization problems, it is necessary to use a mathematical algorithm that iteratively seeks out the solutions because an analytical solution is rarely available [16]. Therefore, many evolution algorithms such as genetic algorithm [7], ant algorithm [4] [5] and particle swarm methodology [8] have been developed and received a great deal of attention in the literature (For surveys on evolution strategies, see Back *et al* [1] and Beyer and Schwefel [2]). These algorithms have been successfully applied in various optimization areas. However, one of the disadvantages of these algorithms is that their abilities in solving optimization problems are reduced remarkably with the accretion of dimensions of the problems.

The aim of this paper is to design a new method called monkey algorithm (MA) to solve the optimization of multivariate systems. The method derives from the simulation of mountain-climbing processes of monkeys. Assume that there are many mountains in a given field (i.e., in the feasible space of the optimization problem), in order to find the highest mountaintop (i.e., find the maximal value of the objective function), monkeys will climb up from their respective positions (this action is called climb process). For each monkey, when it gets to the top of the mountain, it is natural to have a look and to find out whether there are other mountains around it higher than its present whereabout. If yes, it will jump somewhere of the mountain watched by it from the current position (this action is called watch-jump process) and then repeat the climb process until it reaches the top of the mountain. After repetitions of the climb process and the watch-jump process, each monkey will find a locally maximal mountaintop around its initial point. In order to find a much higher mountaintop, it is natural for each monkey to somersault to a new search domain (this action is called somersault process). After many repetitious iterations of the climb process, the watch-jump process, and the somersault process, the highest mountaintop found by the monkeys will be reported as an optimal value.

In optimization problems, it is well known that the number of local optima increases exponentially as the increase of the dimension of the decision vector if the objective function is a multimodal function. On the one hand, an algorithm may be trapped in the local optima of the optimization problem with large dimensions [11]. On the other hand, much CPU time will be expended since a mass of calculation. In the MA, the purpose of the somesault process is to make monkeys find new search domians and this action primely avoids running into local search. In addition, the time consumed by the MA mainly lies in using the climb process to search local optimal solutions. The essential feature of this process is the calculation of the pseudo-gradient of the objective function that only requires two measurements of the objective function regardless of the dimension of

*Corresponding author. Email: zhao@tju.edu.cn (R. Zhao).

the optimization problem. This feature allows for a significant decrease in the cost of optimization, especially in optimization problems with a large dimensions [16].

The paper is organized as follows. Section 2 describes the definition of optimization problems. The MA for the global numerical optimization is described in Section 3. In Section 4, the proposed method is applied to seek global optimal solutions to the benchmark problems with dimensions of 30, 1000 or even 10000.

2 Optimization Model

The mathematical representation of most optimization problems is the maximization (or minimization) of some objective functions with respect to a decision vector. Without loss of generality, this paper will discuss optimization in the context of maximization because a minimization problem can be trivially converted to a maximization one by changing the sign of the objective function. The general form of the global optimization model may be written as follows,

$$\begin{cases} \max \ f(\boldsymbol{x}) \\ \text{s.t.} \\ \qquad g_j(\boldsymbol{x}) \le 0, \quad j = 1, 2, \cdots, p, \end{cases} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is the decision vector in $\Re^n$, $f : \Re^n \to \Re$ the objective function, and $g_j : \Re^n \to \Re$, $j = 1, 2, \cdots, p$ the constraint functions.

Example 1 *The following optimization model was provided by Koziel and Michalewicz [10],*

$$\begin{cases} \max \quad f(x_1, x_2) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \\ s.t. \\ \qquad x_1^2 - x_2 + 1 \le 0 \\ \qquad 1 - x_1 + (x_2 - 4)^2 \le 0 \\ \qquad 0 \le x_i \le 10, \ i = 1, 2. \end{cases} \tag{2}$$

Figure 1 depicts geometrically the contour map of the objective function $f(x_1, x_2)$ in $[0, 10] \times [0, 10]$ which indicates that $f(x_1, x_2)$ is a multimodal function.

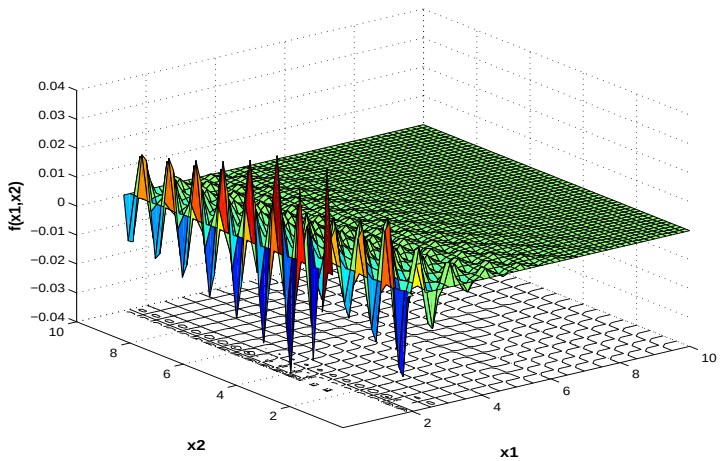

Figure 1: Graph of the objective function $f(x_1, x_2)$

3 Monkey Algorithm

In this section, an MA will be designed to solve the above-mentioned optimization problem. The main components of the algorithm, representation of solution, initialization, climb process, watch-jump process and somersault process, are presented, respectively. The details are listed as follows.

3.1 Representation of Solution

At first an integer M is defined as the population size of monkeys. And then, for the monkey i, its position is denoted as a vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$, and this position will be employed to express a solution of the optimization problem, i.e., the position $\boldsymbol{x}_i$ and the decision vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ possess the same form, $i = 1, 2, \cdots, M$, respectively. For example, in Example 1, we may take $M = 5$ and thus there are five monkeys in the population. The position of the monkey i will be denoted by a vector $\boldsymbol{x}_i = (x_{i1}, x_{i2})$ which has the same dimensions with the decision vector $\boldsymbol{x} = (x_1, x_2)$, $i = 1, 2, \cdots, 5$, respectively.

3.2 Initialization

It is necessary to initialize a position for each monkey. Here we assume that a region which contains the potential optimal solutions can be determined in advance. Usually, this region will be designed to have a nice shape, for example, an n-dimensional hypercube, because the computer can easily sample points from a hypercube. And then a point is generated randomly from the hypercube. The point will be taken as a monkey's position provided that it is feasible. Otherwise, we re-sample points from the hypercube until a feasible point will be found. Repeating the process M times, we obtain M feasible points $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ which will be employed to represent the initial positions of monkeys i, $i = 1, 2, \cdots, M$, respectively. For example, in Example 1, we can firstly make a 2-dimensional hypercube $[0, 10] \times [0, 10]$ and use the following subfunction to initialize M positions:

```
#include "stdlib.h"
for i = 1 to M do
    mark:
    for j = 1 to 2 do
        x[i][j] = 10 · rand()/(RAND_MAX−1);
    endfor
    if x[i][1]^2 − x[i][2] + 1 ≥ 0 goto mark;
    if 1 − x[i][1] + (x[i][2] − 4)^2 ≥ 0 goto mark;
    return (x[i][1], x[i][2]);
endfor
```

where rand() is a function which produces an integer between 0 and RAND_MAX−1 in "stdlib.h" of the C library and thus $10 \cdot \text{rand}()/(\text{RAND\_MAX}-1)$ will produce a real number between 0 and 10.

3.3 Climb Process

Climb process is a step-by-step procedure to change the monkeys' positions from the initial positions to new ones that can make an improvement in the objective function. The gradient-based algorithms such as Newton's method assume that information is available on the gradient vector associated with the objective function (i.e., the gradient of the objective function with respect to the decision variables). However, there has been a growing interest in recursive optimization algorithms such as simultaneous perturbation stochastic approximation (SPSA) that do not depend on direct gradient information or measurements [9] [16]. These algorithms are based on an approximation to the gradient values of the objective function. The focus in this paper is the case where the gradient information is not readily available. We use the idea of SPSA (see Spall [15] [16]) and design the climb process as follows (without loss of generality, we illustrate this process by monkey i with the position $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$, $i = 1, 2, \cdots, M$, respectively).

1). Randomly generate a vector $\boldsymbol{\Delta x}_i = (\Delta x_{i1}, \Delta x_{i2}, \cdots, \Delta x_{in})$, where

$$\Delta x_{ij} = \begin{cases} a, & \text{with probability } \frac{1}{2} \\ -a, & \text{with probability } \frac{1}{2}, \end{cases}$$

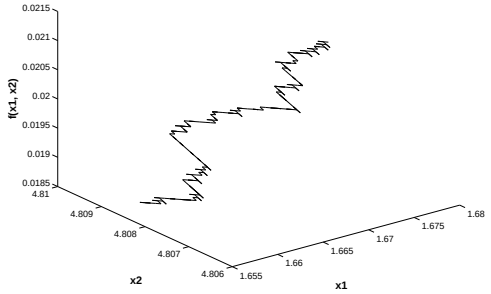

Figure 2: Climb process with $a = 0.001$

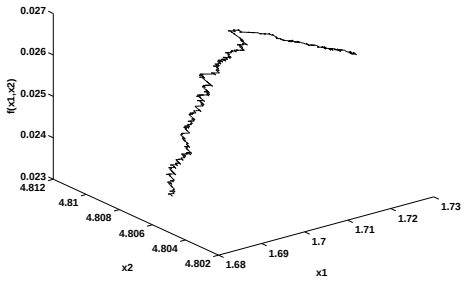

Figure 3: Climb process with $a = 0.00001$

$j = 1, 2, \cdots, n$, respectively. The parameter a ($a > 0$), called the step length of the climb process, can be determined by specific situations. For example, in Example 1, we can take $a = 0.00001$.

2). Calculate

$$f_{ij}^{'}(\boldsymbol{x}_i) = \frac{f\left(\boldsymbol{x}_i + \boldsymbol{\Delta x}_i\right) - f(\boldsymbol{x}_i - \boldsymbol{\Delta x}_i)}{2\Delta x_{ij}},$$

$j = 1, 2, \cdots, n$, respectively. The vector $f_i^{'}(\boldsymbol{x}_i) = \left(f_{i1}^{'}(\boldsymbol{x}_i), f_{i2}^{'}(\boldsymbol{x}_i), \cdots, f_{in}^{'}(\boldsymbol{x}_i)\right)$ is called the pseudo-gradient of the objective function $f(\cdot)$ at the point $\boldsymbol{x}_i$.

3). Set $y_j = x_{ij} + a \cdot \text{sign}\left(f_{ij}^{'}(\boldsymbol{x}_i)\right)$, $j = 1, 2, \cdots, n$, respectively, and let $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$.

4). Update $\boldsymbol{x}_i$ with $\boldsymbol{y}$ provided that $\boldsymbol{y}$ is feasible. Otherwise, we keep $\boldsymbol{x}_i$ unchanged.

5). Repeat steps 1) to 4) until there is little change on the values of objective function in the neighborhood iterations or the maximum allowable number of iterations (called the climb number, denoted by N_c) has been reached. The purpose that we check how the objective function changing during the iterations is to see whether convergence being achieved. This could allow the climb process to stop well before the maximum number of iterations is reached.

Remark 1 *The step length a plays a crucial role in the precision of the approximation of the local solution in the climb process. The curve in Figure 2 depicts a segment of the climb track of a monkey with the step length $a = 0.001$ while the curve in Figure 3 depicts that with $a = 0.00001$ in solving the model (2) in Example 1. From the figures, we can see that the spread of the values of the objective function reduces markedly with the decrease of the step length a. Usually, the more smaller the parameter a is, the more precise solutions are. Alternately, one can consider using a positive sequence $\{a_k\}$ which decreases to zero asymptotically as $k \to +\infty$ instead of the constant a in Step 1. In addition, using standard SPSA (see Spall [15] [16]) for the climb process is also a feasible idea.*

3.4 Watch-Jump Process

After the climb process, each monkey arrives at its own mountaintop. And then it will take a look and determine whether there are other points around it being higher than the current one. If yes, it will jump there from the current position. We define a positive parameter b as the eyesight of the monkey which indicates the maximal distance that the monkey can watch. Let us illustrate this process performed by monkey i, $i = 1, 2, \cdots, M$, respectively.

1). Randomly generate real numbers y_j from $(x_{ij} - b, x_{ij} + b)$, $j = 1, 2, \cdots, n$, respectively. Let $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$.

2). Update $\boldsymbol{x}_i$ with $\boldsymbol{y}$ provided that both $f(\boldsymbol{y}) \geq f(\boldsymbol{x}_i)$ and $\boldsymbol{y}$ is feasible. Otherwise, repeat step 1) until an appropriate point $\boldsymbol{y}$ is found. We only replace $\boldsymbol{x}_i$ with that whose function value is greater than or equal to $f(\boldsymbol{x}_i)$.

3). Repeat the climb process by employing $\boldsymbol{y}$ as an initial position.

Remark 2 *The eyesight b can be determined by specific situations. For example, we may take* $b = 0.5$ *in solving the model (2) in Example 1. Usually, the bigger the feasible space of optimal problem is, the bigger the value of b should be taken.*

3.5 Somersault Process

The main purpose of somersault process is to enable monkeys to find out new searching domains. We select the barycentre of all monkeys' current positions as a pivot. And then the monkeys will somersault along the direction pointing to the pivot. Specifically, the monkey i will somersault to the next point from its current position $\boldsymbol{x}_i$ in the following way, $i = 1, 2, \cdots, M$, respectively.

1). Randomly generate a real number α from the interval $[c, d]$ (called the somersault interval), where the somersault interval $[c, d]$ can be determined by specific situations.

2). Set

$$y_j = x_{ij} + \alpha(p_j - x_{ij}), \tag{3}$$

where $p_j = \dfrac{1}{M}\sum_{i=1}^{M} x_{ij}$, $j = 1, 2, \cdots, n$, respectively. The point $\boldsymbol{p} = (p_1, p_2, \cdots, p_n)$ is called the somersault pivot.

3). Set $\boldsymbol{x}_i = \boldsymbol{y}$ if $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ is feasible. Otherwise, repeat steps 1) and 2) until a feasible solution $\boldsymbol{y}$ is found.

Remark 3 *The somersault interval* $[c, d]$ *in the somersault process governs the maximum distance that monkeys can somersault. For example, we may take* $[c, d] = [-1, 1]$ *in Example 1. If we hope to enlarge the search space of monkeys for those problems that have large feasible spaces, we can increase the values of* $|c|$ *and d, respectively. This alteration sometimes can accelerate the convergence behavior of the MA.*

Remark 4 *In step 2, if* $\alpha \geq 0$*, monkey i will somersault along the direction from its current position pointing at the somersault pivot, otherwise, from the somersault pivot pointing at the current position.*

Remark 5 *The choice of the somersault pivot is not unique. For example, we can set* $p_j^{'} = \dfrac{1}{M-1}\left(\sum_{l=1}^{M} x_{lj} - x_{ij}\right)$, *and further replace the equation (3) with*

$$y_j = p_j^{'} + \alpha(p_j^{'} - x_{ij}) \tag{4}$$

or

$$y_j = x_{ij} + \alpha|p_j^{'} - x_{ij}|, \tag{5}$$

$j = 1, 2, \cdots, n$, *respectively.*

3.6 Termination

Following the climb process, the watch-jump process, and the somersault process, all monkeys are ready for their next actions. The MA will terminate after a given number (called the cyclic number, denoted by N) of cyclic repetitions of the above steps. We now give a flow chart for the MA in Figure 4.

It is known that the best position does not necessarily appear in the last iteration, so the best one should be kept from the beginning. If the monkeys find a better one in the new iteration, then the old one will be replaced by it. This position will be reported as an optimal solution in the end of iterations.

The experimental results found by 20 runs of the MA for solving Model (2) are listed in Table 1. From the table, we can see that the expected value of the results is 0.095824 which is very close to the result $f_{\max} = 0.095825$ obtained by Koziel and Michalewicz [10].

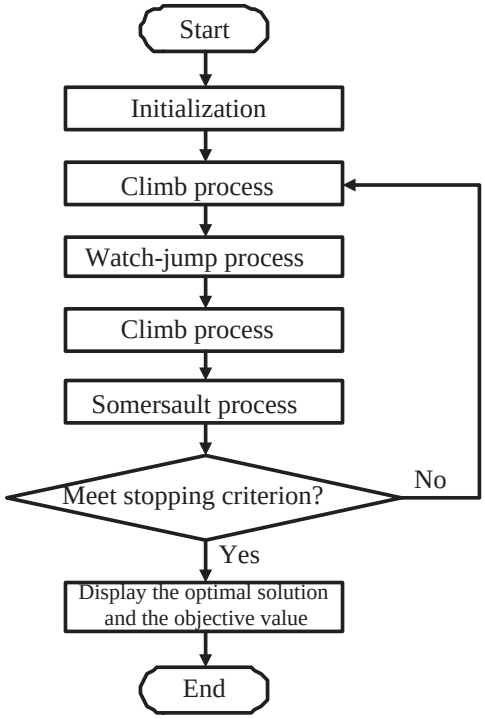

Figure 4: Flow chart for MA

Table 1: Results obtained by the MA for Model (2)

Parameters in MA	M=5, b=0.5,	N_c=2000, $[c,d]=[-1,1]$,	a=0.00001, N=10	
	0.095825	0.095823	0.095825	0.095823
	0.095822	0.095825	0.095824	0.095825
$f_{\max}$	0.095824	0.095825	0.095824	0.095825
	0.095822	0.095823	0.095821	0.095825
	0.095825	0.095825	0.095825	0.095824
The expected value $\overline{f}_{\max}$	0.095824			
The variance $\mathrm{Var}(f_{\max})$	1.57895E-12			

4 Numerical Examples

In this section, we use the benchmark functions (see Table 2) to test the effectiveness of the MA. It is well-known that some of these benchmark functions have so many local minima that they are challengeable enough for performance evaluation. These benchmark functions were tested widely by Chellapilla [3], He *et al.* [6], Tsai *et al.* [17], Tu [18] and Yao *et al.* [20] (for more details on benchmark functions, see http://www.ise.org.cn/∼benchmark).

The convergence behavior of the MA is governed by the selections of the parameters in the climb process, watch-jump process, and somersault process. These parameters used in our experiments are listed in Table 3.

The results by 20 runs of the MA for the test functions are shown in Table 4. According to the table, for the functions $f_2(\boldsymbol{x})$—$f_5(\boldsymbol{x})$ and $f_7(\boldsymbol{x})$—$f_{12}(\boldsymbol{x})$, the MA can find optimal or near-optimal solutions with much faster convergence rates.

Table 2: Test functions

Test functions	Feasible spaces	n	$f_{\min}$
$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	$[-500, 500]^n$	30	$-418.983n$
$f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-5.12, 5.12]^n$	30	0
$f_3(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$ $+20 + \exp(1)$	$[-32, 32]^n$	30	0
$f_4(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$	30	0
$f_5(\boldsymbol{x}) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i+1)$ and $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	$[-50, 50]^n$	30	0
$f_6 = \sum_{i=1}^{n}\left[\sum_{j=1}^{n}(\chi_{ij}\sin\omega_j + \psi_{ij}\cos\omega_j) - \sum_{j=1}^{n}(\chi_{ij}\sin x_j + \psi_{ij}\cos x_j)\right]^2$, where χ_{ij} and ψ_{ij} are random intergers in [-100,100], and ω_j is a random number in $[-\pi, \pi]$	$[-\pi, \pi]^n$	100	0
$f_7(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i-1)^2\right]$	$[-5, 10]^n$	30	0
$f_8(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	30	0
$f_9(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^n$	30	0
$f_{10}(\boldsymbol{x}) = \sum_{i=1}^{n}\lvert x_i \rvert + \prod_{i=1}^{n}\lvert x_i \rvert$	$[-10, 10]^n$	30	0
$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^n$	30	0
$f_{12}(\boldsymbol{x}) = \max\{\lvert x_i \rvert, \quad i = 1, 2, \ldots, n\}$	$[-100, 100]^n$	30	0

For $f_1(\boldsymbol{x})$, all the search processes of the MA in the 20 executions are trapped by poor local minima and then stagnated. One of the reasons is that the feasible space of $f_1(\boldsymbol{x})$ is too large while the somersault interval $[c, d]$ taken is too small. If we hope to enlarge the search space of monkeys for those problems that have large feasible spaces, we can increase the values of $|c|$ and d, respectively. For example, we set $[c, d] = [-10, 30]$ instead of $[c, d] = [-1, 1]$ for the test function $f_1(\boldsymbol{x})$. This alteration sometimes can avoid effectively the local search of the MA. Another reason is that the step length a and the climb number N_c are too small so that sometimes the monkeys cannot arrive at their mountaintops at all in the climb process before the maximal climber number is reached. One of choices is to increase the value of a or N_c, respectively. Increasing the value of a will possibly cut down to the precise of the solution while increasing the climb number N_c will result in much CPU time spend. Here, we replace $a = 0.001$ with $a = 0.1$ and keep the value of N_c unchanged. The revised parameters, the expected value and the variance of the results obtained by 20 runs of the MA with the revised parameters are provided in Table 5. From the table, we can see that the expected value of the results reported by the MA is -12569.1378 which is very close to the maximal value $f_{\max} = -12569.18$

Table 3: Parameters in the MA for the test functions

The population size	$M = 5$
The step length	$a = 0.001$
The climb number	$N_c = 2000$
The eyesight	$b = 0.5$
The somersault interval	$[c, d] = [-1, 1]$
The cyclic number	$N = 60$

Table 4: Results by 20 runs of the MA for the test functions

Test functions	The mean value $\overline{f}_{\min}$	The variance $\mathrm{Var}(\overline{f}_{\min})$
$f_1(\boldsymbol{x})$	−7126.7787	526102.4096
$f_2(\boldsymbol{x})$	0.0055	7.39459E-08
$f_3(\boldsymbol{x})$	0.0027	4.05818E-08
$f_4(\boldsymbol{x})$	0.0001	2.15E-11
$f_5(\boldsymbol{x})$	0.0000	5.15789E-13
$f_6(\boldsymbol{x})$	—	—
$f_7(\boldsymbol{x})$	0.0103	0.001627665
$f_8(\boldsymbol{x})$	0.0157	3.67884E-06
$f_9(\boldsymbol{x})$	0.1852	0.001201088
$f_{10}(\boldsymbol{x})$	0.0594	7.53108E-06
$f_{11}(\boldsymbol{x})$	0.0004	2.07521E-07
$f_{12}(\boldsymbol{x})$	0.0004	4.80708E-07

of $f_1(\boldsymbol{x})$.

Table 5: Results by the MA with the revised parameters for $f_1(\boldsymbol{x})$

The revised parameters	$M = 5$, $a = 0.1$ $N_c = 2000$, $b = 0.5$ $[c, d] = [-10, 30]$, $N = 60$
The expected vale $\overline{f}_1$	−12569.1378
The variance $\mathrm{Var}(\overline{f}_1)$	0.0007

The only exception is for the test function $f_6(\boldsymbol{x})$, because the MA does not fully converge when the maximum cyclic number is reached. The reason is that the large stochastic perturbations in the objective function jumble up the pseudo-gradient of the objective function, resulting in a disordered climbing direction.

In fact, it is better to set the different parameters for different test functions. The step length a in the climb process is a crucial parameter for the precision of solutions reported by the MA. If the objective functions (for example, f_2, f_3, f_4) are so sharp that the slight transfers of the decision variables will result in the tremendous changes in the objective functions, the parameter a should be set to a small value. For instance, we set $a = 10^{-6}$ for f_2, f_3, f_4 in our experiments instead of $a = 0.001$ in previous experiments.

For the eyesight b, we may follow a rule that the value of b is increased with the accretion of the size of the feasible space. For example, we set $b = 1$ in our experiments for the test functions f_2, f_3, f_5, f_7, f_9, and f_{10} since their feasible spaces are very small; while we set $b = 10$ for f_1, f_4, f_8, f_{11} and f_{12} since they have large feasible spaces.

The somersault interval $[c, d]$ in the somersault process govern the maximum distance that monkeys can somersault. Usually, we may take $[c, d] = [-1, 1]$. If we hope to enlarge the search space of monkeys for those problems that have large feasible spaces, we can increase the values of $|c|$ and d. For example, we set $[c, d] = [-10, 30]$ for f_1. This alteration sometimes can accelerate the convergence behavior of the MA.

The revised parameters and the results by the MA for all the test function are listed in Table 6 while Figures 5—15 depict the convergence processes of the MA. From Table 6 and Figures 5—15, we can see that the MA can find very well the optimal values for the test functions.

We will end this section by considering the cases in which the problem dimensions of f_2—f_5 are set as 1000 and 10000 instead of 30, respectively. The population size for the MA is also set to 5. The results obtained by the MA are shown in Tables 7 and 8, respectively. It is revealed from the tables that the MA can find optimal or near-optimal solutions to the problems. This fact indicates that the population size of the MA is almost insensitive to the dimension of the problems.

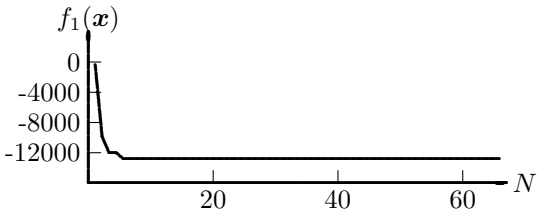

Figure 5: The convergence process for $f_1(\boldsymbol{x})$ with $n = 30$

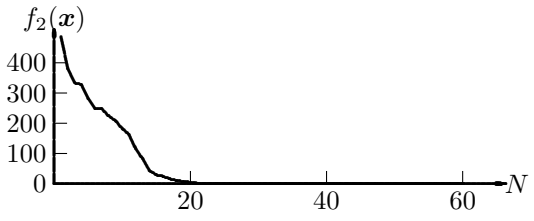

Figure 6: The convergence process for $f_2(\boldsymbol{x})$ with $n = 30$

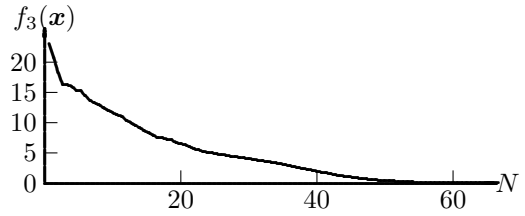

Figure 7: The convergence process for $f_3(\boldsymbol{x})$ with $n = 30$

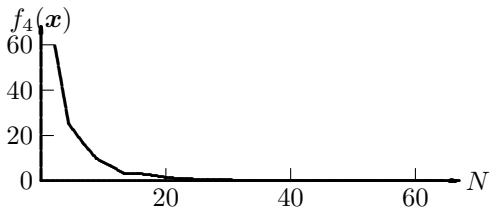

Figure 8: The convergence process for $f_4(\boldsymbol{x})$ with $n = 30$

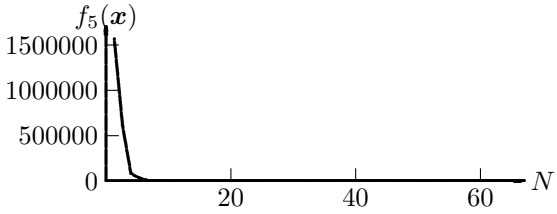

Figure 9: The convergence process for $f_5(\boldsymbol{x})$ with $n = 30$

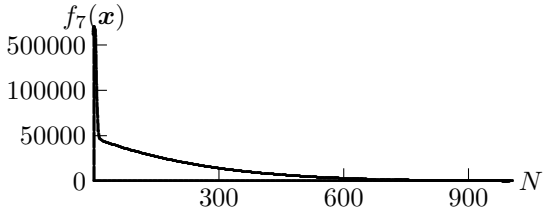

Figure 10: The convergence process for $f_7(\boldsymbol{x})$ with $n = 30$

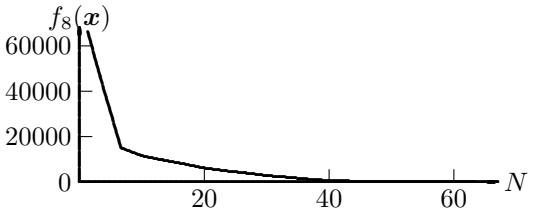

Figure 11: The convergence process for $f_8(\boldsymbol{x})$ with $n = 30$

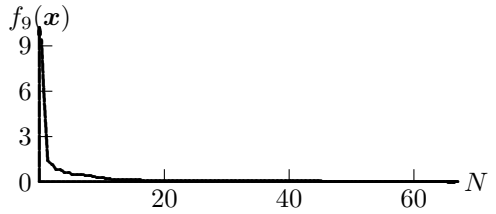

Figure 12: The convergence process for $f_9(\boldsymbol{x})$ with $n = 30$

Table 6: The revised parameters and the results for the test functions

Test functions	M	N_c	N	a	b	$[c,d]$	The function values
f_1	5	2000	60	0.1	10	$[-10,30]$	-12569.18
f_2	5	30	60	0.000001	1	$[-1,1]$	0
f_3	5	30	60	0.000001	1	$[-1,1]$	0
f_4	5	30	60	0.000001	10	$[-1,1]$	0
f_5	5	30	60	0.01	1	$[-1,1]$	0
f_6	-	-	-	-	-	-	-
f_7	5	500	1100	0.0001	1	$[-1,1]$	0
f_8	5	500	60	0.0001	10	$[-1,1]$	0
f_9	5	1000	60	0.0001	1	$[-1,1]$	0
f_{10}	5	1000	60	0.0001	1	$[-1,1]$	0
f_{11}	5	20	60	0.0001	10	$[-1,1]$	0
f_{12}	5	20	60	0.0001	10	$[-1,1]$	0

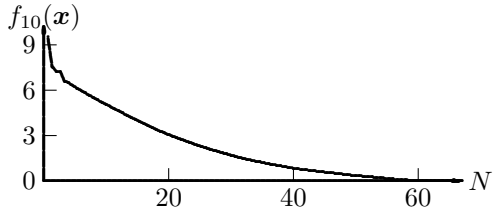

Figure 13: The convergence process for $f_{10}(\boldsymbol{x})$ with $n = 30$

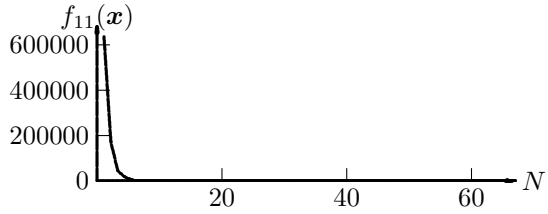

Figure 14: The convergence process for $f_{11}(\boldsymbol{x})$ with $n = 30$

5 Further Applications

The designed MA can also be used to solve varieties of uncertain programming models provided by [12] [13] [14] [19], such as

$$\begin{cases} \max\ E[f(\boldsymbol{x},\boldsymbol{\xi})] \\ \text{subject to} \\ \quad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \le 0, j = 1,2,\ldots,p, \end{cases} \tag{6}$$

where $\boldsymbol{x}$ is a decision vector and $\boldsymbol{\xi}$ is a fuzzy vector, E is the fuzzy expected operator, f is the objective function, and g_j are the constraint functions for $j = 1,2,\ldots,p$. For a given $\boldsymbol{x}$, the values of $E[f(\boldsymbol{x},\boldsymbol{\xi})]$ and $E[g_j(\boldsymbol{x},\boldsymbol{\xi})]$ can be estimated by fuzzy simulation ([12]). And then use the MA to search the optimal solutions of the model.

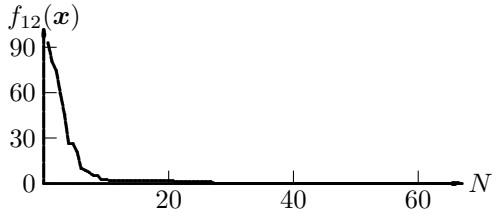

Figure 15: The convergence process for $f_{12}(\boldsymbol{x})$ with $n = 30$

Table 7: The parameters and results for test functions with dimension $n = 1000$

Test functions	M	N_c	N	a	b	$[c, d]$	The expected values	The variances
f_2	5	30	60	0.000001	1	$[-1, 1]$	0.0053401	7.78378E-08
f_3	5	30	60	0.000001	1	$[-1, 1]$	0.0085306	1.03871E-08
f_4	5	30	60	0.000001	10	$[-1, 1]$	0.0001419	2.15E-11
f_5	5	10000	150	0.005	1	$[-1, 1]$	0.0159121	6.58307E-06

Table 8: The parameters and results for test functions with dimension $n = 10000$

Test functions	M	N_c	N	a	b	$[c, d]$	The expected values	The variances
f_2	5	30	60	0.000001	1	$[-1, 1]$	0.0356499	7.91315E-07
f_3	5	30	60	0.000001	1	$[-1, 1]$	0.0102022	2.43786E-09
f_4	5	30	60	0.000001	10	$[-1, 1]$	0.0005690	2.76568E-08
f_5	5	10000	150	0.005	1	$[-1, 1]$	0.3773113	0.0067776

6 Conclusions

Monkey algorithm is a population-based algorithm, which is inspired by the mountain-climbing process of monkeys. Similar to other population-based algorithms, as one of the evolutionary algorithms, monkey algorithm can solve a variety of difficult optimization problems featuring non-linearity, non-differentiability, and high dimensionality with a faster convergence rate. Another advantage of monkey algorithm is that it has a few of the parameters to adjust, which makes it particularly easy to implement.

Acknowledgments

This work was partly supported by National Natural Science Foundation of China Grant No. 70571056, 70471049 and Program for New Century Excellent Talents in University. The authors are grateful to Prof. Ningzhong Shi, Prof. Baoding Liu, and the members of the project team for helpful discussions, but will take full responsibility for the views expressed in this paper.

References

[1] Back,T., F. Hoffmeister, and H. Schwefel, A survey of evolution strategies, *Proceedings of the Fourth International Conference of Genetic Algorithms*, San Diego, pp.2-9, 1991.

[2] Beyer, H., and H. Schwefel, Evolution strategies—A comprehensive introduction, *Natural Computing*, vol.1, pp.3C52, 2002.

[3] Chellapilla, K., Combining mutation operators in evolutionary programming, *IEEE Transactions on Evolutionary Computation*, vol.2, pp.91-96, 1998.

[4] Dorigo, M., *Optimization, learning and natural algorithms*, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[5] Dorigo, M., V. Maniezzo, and A. Colorni, The ant system: optimization by a colony of cooperative agents, *IEEE Transactions on System Man Cybernet*, vol.26, pp.29-41, 1996.

[6] He, S., Q. Wu, *et al.*, A particle swarm optimizer with passive congregation, *BioSystems*, vol.78, pp.135-147, 2004.

[7] Holland, J.H., *Adaptation in natural and artificial systems*, University of Michigan Press, 1975, Extended new Edition, MIT Press, Cambridge, 1992.

[8] Kennedy, J., and R. Eberhart, Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, vol.4, pp.1942-1948, 1995.

[9] Kiefer, J., and J. Wolfowitz, Stochastic estimation of a regression function, *Ann. Math. Stat.*, vol.23, pp.462-466, 1952.

[10] Koziel, S., and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation*, vol.7, pp.19-44, 1999.

[11] Leung, Y., and Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol.5, pp.41-53, 2001.

[12] Liu, B., *Theory and Practice of Uncertain Programming*, Physica-Verlag, Heidelberg, 2002.

[13] Liu, B., A survey of entropy of fuzzy variables, *Journal of Uncertain Systems*, vol.1, pp.4-13, 2007.

[14] Liu, B., and Y.K. Liu, Expected value of fuzzy variable and fuzzy expected value model,*IEEE Transactions on Fuzzy Systems*, vol.10, no.4, pp.445-450, 2002.

[15] Spall, J., Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Transactions on Automatic Control*, vol.37, pp.332-341, 1992.

[16] Spall, J., An overview of the simultaneous perturbation method for efficient optimization, *Johns Hopkins APL Technical Digest*, vol.19, pp.482-492, 1998.

[17] Tsai, J., T. Liu, and J. Chou, Hybrid taguchi-genetic algorithm for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol.8, pp.365-377, 2004.

[18] Tu, Z., and Y. Lu, A robust stochastic genetic algorithm (StGA) for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol.8, pp.456-470, 2004.

[19] Wang, C., W. Tang, and R. Zhao, The continuity and convexity analysis of the expected value function of a fuzzy mapping, *Journal of Uncertain Systems*, vol.1, pp.148-160, 2007.

[20] Yao, X., and Y.K. Liu, Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, vol.3, pp.82-102, 1999.