Fireworks Algorithm for Optimization

Conference Paper · June 2010

DOI: 10.1007/978-3-642-13495-1_44 · Source: DBLP

2 authors, including:

Ying Tan
Peking University
333 PUBLICATIONS **7,935** CITATIONS

Fireworks Algorithm for Optimization

Ying Tan and Yuanchun Zhu

Key Laboratory of Machine Perception (MOE), Peking University
Department of Machine Intelligence,
School of Electronics Engineering and Computer Science, Peking University,
Beijing, 100871, China
{ytan,ychzhu}@pku.edu.cn

Abstract. Inspired by observing fireworks explosion, a novel swarm intelligence algorithm, called Fireworks Algorithm (FA), is proposed for global optimization of complex functions. In the proposed FA, two types of explosion (search) processes are employed, and the mechanisms for keeping diversity of sparks are also well designed. In order to demonstrate the validation of the FA, a number of experiments were conducted on nine benchmark test functions to compare the FA with two variants of particle swarm optimization (PSO) algorithms, namely Standard PSO and Clonal PSO. It turns out from the results that the proposed FA clearly outperforms the two variants of the PSOs in both convergence speed and global solution accuracy.

Keywords: natural computing, swarm intelligence, fireworks algorithm, particle swarm optimization, function optimization.

1 Introduction

In recent years, Swarm Intelligence (SI) has become popular among researchers working on optimization problems all over the world [1,2]. SI algorithms, e.g. Particle Swarm Optimization (PSO) [3], Ant System [4], Clonal Selection Algorithm [5], and Swarm Robots [6], etc., have advantages in solving many optimization problems. Among all the SI algorithms, PSO is one of the most popular algorithm for searching optimal locations in a D-dimensional space. In 1995, Kennedy and Eberhart proposed PSO as a powerful global optimization algorithm inspired by the behavior of bird blocks [3]. Since then, the PSO has attracted the attentions of researchers around the globe, and a number of variants of PSO have been continually proposed [7,8].

Like PSO, most of swarm intelligence algorithms are inspired by some intelligent colony behaviors in nature. In this paper, inspired by the emergent swarm behavior of fireworks, a novel swarm intelligence algorithm called Fireworks Algorithm (FA) is proposed for function optimization. The FA is presented and implemented by simulating the explosion process of fireworks. In the FA, two explosion (search) processes are employed and mechanisms for keeping diversity of sparks are also well designed. To validate the performance of the proposed FA, comparison experiments were conducted on nine benchmark test functions

Y. Tan, Y. Shi, and K.C. Tan (Eds.): ICSI 2010, Part I, LNCS 6145, pp. 355–364, 2010.

among the FA, the Standard PSO (SPSO), and the Clonal PSO (CPSO) [8]. It is shown that the FA clearly outperforms the SPSO and the CPSO in both optimization accuracy and convergence speed.

The remainder of this paper is organized as follows. Section 2 describes the framework of the FA and introduces two types of search processes and mechanisms for keeping diversity. In Section 3, experimental results are presented to validate the performance of the FA. Section 4 concludes the paper.

2 Fireworks Algorithm

2.1 FA Framework

When a firework is set off, a shower of sparks will fill the local space around the firework. In our opinion, the explosion process of a firework can be viewed as a search in the local space around a specific point where the firework is set off through the sparks generated in the explosion. When we are asked to find a point x_j satisfying $f(x_j) = y$, we can continually set off 'fireworks' in potential space until one 'spark' targets or is fairly near the point x_j. Mimicking the process of setting off fireworks, a rough framework of the FA is depicted in Fig. 1.

In the FA, for each generation of explosion, we first select n locations, where n fireworks are set off. Then after explosion, the locations of sparks are obtained and evaluated. When the optimal location is found, the algorithm stops. Otherwise, n other locations are selected from the current sparks and fireworks for the next generation of explosion.

From Fig. 1, it can be seen that the success of the FA lies in a good design of the explosion process and a proper method for selecting locations, which are respectively elaborated in subsection 2.2 and subsection 2.3.

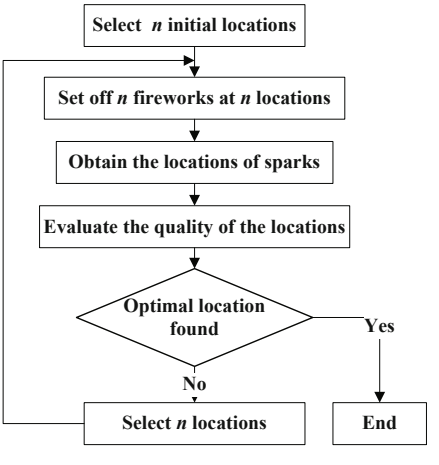
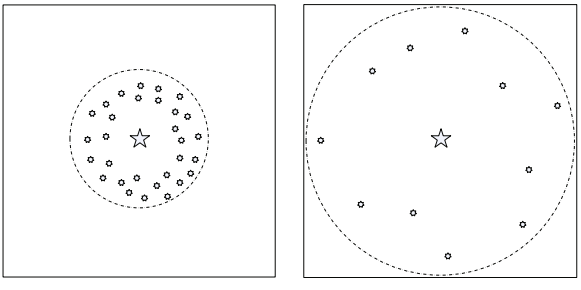


Fig. 1. Framework of fireworks algorithm

2.2 Design of Fireworks Explosion

Through observing fireworks display, we have found two specific behavior of fireworks explosion. When fireworks are well manufactured, numerous sparks are generated, and the sparks centralize the explosion center. In this case, we enjoy the spectacular display of the fireworks. However, for a bad firework explosion, quite few sparks are generated, and the sparks scatter in the space.

The two manners are depicted in Fig. 2. From the standpoint of a search algorithm, a good firework denotes that the firework locates in a promising area which may be close to the optimal location. Thus, it is proper to utilize more sparks to search the local area around the firework. In the contrast, a bad firework means the optimal location may be far from where the firework locates. Then, the search radius should be larger. In the FA, more sparks are generated and the explosion amplitude is smaller for a good firework, compared to a bad one.



(a) Good explosion (b) Bad explosion

Fig. 2. Two types of fireworks explosion

Number of Sparks. Suppose the FA is designed for the general optimization problem:

$$\text{Minimize } f(\boldsymbol{x}) \in \mathbb{R},\ \boldsymbol{x}_{\min} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\max}\ , \tag{1}$$

where $\boldsymbol{x} = x_1, x_2, \ldots, x_d$ denotes a location in the potential space, $f(\boldsymbol{x})$ is an objective function, and $\boldsymbol{x}_{\min}$ and $\boldsymbol{x}_{\max}$ denote the bounds of the potential space.

Then the number of sparks generated by each firework $\boldsymbol{x_i}$ is defined as follows.

$$s_i = m \cdot \frac{y_{\max} - f(\boldsymbol{x_i}) + \xi}{\sum_{i=1}^{n} (y_{\max} - f(\boldsymbol{x_i})) + \xi}\ , \tag{2}$$

where m is a parameter controlling the total number of sparks generated by the n fireworks, $y_{\max} = \max(f(\boldsymbol{x_i}))\ (i = 1, 2, \ldots, n)$ is the maximum (worst) value of the objective function among the n fireworks, and ξ, which denotes the smallest constant in the computer, is utilized to avoid zero-division-error.

To avoid overwhelming effects of splendid fireworks, bounds are defined for s_i, which is shown in Eq. 3.

$$\hat{s}_i = \begin{cases} round(a \cdot m) & \text{if } s_i < am \\ round(b \cdot m) & \text{if } s_i > bm \\ round(s_i) & otherwise \end{cases}, \quad a < b < 1 \ , \tag{3}$$

where a and b are const parameters.

Amplitude of Explosion. In contrast to the design of sparks number, the amplitude of a good firework explosion is smaller than that of a bad one. Amplitude of explosion for each firework is defined as follows.

$$A_i = \hat{A} \cdot \frac{f(\boldsymbol{x_i}) - y_{\min} + \xi}{\sum_{i=1}^{n} (f(\boldsymbol{x_i}) - y_{min}) + \xi} \ , \tag{4}$$

where $\hat{A}$ denotes the maximum explosion amplitude,and $y_{\min} = \min(f(\boldsymbol{x_i}))$ $(i = 1, 2, \ldots, n)$ is the minimum (best) value of the objective function among the n fireworks.

Generating Sparks. In explosion, sparks may undergo the effects of explosion from random z directions (dimensions). In the FA, we obtain the number of the affected directions randomly as follows.

$$z = round(d \cdot rand(0, 1)) \ , \tag{5}$$

where d is the dimensionality of the location $\boldsymbol{x}$, and $rand(0, 1)$ is an uniform distribution over [0,1].

The location of a spark of the firework $\boldsymbol{x_i}$ is obtained using Algorithm 1. Mimicking the explosion process, a spark's location $\tilde{\boldsymbol{x}}_j$ is first generated. Then if the obtained location is found to fall out of the potential space, it is mapped to the potential space according to the algorithm.

Algorithm 1. Obtain the location of a spark

Initialize the location of the spark: $\tilde{\boldsymbol{x_j}} = \boldsymbol{x_i}$;
z=$round(d \cdot rand(0, 1))$;
Randomly select z dimensions of $\tilde{\boldsymbol{x_j}}$;
Calculate the displacement: $h = A_i \cdot rand(-1, 1)$;
for each dimension $\tilde{x}_k^j \in$ {pre-selected z dimensions of $\tilde{\boldsymbol{x_j}}$} **do**
  $\tilde{x}_k^j = \tilde{x}_k^j + h$;
  if $\tilde{x}_k^j < x_k^{\min}$ or $\tilde{x}_k^j > x_k^{\max}$ **then**
    map $\tilde{x}_k^j$ to the potential space: $\tilde{x}_k^j = x_k^{\min} + \mid \tilde{x}_k^j \mid \%(x_k^{\max} - x_k^{\min})$;
  end if
end for

To keep the diversity of sparks, we design another way of generating sparks — Gaussian explosion, which is show in Algorithm 2. A function $Gaussian(1, 1)$, which denotes a Gaussian distribution with mean 1 and standard deviation 1, is utilized to define the coefficient of the explosion. In our experiments, $\hat{m}$ sparks of this type are generated in each explosion generation.

Algorithm 2. Obtain the location of a specific spark

Initialize the location of the spark: $\hat{\boldsymbol{x}}_{\boldsymbol{j}} = \boldsymbol{x}_{\boldsymbol{i}}$;
z=$round(d \cdot rand(0,1))$;
Randomly select z dimensions of $\hat{\boldsymbol{x}}_{\boldsymbol{j}}$;
Calculate the coefficient of Gaussian explosion: $g = Gaussian(1,1)$;
for each dimension $\hat{x}_k^j \in$ {pre-selected z dimensions of $\hat{\boldsymbol{x}}_{\boldsymbol{j}}$} **do**
  $\hat{x}_k^j = \hat{x}_k^j \cdot g$;
  if $\hat{x}_k^j < x_k^{\min}$ or $\hat{x}_k^j > x_k^{\max}$ **then**
    map $\hat{x}_k^j$ to the potential space: $\hat{x}_k^j = x_k^{\min} + \mid \hat{x}_k^j \mid \%(x_k^{\max} - x_k^{\min})$;
  end if
end for

2.3 Selection of Locations

At the beginning of each explosion generation, n locations should be selected for the fireworks explosion. In the FA, the current best location $\boldsymbol{x}^*$, upon which the objective function $f(\boldsymbol{x}^*)$ is optimal among current locations, is always kept for the next explosion generation. After that, $n-1$ locations are selected based on their distance to other locations so as to keep diversity of sparks. The general distance between a location $\boldsymbol{x}_{\boldsymbol{i}}$ and other locations is defined as follows.

$$R(\boldsymbol{x}_{\boldsymbol{i}}) = \sum_{j \in K} d(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_{\boldsymbol{j}}) = \sum_{j \in K} \parallel \boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{j}} \parallel \ , \tag{6}$$

where K is the set of all current locations of both fireworks and sparks.

Then the selection probability of a location $\boldsymbol{x}_{\boldsymbol{i}}$ is defined as follows.

$$p(\boldsymbol{x}_{\boldsymbol{i}}) = \frac{R(\boldsymbol{x}_{\boldsymbol{i}})}{\sum_{j \in K} R(\boldsymbol{x}_{\boldsymbol{j}})} \ . \tag{7}$$

When calculating the distance, any distance measure can be utilized including Manhattan distance, Euclidean distance, Angle-based distance, and so on [9]. When $d(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_{\boldsymbol{j}})$ is defined as $\mid f(\boldsymbol{x}_{\boldsymbol{i}}) - f(\boldsymbol{x}_{\boldsymbol{j}}) \mid$, the probability is equivalent to the definition of the immune density based probability in Ref. [10].

2.4 Summary

Algorithm 3 summarizes the framework of the FA. During each explosion generation, two types of sparks are generated respectively according to Algorithm 1 and Algorithm 2. For the first type, the number of sparks and explosion amplitude depend on the quality of the corresponding firework ($f(\boldsymbol{x}_{\boldsymbol{i}})$). In the contrast, the second type is generated using a Gaussian explosion process, which conducts search in a local Gaussian space around a firework. After obtaining the locations of the two types of sparks, n locations are selected for the next explosion generation. In the FA, approximate $n + m + \hat{m}$ function evaluations are done in each generation. Suppose the optimum of a function can be found in T generations, then we can deduce that the complexity of the FA is $O(T * (n + m + \hat{m}))$.

Algorithm 3. Framework of the FA

```
Randomly select n locations for fireworks;
while stop criteria=false do
  Set off n fireworks respectively at the n locations:
  for each firework x_i do
    Calculate the number of sparks that the firework yields: ŝ_i, according to Eq. 3;
    Obtain locations of ŝ_i sparks of the firework x_i using Algorithm 1;
  end for
  for k=1:m̂ do
    Randomly select a firework x_j;
    Generate a specific spark for the firework using Algorithm 2;
  end for
  Select the best location and keep it for next explosion generation;
  Randomly select n − 1 locations from the two types of sparks and the current
  fireworks according to the probability given in Eq. 7;
end while
```

3 Experiments

3.1 Benchmark Functions

To investigate the performance of the proposed FA, we conducted experiments on nine benchmark functions. The feasible bounds for all functions are set as $[-100, 100]^D$. The expression of the functions, initialization intervals and dimensionalities are listed in Table 1.

Table 1. Nine benchmark functions utilized in our experiments

Function	Expression	Initialization	D
Sphere	$F_1 = \sum_{i=1}^{D} \boldsymbol{x}_i^2$	$[30, 50]^D$	30
Rosenbrock	$F_2 = \sum_{i=1}^{D-1}(100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (\boldsymbol{x}_i - 1)^2)$	$[30, 50]^D$	30
Rastrigrin	$F_3 = \sum_{i=1}^{D}(\boldsymbol{x}_i^2 - 10\cos(2\pi\boldsymbol{x}_i) + 10)$	$[30, 50]^D$	30
Griewank	$F_4 = 1 + \sum_{i=1}^{D} \frac{\boldsymbol{x}_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{\boldsymbol{x}_i}{\sqrt{i}})$	$[30, 50]^D$	30
Ellipse	$F_5 = \sum_{i=1}^{D} 10^4 \frac{i-1}{D-1} \boldsymbol{x}_i^2$	$[15, 30]^D$	30
Cigar	$F_6 = \boldsymbol{x}_1^2 + \sum_{i=2}^{D} 10^4 \boldsymbol{x}_i^2$	$[15, 30]^D$	30
Tablet	$F_7 = 10^4 x_1^2 + \sum_{i=2}^{D} x_i^2$	$[15, 30]^D$	30
Schwefel	$F_8 = \sum_{i=1}^{D}((\boldsymbol{x}_1 - \boldsymbol{x}_i^2)^2 + (\boldsymbol{x}_i - 1)^2)$	$[15, 30]^D$	30
Ackley	$F_9 = 20 + \mathrm{e} - 20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{x}_i^2}\right)$ $-exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi \boldsymbol{x}_i^2)\right)$	$[15, 30]^D$	30

3.2 Comparison Experiments among the FA, the CPSO and the SPSO

In this section, we compare the performance of the FA with the CPSO and the SPSO in terms of both convergence speed and optimization accuracy.

Table 2. Statistical mean and standard deviation of solutions found by the FA, the CPSO and the SPSO on nine benchmark functions over 20 independent runs

Function	Function evluations	FA's mean (StD)	CPSO's mean (StD)	SPSO's mean (StD)
Sphere	500000	0.000000 (0.000000)	0.000000 (0.000000)	1.909960 (2.594634)
Rosenbrock	600000	9.569493 (12.12829)	33.403191 (42.513450)	410.522552 (529.389139)
Rastrigrin	500000	0.000000 (0.000000)	0.053042 (0.370687)	167.256119 (42.912873)
Griewank	200000	0.000000 (0.000000)	0.632403 (0.327648)	2.177754 (0.294225)
Ellipse	500000	0.000000 (0.000000)	0.000000 (0.000000)	53.718807 (68.480173)
Cigar	600000	0.000000 (0.000000)	0.000000 (0.000000)	0.002492 (0.005194)
Tablet	500000	0.000000 (0.000000)	0.000000 (0.000000)	1.462832 (1.157021)
Schwefel	600000	0.000000 (0.000000)	0.095099 (0.376619)	0.335996 (0.775270)
Ackley	200000	0.000000 (0.000000)	1.683649 (1.317866)	12.365417 (1.265322)

The parameters of both the CPSO and the SPSO are set as those in Ref. [8]. For the FA, the parameters were selected by some preliminary experiments. We found that the FA worked quite well at the setting: $n = 5$, $m = 50$, $a = 0.04$, $b = 0.8$, $\hat{A} = 40$, and $\hat{m} = 5$, which is applied in all the comparison experiments.

Table 2 depicts the optimization accuracy of the three algorithms on nine benchmark functions, which are averaged over 20 independent runs. It can be seen that the proposed FA clearly outperforms both the CPSO and SPSO on all the functions. In addition, the FA can find optimal solutions on most benchmark functions in less than 10000 function evaluations, as shown in Table 3. However, the optimization accuracy of the CPSO and the SPSO is unacceptable within 10000 function evaluations.

Besides optimization accuracy, convergence speed is quite essential to an optimizer. To validate the convergence speed of the FA, we conducted more thorough experiments. Fig. 3 depicts the convergence curves of the FA, the CPSO and the SPSO on eight benchmark functions averaged over 20 independent runs. From these results, we can arrive at a conclusion that the proposed FA has a much faster speed than the CPSO and the SPSO. From Table 3, we can find that the FA can find excellent solutions with only 10000 times of function evaluations. This also reflects the fast convergence speed of the proposed FA.

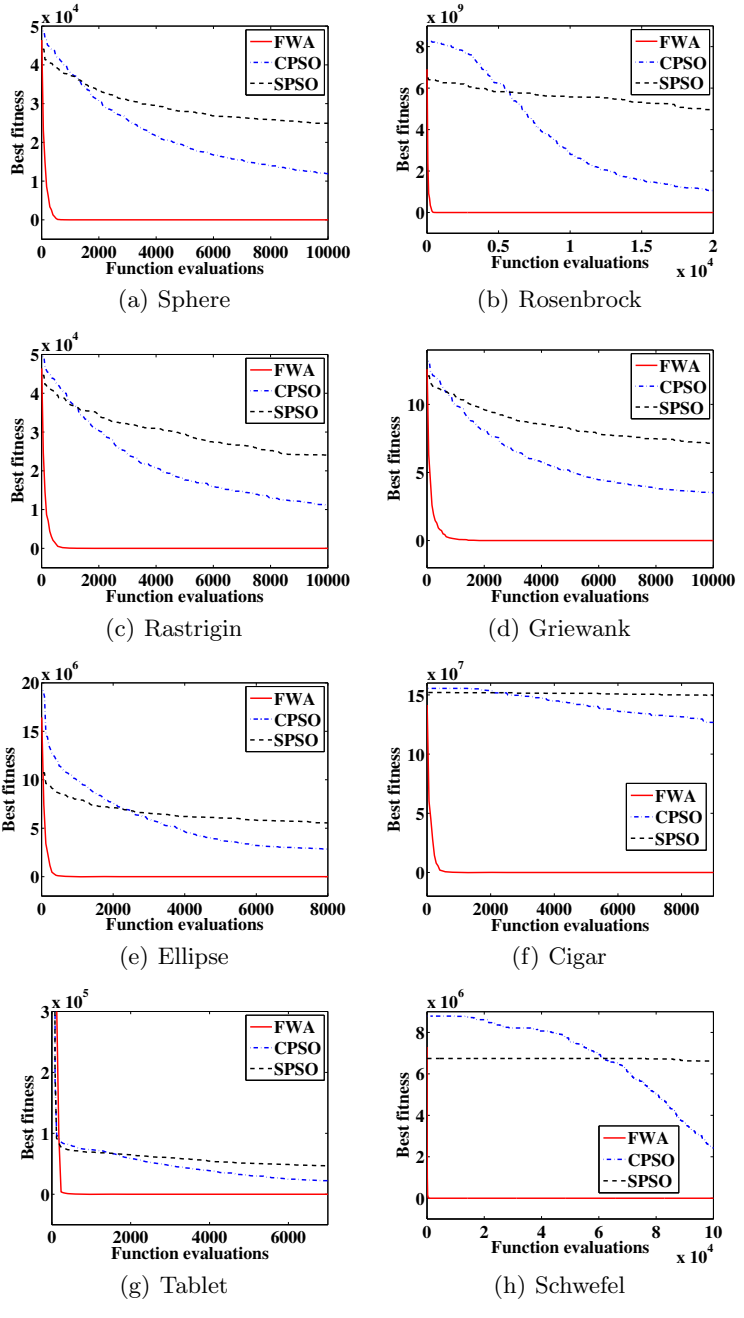


Fig. 3. Convergence curves of the FA, the CPSO and the SPSO on eight benchmark functions. The function fitness are averaged over 20 independent runs.

Table 3. Statistical mean and standard deviation of solutions found by the FA, the CPSO and the SPSO on nine benchmark functions over 20 independent runs of 10000 function evaluations

Function	FA's mean (StD)	CPSO's mean (StD)	SPSO's mean (StD)
Sphere	0.000000 (0.000000)	11857.425781 (3305.973067)	24919.099609 (3383.241523)
Rosenbrock	19.38330 (11.94373)	2750997504.000000 (1741747548.420642)	5571942400.000000 (960421617.568024)
Rastrigrin	0.000000 (0.000000)	10940.148438 (3663.484331)	24013.001953 (4246.961530)
Griewank	0.000000 (0.000000)	3.457273 (0.911027)	7.125976 (0.965788)
Ellipse	0.000000 (0.000000)	2493945.500000 (1199024.648305)	5305106.500000 (1117954.409340)
Cigar	0.000000 (0.000000)	122527168.000000 (28596381.089661)	149600864.000000 (13093322.778560)
Tablet	0.000000 (0.000000)	15595.107422 (8086.792234)	42547.488281 (8232.221882)
Schwefel	4.353733 (1.479332)	8775860.000000 (1217609.288290)	6743699.000000 (597770.084232)
Ackley	0.000000 (0.000000)	15.907665 (1.196082)	18.423347 (0.503372)

3.3 Discussion

As shown in the experiments, the FA has a faster convergence speed and a better optimization accuracy, compared to the PSO. We consider the success of the FA lies in the following two aspects.

- In the FA, sparks suffer the power of explosion from z dimensions simultaneously, and the z dimensions are randomly selected for each spark $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}$. Thus, there is a probability that the differences between the firework and the target location happen to lie in these z dimensions. In this scenario, the sparks of the firework can move towards the target location from z directions simultaneously, which endues the FA with a fast convergence speed.
- Two types of sparks are generated to keep the diversity of sparks, and the specific selection process for locations is a mechanism for keeping diversity. Therefore, the FA has the capability of avoiding premature convergence.

4 Conclusions

Mimicking the explosion process of fireworks, the so-called FA is proposed and implemented for function optimization. The experiments among the FA, the CPSO and the SPSO have shown that the proposed FA has a promising performance. It clearly outperforms the CPSO and the SPSO on nine benchmark

functions in terms of both optimization accuracy and convergence speed, which endues the FA with a promising prospect of application and extension. In future work, we will seek a deep theoretical analysis on the FA and try to apply the FA to some practical engineering applications. Finally, we intend to discuss the relationship between the FA and other general-purpose optimization algorithms.

Acknowlegements. This work was supported by National Natural Science Foundation of China (NSFC), under Grant No. 60875080 and 60673020, and partially supported by the National High Technology Research and Development Program of China (863 Program), with Grant No. 2007AA01Z453. Authors appreciate Mr. Chao Rui for his facility with the early phase of the experiments.

References

1. Garnier, S., Gautrais, J., Theraulaz, G.: The biological principles of swarm intelligence. Swarm Intelligence 1(1), 3–31 (2007)
2. Das, S., Abraham, A., Konar, A.: Swarm intelligence algorithms in bioinformatics. Studies in Computational Intelligence 94, 113–147 (2008)
3. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
4. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 26(1), 29–41 (1996)
5. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6(3), 239–251 (2002)
6. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. In: Proceedings of NATO Advanced Workshop on Robots and Biological Systems (1989)
7. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)
8. Tan, Y., Xiao, Z.M.: Clonal particle swarm optimization and its applications. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2303–2309 (2007)
9. Perlibakas, V.: Distance measures for PCA-based face recognition. Pattern Recognition Letters 25(6), 711–724 (2004)
10. Lu, G., Tan, D., Zhao, H.: Improvement on regulating definition of antibody density of immune algorithm. In: Proceedings of the 9th International Conference on Neural Information Processing, vol. 5, pp. 2669–2672 (2002)