Learning Automata-Based Fireworks Algorithm on Adaptive Assigning Sparks

Junqi Zhang^{1,2(✉)}, Lei Che^{1,2(✉)}, and Jianqing Chen^{1,2(✉)}

¹ Department of Computer Science and Technology, Tongji University, Shanghai, China

zhangjunqi@tongji.edu.cn, 623685281@qq.com, 2096536579@qq.com

² Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai, China

Abstract. Fireworks algorithm (FWA) is an emerging swarm intelligence inspired by the phenomenon of fireworks explosion. The numbers of sparks generated by fireworks have a great impact on the algorithm performance. It is widely accepted that promising fireworks should generate more sparks. However, in many researches, the quality of a firework is judged only on its current fitness value. This work proposes a Learning Automata-based Fireworks Algorithm (LA-FWA) introduced Learning automata (LA) to assign sparks for a better algorithm performance. Sparks are assigned to fireworks according to a state probability vector, which is updated constantly based on feedbacks from an environment so that it accumulates historical information. The probability vector converges as the search proceeds so that the local search ability of the LAFWA turns strong in the late search stage. Experimental results performed on CEC2013 benchmark functions show that the LAFWA outperforms several pioneering FWA variants.

Keywords: Evolutionary algorithms · Fireworks algorithm · Learning automata · Swarm intelligence

1 Introduction

Fireworks algorithm (FWA) is inspired by the phenomenon of fireworks explosion and proposed by Tan [1, 2]. Fireworks are initialized in solution space randomly in FWA and sparks are generated by the explosion process of fireworks. All fireworks and sparks are regarded as candidate solutions, and the explosion process is considered to be a stochastic search around the fireworks. The original FWA works as follows: N fireworks are initialized randomly in a search space, and their quality (the fitness value is used to represent the quality of fireworks in the original FWA) is evaluated to determine the number of sparks and explosion amplitude for all fireworks. Afterwards, the fireworks explode and generate sparks within their local space. Finally, N candidate fireworks are selected from all the fireworks and sparks as new fireworks of the next generation. The workflow continues until the termination criterion is reached.

Since FWA is raised in [1], it arouses lots of interests from researchers. The FWA has been applied to many real word optimization problems, including optimizing anti-spam model [3], solving the network reconfiguration [4], solving the path problem of vehicle congestion [5], swarm robotics [6, 7] modern web information retrieval [8], single-row facility layout problem [9], etc.

At the same time, there has been many researches attempting improving the performance of FWA. Zheng proposes the Enhanced Fireworks Algorithm (EFWA) [10], five modifications combined with conventional FWA eliminate some disadvantages of the original algorithm among it. Li proposes GFWA which puts forward a simple and efficient mutation operator called guiding vector. The Adaptive Fireworks Algorithm (AFWA) [11] uses a new adaptive amplitude calculated according to the fitness value instead of the amplitude operator in EFWA. Based on EFWA, Zheng proposes the Dynamic Search in Fireworks Algorithm (dynFWA) [12] as an improvement. In dynFWA, the firework with the smallest fitness value uses the dynamic explosion amplitude strategy. Variants mentioned above optimize the performance by adjusting the explosion amplitude adaptively. [13] proposes a fireworks algorithm based on a loser-out tournament, which also uses an independent selection operators to select fireworks for the next generation.

Learning automata (LA) [14] is a kind of machine learning algorithm which can be used as a general-purpose stochastic optimization tool. An learning automaton maintains a state probability vector where each component represents a reward probability of an action. The vector is updated through interactions with a stochastic unknown environment. LA tries to find the optimal action from a finite number of actions by applying actions to environment constantly. Environment returns a reinforcement signal which shows the relative quality of a selected action. An learning automaton receives signals and updates the vector according to its own strategy. When the termination criterion is satisfied, the optimal action is found out. So far, several PSO algorithms combined with LA have been proposed. Hashemi [15] proposes a PSO variant using LA to adaptively select parameters of PSO. A PSO variant that integrates with LA in a noisy environment is proposed by Zhang [16]. It uses LA through its unique selection mechanism to allocate re-evaluations in an adaptive manner and reduce computing resources.

Since the state probability vector of LA is updated constantly, it accumulates historical information and evaluates the quality of each action. It is more reasonable to apply learning automata to determine the number of sparks of each firework than the use of current fitness value only. On the other side, the probability vector converges gradually as the search proceeds, which leads to a strong local search ability in the late search stage.

In this paper, a Learning Automata-based Fireworks Algorithm (LAFWA) is proposed. Fireworks obtain reasonable numbers of sparks by applying LA to FWA, which leads to a competitive performance, as sparks will only be assigned to promising fireworks which brings a strong local search ability.

The rest of this paper is organized as follows. Section 2 reviews the related works of FWA and Learning Automata. Section 3 proposes the LAFWA. Experimental results based on the CEC 2013 benchmark suite are given in Sect. 4 and compared with its peers. Conclusions are drawn in Sect. 5.

2 Related Work

2.1 Fireworks Algorithm

This paper is based on GFWA [17]. In this section, GFWA will be introduced first. Without loss of generality, a minimization problem is considered as the optimization problem in this paper:

$$\min f(x) \quad (1)$$

where x is a vector in the solution space.

Explosion Strategy. GFWA follows the explosion strategy of dynFWA. In GFWA, the number of explosion sparks of each firework is calculated as following:

$$\lambda_i = \hat{\lambda} \cdot \frac{\max_j (f(X_j)) - f(X_i)}{\sum_k (\max_k (f(X_k)) - f(X_j))}, \quad (2)$$

where λ is a parameter to control the number of explosion sparks. A firework with smaller fitness value generates more sparks according to this formula. Secondly, GFWA adopts a dynamic explosion amplitude update strategy for each firework from dynFWA. The explosion amplitude of each firework is calculated as follows:

$$A_i(t) = \begin{cases} A_i(t-1) \cdot \rho^+ & \text{if } f(X_i(t)) - f(X_i(t-1)) < 0 \\ A_i(t-1) \cdot \rho^- & \text{otherwise} \end{cases} \quad (3)$$

where $A_i(t)$ and $X_i(t)$ represent the explosion amplitude and the position of i -th firework at generation t . $\rho^- \in (0, 1)$ is the reduction coefficient while $\rho^+ \in (1, +\infty)$ is the amplification coefficient. Sparks are generated uniformly within a hypercube. The explosion amplitude is the radius of the hypercube and the center of the hypercube is the firework. Algorithm 1 shows the process of sparks generated by a firework where D is the dimension, B_U and B_L are the upper and lower bounds of the search space, respectively.

Algorithm 1. Generating explosion sparks for the i -th firework

```

1: for  $j = 1$  to  $\lambda_i$  do
2:   for  $d = 1, 2, \dots, D$  do
3:      $s_{i,j}^d = X_i^d + A_i \cdot \text{rand}(-1, 1)$ 
4:     if  $s_{i,j}^d < B_L$  or  $s_{i,j}^d > B_U$  then
5:        $s_{i,j}^d = B_L + \text{rand}(0, 1) \cdot (B_U - B_L)$ 
6:     end if
7:   end for
8: end for
9: return all the  $s_{i,j}$ 

```

Guiding Vector. A mechanism called guiding vector (GV) is proposed in GFWA. A group of sparks with good quality and another group of sparks with bad quality are utilized to build a guiding vector. The GV guides a firework to move farther. Note that each firework only generates one guiding vector. The GV of i -th firework named Δ_i is calculated from its explosion sparks $s_{i,j}$ ($1 \leq j \leq \lambda_i$) as follows:

$$\Delta_i = \frac{1}{\sigma \lambda_i} \sum_{j=1}^{\sigma \lambda_i} (s_{i,j} - s_{i,\lambda_i-j+1}) \quad (4)$$

where σ is a parameter to control the proportion of adopted explosion sparks and $s_{i,j}$ means the spark of i -th firework with j -th smallest fitness value. A guiding spark (GS_i) is generated by add a GV to the i -th firework as shown in (5).

$$GS_i = X_i + \Delta_i \quad (5)$$

The main process of GFWA is described in Algorithm 2.

Algorithm 2. GFWA

```

1: Initialize  $n$  fireworks and evaluate their fitness
2: while (the stopping criterion not met) do
3:   for  $i = 1$  to  $n$  do
4:     Calculate the number of sparks by (2)
5:     Calculate the explosion amplitude by (3)
6:     Generate explosion sparks with the explosion amplitude
7:     Generate the guiding spark by (4) and (5)
8:     Evaluate all the sparks
9:     Select the best individual among the sparks, guiding spark, and the firework
       as a new firework for the next generation
10:   end for
11: end while
12: return the position and the fitness value of the best individual

```

2.2 Learning Automata

LA with a variable structure can be represented as a quadruple $\{\alpha, \beta, P, T\}$, where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is a set of actions; $\beta = \{\beta_1, \beta_2, \dots, \beta_s\}$ is a set of inputs; $P = \{p_1, p_2, \dots, p_r\}$ is a state probability vector of actions and T is a pursuit scheme to update the state probability vector, $P(t+1) = T(\alpha(t), \beta(t), P(t))$. The most popular pursuit scheme DP_{RI} is proposed in [18, 19] which increases the state probability of the estimated optimal action and decreases others. The pursuit scheme can be described as follows:

$$p_w(t+1) = \max(0, p_w(t) - \Delta) \quad w \neq i \quad (6)$$

$$p_i(t+1) = \sum p_w(t) - p_w(t+1) + p_i(t) \quad (7)$$

where the optimal action is the i -th action. Another famous pursuit scheme DGPA is proposed in [20], it increases the state probability of the actions with higher reward estimates than the current chosen action and decreases others. It can be described as follows:

$$p_w(t+1) = \max(0, p_w(t) - \Delta) \quad p_w < p_i \quad (8)$$

$$p_w(t+1) = \frac{\sum p_i(t) - p_i(t+1)}{k} + p_w(t) \quad p_w \geq p_i \quad (9)$$

where i represents i -th action selected this time and k is the number of the actions whose probability is not less than the i -th action. And Zhang [21] proposes a new pursuit scheme Last-position Elimination-based Learning Automata (LELA) inspired by a reverse philosophy. $Z(t)$ is the set of actions whose probability is not zero at time t . LELA decreases the state probability of the estimated worst action in $Z(t)$, and increases others in $Z(t)$. It can be described as follows:

$$p_w(t+1) = \max(0, p_w(t) - \Delta) \quad (10)$$

$$p_i(t+1) = \frac{p_w(t) - p_w(t+1)}{||Z(t)||} + p_w(t) \quad \forall i \neq w, i \in Z(t) \quad (11)$$

3 Learning Automata-Based Fireworks Algorithm

3.1 m-DP_{RI}

In this paper, we modify DP_{RI} to make it more suitable for our algorithm. In the classic DP_{RI} , it rewards the estimated optimal action and punishes others. However, the pursuit scheme leads to a fast convergence of the state probability vector which is harmful for the global search ability in the early search stage. In the m - DP_{RI} , the best m actions will be rewarded besides the best one. And m decreases linearly as the search progresses to enhance the local search ability gradually. The update strategy can be expressed as follows:

$$p_w(t+1) = \max(0, p_w(t) - \Delta) \quad w \in [m+1, n] \quad (12)$$

$$p_w(t+1) = \frac{\sum_{m+1}^n p_i(t) - p_i(t+1)}{m} + p_w(t) \quad w \in [1, m] \quad (13)$$

$$m = m - 1 \quad \text{if } g\% \frac{MG}{M} = 0 \quad (14)$$

where Δ is the step size, g is the generation number now, MG is maximum number of generation allowed and M is the initial number of m . The state probability vector will be sorted after the update to decide the m actions to be rewarded again.

3.2 Assigning Sparks

LA is applied to assigning sparks to fireworks according to the state probability vector in this paper. The firework with larger probability will generate more sparks. The probability vector converges as the search proceeds, the promising fireworks generates most sparks in the late search stage so that the algorithm has a strong local search ability. n probability intervals P are calculated to assign sparks by (15) based on the state probability vector p . Algorithm 3 shows how the LAFWA assigns sparks by the probability intervals P .

$$P_i = \left[\sum_1^{i-1} p_j, \sum_1^{i-1} p_j + p_i \right] \quad (15)$$

Algorithm 3. Assign SPARKS

```

1: for  $i = 1$  to  $\lambda$  do
2:   Randomly generate a number  $Rand$  in  $(0, 1)$ 
3:   Find the probability interval  $i$  where  $Rand$  in
4:    $\lambda_i = \lambda_i + 1$ 
5: end for

```

3.3 Learning Automata-Based Fireworks Algorithm

The procedure of LAFWA is given as the pseudo code shown in Algorithm 4 and explained as follows:

Step 1 Initialization: Generate the positions and velocities of n fireworks randomly. Initialize the state probability vector of assigning the sparks p evenly and step size Δ , where p represents the probability that the spark assigned to the firework and Δ represents the step size that p decreases or increases.

Step 2 Assign Sparks: Each one of the λ sparks will be assigned to n fireworks according to Algorithm 3, the firework with greater probability will generate more sparks.

Step 3 Perform Explosion: For each firework, the explosion amplitude is calculated by (3). Sparks generated uniformly within a hypercube. The explosion amplitude is the radius of the hypercube and the center of the hypercube is the firework. Generate sparks by Algorithm 1

Step 4 Generate Guiding Sparks: Generate the guiding spark by (4) and (5).

Step 5 Select Fireworks: Evaluate the fitness value of sparks and guiding vector. Select the best individual among the sparks, guiding spark, and the firework as a new firework for each firework.

Step 6 Update Probability: Update p according to (12) and (13) and sort p .

Step 7 Decrease Linearly: Complete linear decrement of m by performing (14).

Step 8 Terminal Condition Check: If any of pre-defined termination criteria is satisfied, the algorithm terminates. Otherwise, repeat from Step 2.

Algorithm 4. LAFWA

```

1: BEGIN
2: Initialize  $n$  fireworks, the initial state probability vector  $p=(1/n,...,1/n)$ , step size  $\Delta$ 
3: Evaluate the fitness value of the  $n$  fireworks
4: while stopping criterion not met do
5:   Assign sparks according to Algorithm 3
6:    $i=1$ 
7:   for  $i < n$  do
8:     Calculate the explosion amplitude according to (3).
9:     Generate the sparks by Algorithm 1
10:    Generate the guiding spark by (4) and (5)
11:    Evaluate the fitness value of sparks and guiding vector. Select the optimal one
        as the new firework
12:     $i=i+1$ 
13:   end for
14:   Update  $p$  according to (12) and (13) and sort  $p$ 
15:   Perform linear decrement of  $m$  by (14).
16: end while
17: Output the optimal firework.
18: END

```

Table 1. 28 test functions

	Function notation	Optimum (f_i^*)	Function name
Unimodal	f_1	-1400	Sphere function
	f_2	-1300	Rotated high conditioned elliptic function
	f_3	-1200	Rotated Bent Cigar function
	f_4	-1100	Rotated discus function
	f_5	-1000	Different powers function
Basic multimodal	f_6	-900	Rotated Rosenbrock's function
	f_7	-800	Rotated Schaffers F7 function
	f_8	-700	Rotated Ackley's function
	f_9	-600	Rotated Weierstrass function
	f_{10}	-500	Rotated Griewank's function
	f_{11}	-400	Rastrigin's function
	f_{12}	-300	Rotated Rastrigin's function
	f_{13}	-200	Non-continuous rotated Rastrigin's function
	f_{14}	-100	Schwefel's function
	f_{15}	100	Rotated Schwefel's function
	f_{16}	200	Rotated Katsuura function
	f_{17}	300	Lunacek Bi_Rastrigin function
	f_{18}	400	Rotated Lunacek Bi_Rastrigin function
	f_{19}	500	Expanded Griewank's plus Rosenbrock's function
	f_{20}	600	Expanded Scaffer's F6 function
Composition	f_{21}	700	Composition function 1
	f_{22}	800	Composition function 2
	f_{23}	900	Composition function 3
	f_{24}	1000	Composition function 4
	f_{25}	1100	Composition function 5
	f_{26}	1200	Composition function 6
	f_{27}	1300	Composition function 7
	f_{28}	1400	Composition function 8

*All test functions are described in [23]. Please refer to [23] and also the website http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm

4 Experimental Results and Comparisons

In this section, experiments are carried out to illustrate the advantages of LAFWA in comparison with four pioneering FWA variants.

4.1 Benchmark and Experimental Settings

Parameter setting in LAFWA is given in the following. The main parameters include:

- n : The number of fireworks.
- λ : The total number of sparks.
- ρ^- and ρ^+ : The reduction and amplification factors.
- Δ : The step size of LA.
- M : The initial number of m .

For each firework, a larger n can explore more but generate less sparks. In the proposed LAFWA, we set $n=10$ to get a good global search ability in the early stage. The reduction and amplification factors ρ^- and ρ^+ are two important parameters for dynamic search. We set the two coefficients to 0.9 and 1.2 respectively according to [12]. Δ and M is set to 0.01 and 4 according our experiments.

The experimental results are evaluated on the CEC 2013 single objective optimization benchmark suite [22], including 5 single-mode functions and 23 multi-mode functions (shown in Table 2). Standard settings are adopted for the range of parameters, such as dimensions, maximum functional evaluation numbers, and have been widely used for testing algorithms. The search ranges of all the 28 test functions are set to $[-100, 100]^D$ and D is set to 30. According to the suggestions of this benchmark suite, all the algorithms repeated for 51 times for each function and the maximal number of function evaluations in each run is $1000 \times D$. All the experiments are carried out using MATLAB R2016a on a PC with Intel(R) Core(TM) i5-8400 running at 2.80 GHz with 8G RAM.

4.2 Experimental Results and Comparison

To validate the effectiveness of LAFWA, we compare it with four pioneering FWA variants, including AFWA, dynFWA, COFFWA [24], GFWA. Parameters for these four algorithms are set to the suggested values according to their published papers. The results of the performance on the solution accuracy are listed in Table 2. Boldface indicates the best results from all listed algorithms, “Mean” is the mean results of 51 independent runs, “AR” is the average ranking of an algorithm, calculated by the sum of ranking on the 28 functions divided by the number of functions. The less AR, the better performance. LAFWA shows its outstanding convergence accuracy among all the listed FWA variants. In the unimodal part of Table 3, LAFWA performs well, achieves first 3 times among 5 functions. For the challenging multimodal and composition functions, the global optimal value is more difficult to locate. LAFWA shows its superiority and get first 15 times and second 6 times among 23 functions. The A.R. of LAFWA is 1.5 in general, which ranks the first compared with other competitors.

Table 2. LAFWA accuracy compared with other FWAs.

Func		AFWA	dynFWA	COFFWA	GFWA	LAFWA
f_1	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Rank	1	1	1	1	1
f_2	Mean	8.92E+05	7.87E+05	8.80E+05	6.96E+05	5.63E+05
	Rank	5	3	4	2	1
f_3	Mean	1.26E+08	1.57E+08	8.04E+07	3.74E+07	1.91E+07
	Rank	4	5	3	2	1
f_4	Mean	1.14E+01	1.28E+01	2.01E+03	5.00E-05	1.70E+00
	Rank	3	4	5	1	2
f_5	Mean	6.04E-04	5.42E-04	7.41E-04	1.55E-03	1.43E-03
	Rank	2	1	3	5	4
f_6	Mean	2.99E+01	3.15E+01	2.47E+01	3.49E+01	9.84E+00
	Rank	3	4	2	5	1
f_7	Mean	9.19E+01	1.03E+02	8.99E+01	7.58E+01	7.64E+01
	Rank	4	5	1	1	2
f_8	Mean	2.09E+01	2.09E+01	2.09E+01	2.09E+01	2.09E+01
	Rank	1	1	1	1	1
f_9	Mean	2.48E+01	2.56E+01	2.40E+01	1.83E+01	1.74E+01
	Rank	4	5	3	2	1
f_{10}	Mean	4.73E-02	4.20E-02	4.10E-02	6.08E-02	5.42E-03
	Rank	4	3	2	5	1
f_{11}	Mean	1.05E+02	1.07E+02	9.90E+01	7.50E+01	8.88E+01
	Rank	4	5	3	1	2
f_{12}	Mean	1.52E+02	1.56E+02	1.40E+02	9.41E+01	8.64E+01
	Rank	4	5	3	2	1
f_{13}	Mean	2.36E+02	2.44E+02	2.50E+02	1.61E+02	1.98E+02
	Rank	3	4	5	1	2
f_{14}	Mean	2.97E+03	2.95E+03	2.70E+03	3.49E+03	2.66E+03
	Rank	4	3	2	5	1
f_{15}	Mean	3.81E+03	3.71E+03	3.37E+03	3.67E+03	3.08E+03
	Rank	5	4	2	3	1
f_{16}	Mean	4.97E-01	4.77E-01	4.56E-01	1.00E-01	9.56E-02
	Rank	5	4	3	2	1
f_{17}	Mean	1.45E+02	1.48E+02	1.10E+02	8.49E+01	1.06E+02
	Rank	4	5	3	1	2
f_{18}	Mean	1.75E+02	1.89E+02	1.80E+02	8.60E+01	1.12E+02
	Rank	3	5	4	1	2
f_{19}	Mean	6.92E+00	6.87E+00	6.51E+00	5.08E+00	4.58E+00
	Rank	5	4	3	2	1
f_{20}	Mean	1.30E+01	1.30E+01	1.32E+01	1.31E+01	1.33E+01
	Rank	1	1	4	3	5
f_{21}	Mean	3.16E+02	2.92E+02	2.06E+02	2.59E+02	2.00E+02
	Rank	5	4	2	3	1
f_{22}	Mean	3.45E+03	3.41E+03	3.32E+03	4.27E+03	3.41E+03
	Rank	4	2	1	5	2
f_{23}	Mean	4.70E+03	4.55E+03	4.47E+03	4.32E+03	3.59E+03
	Rank	5	4	3	2	1
f_{24}	Mean	2.70E+02	2.72E+02	2.68E+02	2.56E+02	2.55E+02
	Rank	5	4	3	2	1
f_{25}	Mean	2.99E+02	2.97E+02	2.94E+02	2.89E+02	2.83E+02
	Rank	5	4	3	2	1
f_{26}	Mean	2.73E+02	2.62E+02	2.13E+02	2.05E+02	2.00E+02
	Rank	5	4	3	2	1
f_{27}	Mean	9.72E+02	9.92E+02	8.71E+02	8.15E+02	8.08E+02
	Rank	4	5	3	2	1
f_{28}	Mean	4.37E+02	3.40E+02	2.84E+02	3.60E+02	2.20E+02
	Rank	5	3	2	4	1
Overall	A_R	3.82	3.64	2.75	2.43	1.50

5 Conclusion

In this work, Learning Automata-based Fireworks Algorithm (LAFWA) is proposed to assign sparks to the fireworks more reasonable by using LA. The state probability vector is average so that the global search ability is well at the early search stage. As the search proceeds, the probability vector converges which leads to a strong local search ability in the late search stage. Experimental results performed on CEC2013 benchmark functions show that the LAFWA outperforms several pioneering FWA variants. Future work focuses on improving the update strategy of the state probability vector of LA.

References

1. Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Proceedings of the International Conference on Swarm Intelligence, Beijing, China, pp. 355–364, June 2010
2. Li, J., Tan, Y.: A comprehensive review of the fireworks algorithm. *ACM Comput. Surv.* **52**(6), 1–28 (2019)
3. He, W., Mi, G., Tan, Y.: Parameter optimization of local-concentration model for spam detection by using fireworks algorithm. In: Tan, Y., Shi, Y., Mo, H. (eds.) *ICSI 2013*. LNCS, vol. 7928, pp. 439–450. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38703-6_52
4. Imran, A.M., Kowsalya, M.: A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm. *Int. J. Electr. Power Energ. Syst.* **62**, 312–322 (2014)
5. Abdelaziz, M.M., Elghareeb, H.A., Ksasy, M.S.M.: Hybrid heuristic algorithm for solving capacitated vehicle routing problem. *Int. J. Comput. Technol.* **12**(9), 3845–3851 (2014)
6. Zheng, Z., Ying, T.: Group explosion strategy for searching multiple targets using swarm robotic. In: *IEEE Congress on Evolutionary Computation (CEC)* (2013)
7. Zheng, Z., Li, J., Jie, L., Ying, T.: Avoiding decoys in multiple targets searching problems using swarm robotics. In: *2014 IEEE Congress on Evolutionary Computation (CEC)* (2014)
8. Hamou, H.A., Rahmani, A., Bouarara, H.A., Amine, A.: A fireworks algorithm for modern web information retrieval with visual results mining. *Int. J. Swarm Intell. Res.* **6**(3), 1–23 (2015)
9. Liu, S., Zhang, Z., Guan, C., Zhu, L., Zhang, M., Guo, P.: An improved fireworks algorithm for the constrained single-row facility layout problem. *Int. J. Prod. Res.* 1–19 (2020)
10. Zheng, S., Janecek, A., Tan, Y.: Enhanced fireworks algorithm. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 2069–2077, June 2013
11. Li, J., Zheng, S., Tan, Y.: Adaptive fireworks algorithm. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 3214–3221, July 2014
12. Zheng, S., Janecek, A., Li, J., Tan, Y.: Dynamic search in fireworks algorithm. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 3222–3229, July 2014
13. Li, J., Tan, Y.: Loser-out tournament-based fireworks algorithm for multimodal function optimization. *IEEE Trans. Evol. Comput.* **22**(5), 679–691 (2017)
14. Narendra, K.S., Thathachar, M.A.L.: Learning automata - a survey. *IEEE Trans. Syst. Man Cybern.* **SMC-4**(4), 323–334 (1974)

15. Hashemi, A.B., Meybodi, M.R.: A note on the learning automata based algorithms for adaptive parameter selection in PSO. *Appl. Soft Comput. J.* **11**(1), 689–705 (2009)
16. Zhang, J., Xu, L., Li, J., Kang, Q., Zhou, M.: Integrating particle swarm optimization with learning automata to solve optimization problems in noisy environment. In: *Proceeding IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1432–1437, October 2014
17. Li, J., Zheng, S., Tan, Y.: The effect of information utilization: introducing a novel guiding spark in the fireworks algorithm. *IEEE Trans. Evol. Comput.* **21**(1), 153–166 (2017)
18. Oommen, B.J., Lanctot, J.K.: Discretized pursuit learning automata. *IEEE Trans. Syst. Man Cybern.* **20**(4), 931–938 (1990)
19. Oommen, B.J., Agache, M.: Continuous and discretized pursuit learning schemes: various algorithms and their comparison. *IEEE Trans. Syst. Man Cybern.* **31**(3), 277–287 (2001)
20. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: new families of continuous and discretized learning automata. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **32**(6), 738–749 (2002)
21. Zhang, J., Cheng, W., Zhou, M.: Last-position elimination-based learning automata. *IEEE Trans. Cybern.* **44**(12), 2484–2492 (2014)
22. Zambrano, H., Felipe, R.M.R.: Standard particle swarm optimization 2011 at CEC-2013: a baseline for future PSO improvements. In: *IEEE Congress on Evolutionary Computation (CEC)* (2013)
23. Liang, J.J., Qu, B.Y., Suganthan, P.N., Hernández-Díaz, A.G.: Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Zhengzhou University, China and Nanyang Technological University, Singapore, 201212, January 2013
24. Zheng, S., Li, J., Janecek, A., Tan, Y.: A cooperative framework for fireworks algorithm. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **14**(1), 27–41 (2017)