Applied Soft Computing 62 (2018) 454–462

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

The bare bones fireworks algorithm: A minimalist global optimizer☆

Junzhi Li, Ying Tan*

Key Laboratory of Machine Perception (Ministry of Education), Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, PR China

ARTICLE INFO

Article history:
Received 20 May 2017
Received in revised form 12 October 2017
Accepted 30 October 2017
Available online 7 November 2017

Keywords:
Bare bones fireworks algorithm
Swarm intelligence
Global optimization

ABSTRACT

The fireworks algorithm is a newly proposed swarm algorithm for global optimization, which adopts a novel manner of search called explosion. In this paper, we introduce a simplified version of the fireworks algorithm, where only the essential explosion operation is kept, called the bare bones fireworks algorithm. The bare bones fireworks algorithm is simple, fast and easy to implement. Sufficient conditions for local convergence are given. Experimental results on benchmark functions and real-world problems indicate that its performance is competitive and serviceable and it is extremely efficient.

1. Introduction

Meta-heuristic algorithms are a class of efficient algorithms for global optimization. In the past decades, many meta-heuristics have been proposed, such as evolution strategy [1], particle swarm optimization [2], simulated annealing [3], genetic algorithm [4], differential evolution [5], ant colony optimization [6], hunting search [7], water wave optimization [8], brain storm optimization [9] etc. The fireworks algorithm [10] is a newly proposed swarm intelligence algorithm. It conducts the global search with a novel kind of search manner called explosion. These meta-heuristic algorithms have been rapidly developed since they were proposed. Numerous algorithmic improvements have greatly promoted their performances.

However, sophisticated algorithms are sometimes unfriendly to practitioners. Some algorithms require practitioners to tune the parameters very carefully because their performance may fluctuate fiercely [11]. Some algorithms require a profound mathematical foundation of the practitioners to understand their mechanisms [12]. Some algorithms cannot be directly used in large scale applications because their computational complexities are super-linear [13]. Some algorithms are not easy to implement using low-level programming languages because matrix decomposition operations are involved [14]. These factors limit the use of these algorithms in real-world applications.

In this paper, we introduce a simplified version of the fireworks algorithm, called the bare bones fireworks algorithm (BBFWA). The BBFWA only keeps the essential explosion operation in the fireworks algorithm. All unnecessary mechanisms are removed. It has the following advantages:

1. It is extremely easy to implement.
2. It has very few parameters.
3. Its computational complexity is linear.
4. Its performance is competitive.

These properties are convenient for real-world applications. The BBFWA can be considered as a new baseline for fireworks algorithms or other meta-heuristics.

The remainder of this paper is organized as follows. In Section 2 we give a brief introduction and the background of the fireworks algorithm. In Section 3 we introduce the bare bones fireworks algorithm along with sufficient conditions for local convergence. The performance and efficiency of the BBFWA are examined in Section 4. Finally Section 5 concludes this paper.

2. Fireworks algorithm

The fireworks algorithm (FWA) is a swarm intelligence algorithm for global optimization proposed by Tan and Zhu in 2010 [10]. It conducts global search in the feasible space mainly by imitating the process of fireworks explosion. It consists of three main operators. The explosion operator generates numerous explosion sparks around the locations of the fireworks. The mutation operator mutates the locations of the fireworks to generate mutation

☆ The source code of this paper is available at http://www.cil.pku.edu.cn/research/fwa/resources/index.html.

* Corresponding author.
E-mail addresses: ljz@pku.edu.cn (J. Li), ytan@pku.edu.cn (Y. Tan).

https://doi.org/10.1016/j.asoc.2017.10.046
1568-4946/

sparks. The selection operator selects the fireworks of the new generation from the sparks in the last generation. The fireworks algorithm basically follows the general framework of evolutionary computation, which resembles the phenomenon of genetic clone, mutation and natural selection. Many scholars have placed their focus on improving the performance of the FWA on single objective functions [15–19].

Other than single objective optimization, some researchers have placed their focus on applying the FWA in multi-objective optimization problems [20,21].

It has proven useful in many real-world applications including distributed resource scheduling in smart grid [22], training of neural network for pattern classification [23], fitting of Bezier surfaces [24], parameter extraction of two diode solar PV model [25], design of hybrid sliding mode controller [26], medical data mining [27], stock price estimation [28], etc.

The FWA has also been successfully implemented on parallel platforms including CUDA [29,30] and MapReduce [31].

It is worth mentioning that the FWA with a stochastic mutation operator is globally convergent [32].

The conventional fireworks algorithm [10] consists of three main operations: explosion, Gaussian mutation and selection. And there is a mapping rule to replace out-of-boundary solutions with ones in the feasible search space. The conventional fireworks algorithm outperforms particle swarm optimization on a number of test functions. However, there are several drawbacks in the conventional fireworks algorithm. Firstly, the performance of the conventional FWA suffers dramatically when the optimal point of the objective function is shifted from the origin. This is mainly because the Gaussian mutation operation and the mapping rule are not reasonably designed [16]. Secondly, the explosion amplitude of the core firework (defined as Eq. (3)) is very close to zero. Therefore, the search capability of the core firework is wasted in the conventional fireworks algorithm. Thirdly, the distance based selection operation is very time consuming.

In the enhanced fireworks algorithm (EFWA) [16], the Gaussian mutation operation and the mapping rule are redesigned so that the performance of the algorithm becomes invariant on shifted objective functions. The explosion amplitude of the core firework is set to a value which decreases with the iteration number. The distance based selection is replaced with an elitism-random selection, which is much faster. These severe drawbacks have been eliminated in the EFWA.

The dynamic search fireworks algorithm (dynFWA) [19] is a state-of-the-art version of the fireworks algorithm. It performs much better than the conventional FWA or the EFWA on test functions. There are only two differences between the dynFWA and the EFWA. Firstly, the explosion amplitude of the core firework is dynamically controlled in the dynFWA. Secondly, the mutation operator is removed in the dynFWA because according to empirical analyses, the Gaussian mutation operator is not significantly effective in the dynFWA.

The dynFWA keeps searching for better solutions by the iteration of generating sparks around the fireworks and selection of fireworks among the sparks. Each iteration consists of the following two steps:

(1) Explosion operation: Each firework explodes and generates a certain number of explosion sparks within a certain range (explosion amplitude). The numbers of explosion sparks (Eq. (2)) and the explosion amplitudes (Eq. (4)) are calculated according to the qualities of the fireworks. The principle of the calculation is to make better fireworks generate more sparks in smaller ranges so that they can conduct exploitation and worse fireworks generate fewer sparks in larger ranges so that they can conduct exploration.

(2) Selection operation: Fireworks of the new generation are selected from the candidates including the current fireworks and sparks. In the dynFWA, the best individual among the candidates is selected as a firework of the next iteration firstly, and the other fireworks are selected from the rest of the individuals uniformly randomly.

Besides, if the optimization problem is constrained, there is a mapping rule to map the out-of-bound sparks back into the feasible space. But the mapping rule is not to be discussed in detail in this paper. We refer interested readers to a recent monograph [33].

In the following, the explosion operator of the dynFWA will be described in detail. Without loss of generality, the following minimization problem is considered in this paper:

$$\min_{\mathbf{x}\in[\mathbf{lb},\mathbf{ub}]} f(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a vector within the Euclidean space, $\mathbf{ub}$ and $\mathbf{lb}$ represent the upper bound and the lower bound vectors of the search space respectively.

For each firework $\mathbf{x}_i$, its explosion sparks' number is calculated as follows:

$$n_i = \hat{n}\cdot\frac{\max_k(f(\mathbf{x}_k)) - f(\mathbf{x}_i)}{\sum_j(\max_k(f(\mathbf{x}_k)) - f(\mathbf{x}_j))}, \tag{2}$$

where j and k are the indices of fireworks ranging from 1 to the number of fireworks, $\hat{n}$ is a constant parameter which controls the total number of explosion sparks in one generation. The way n_i is calculated in the dynFWA is similar to how the numbers of clones are determined in some artificial immune algorithms (see Figs. 3 and 6 in [34]).

In each generation, the firework with the best fitness is called the core firework (CF):

$$\mathbf{x}_{\mathrm{CF}} = arg\min_{\mathbf{x}_i}(f(\mathbf{x}_i)). \tag{3}$$

In the dynFWA, the fireworks' explosion amplitudes (except for the CF's) are calculated just as in the previous versions of the FWA:

$$A_i = \hat{A}\cdot\frac{f(\mathbf{x}_i) - f(\mathbf{x}_{\mathrm{CF}})}{\sum_j(f(\mathbf{x}_j) - f(\mathbf{x}_{\mathrm{CF}}))}, \tag{4}$$

where $\hat{A}$ is a constant parameter which controls the explosion amplitudes generally.

But the explosion amplitude of the CF is controlled dynamically. In each generation, the explosion amplitude is adjusted according to the search results in the last generation:

$$A_{\mathrm{CF,g}} = \begin{cases} C_r A_{\mathrm{CF,g-1}} & f(\mathbf{x}_{\mathrm{CF,g}}) = f(\mathbf{x}_{\mathrm{CF,g-1}}) \\ C_a A_{\mathrm{CF,g-1}} & f(\mathbf{x}_{\mathrm{CF,g}}) < f(\mathbf{x}_{\mathrm{CF,g-1}}) \end{cases} \tag{5}$$

where $A_{\mathrm{CF,g}}$ is the explosion amplitude of the CF in generation g. In the first generation, the CF is the best among all the randomly initialized fireworks, and its amplitude is preset to a constant number which is usually the diameter of the search space. After that, if in generation $g-1$, the algorithm found a better solution than the best in generation $g-2$, the amplitude of the CF will be multiplied by an amplification coefficient $C_a > 1$, otherwise it will be multiplied by a reduction coefficient $C_r < 1$. The best solution in generation $g-1$ is always selected into generation g as the CF, so the right hand conditions in Eq. (5) indicate whether the best solution found has been improved.

The core idea of this dynamic explosion amplitude is described as follows: if in one generation no better solution is found, that means the explosion amplitude is too long (aggressive) and thus needs to be reduced to increase the probability of finding a better solution, and otherwise it may be too short (conservative) to

make the largest progress and thus needs to be amplified. With the dynamic control, the algorithm can keep the amplitude appropriate for the search. That is, the dynamic explosion amplitude of the CF is long in early phases to perform exploration, and is short in late phases to perform exploitation.

Algorithm 1 shows how the explosion sparks are generated for each firework. For each firework, its sparks are generated with a uniform distribution within a hypercube around the firework.

Algorithm 1. Generating explosion sparks for $\mathbf{x}_i$.

Require: $\mathbf{x}_i$, A_i and n_i
for $j = 1$ **to** n_i **do**
  for each dimension $k = 1, 2, \ldots, d$ **do**
    sample κ from $U(0, 1)$
    if $\kappa < 0.5$ **then**
      sample η from $U(-1, 1)$
      $\mathbf{s}_{ij}^{(k)} \longleftarrow \mathbf{x}_i^{(k)} + \eta \cdot A_i$
    else
      $\mathbf{s}_{ij}^{(k)} \longleftarrow \mathbf{x}_i^{(k)}$
    end if
  end for
end for
return all the $\mathbf{s}_{ij}$

Following the conventional FWA and the EFWA, there is a dimension selection mechanism in the explosion operator (steps 3, 4, 7, 8, 9 in Algorithm 1), which ensures only about half of the explosion sparks' dimensions are distinct from the firework.

The dynFWA performs significantly better than the EFWA on a large number of test functions. However, as we will see in the rest of this paper, the dynFWA can be further improved, simplified and accelerated by removing unnecessary mechanisms.

3. Bare bones fireworks algorithm

3.1. Algorithm

The BBFWA is shown in Algorithm 2.

Algorithm 2. Bare bones fireworks algorithm.

sample $\mathbf{x} \sim U(\mathbf{lb}, \mathbf{ub})$
evaluate $f(\mathbf{x})$
$\mathbf{A} \longleftarrow \mathbf{ub} - \mathbf{lb}$
repeat
  for $i = 1$ **to** n **do**
    sample $\mathbf{s}_i \sim U(\mathbf{x} - \mathbf{A}, \mathbf{x} + \mathbf{A})$
    evaluate $f(\mathbf{s}_i)$
  end for
  if $\min(f(\mathbf{s}_i)) < f(\mathbf{x})$ **then**
    $\mathbf{x} \longleftarrow \mathrm{argmin} f(\mathbf{s}_i)$
    $\mathbf{A} \longleftarrow C_a\mathbf{A}$
  else
    $\mathbf{A} \longleftarrow C_r\mathbf{A}$
  end if
until termination criterion is met.
return $\mathbf{x}$.

lb and **ub** are the lower and upper boundaries of the search space. $\mathbf{x}$ is the location of the firework, $\mathbf{s}_i$ are the locations of explosion sparks and $\mathbf{A}$ is the explosion amplitude.

In each generation, n sparks are generated uniformly within a hyperrectangle bounded by $\mathbf{x} - \mathbf{A}$ and $\mathbf{x} + \mathbf{A}$. After that, if the best spark is a better solution than the firework, it will take the place of the firework and the explosion amplitude will be multiplied by an amplification coefficient $C_a > 1$. Otherwise, the explosion amplitude will be multiplied by a reduction coefficient $C_r < 1$ and the current firework will be kept. Note that in step 6, if a spark is located outside the boundaries, it can be replaced by a randomly chosen one in the feasible space.

The main part of the BBFWA can be implemented within about 10 lines with MATLAB. Even with low-level programming languages, it can be easily implemented as long as there is a random number generator.

There are four main differences between the BBFWA and the dynFWA.

1. The BBFWA only adopts one firework instead of multiple fireworks in the dynFWA. That is, all non-core fireworks are removed.
2. The number of sparks is no longer calculated according to Eq. (2). It is now a constant parameter n.
3. The selection operator in the BBFWA degenerates to the greedy (elite) selection.
4. The dimension selection mechanism (steps 3, 4, 7, 8, 9 in Algorithm 1) is removed.

In short, the BBFWA is a simplified version of the dynFWA. These differences make the BBFWA more clear and efficient. Therefore it is more suitable for theoretical analyses and applications. On the other hand, the BBFWA maintains important properties of the dynFWA.

3.2. Local convergence

An advantage of the greedy selection is the fitness of the firework will never suffer, which makes the algorithm robust in some sense.

Proposition 1. *In Algorithm* 1, $f(\mathbf{x})$ *is monotonic non-increasing.*

The BBFWA is not globally convergent because there is no stochastic mutation operator. This situation can be fixed easily by setting $C_r \geq 1$ or adopting a suitable mutation operator if necessary. However, on the other hand, it can be proven locally convergent with some sufficient conditions.

Consider an objective function $f(\mathbf{x})$ which is twice differentiable and there exists a point x^* such that $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is positive definite, i.e., x^* is a local minimum. Such objective functions behave like quadratic forms near the minimum [35], and a quadratic form $f(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x}$ can be reduced to $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ by a linear transformation. Therefore we only consider the spherical function in the following. The first theorem is adapted from [36].

Theorem 1. *Suppose* $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ *and* $C_a = C_r < 1$, *then* $\mathbf{x}$ *in Algorithm* 2 *converges to the minimum as long as* $C_r \geq \frac{\beta}{1-\beta}$ *and* $\beta < 0.5$ *where* $\beta = (2/\sqrt{\pi}) \sqrt[d]{1/n\Gamma(d/2+1)}$.

Proof. For the convenience of discussion, assume **lb** and **ub** are invariant in every dimension, thus $\mathbf{A}$ can be written as a scalar A. Let $\mathbf{x}_g$ and A_g be the location of the firework and the explosion amplitude in the gth generation respectively, $R_g = [\mathbf{x}_g^{(1)} - A_g, \mathbf{x}_g^{(1)} + A_g] \times \cdots \times [\mathbf{x}_g^{(d)} - A_g, \mathbf{x}_g^{(d)} + A_g]$ be the search range in the gth generation. Consider S_{α_g} is a hypersphere with center origin and radius α_g. Evidently, for any $\mathbf{x}_1 \in S_{\alpha_g}$ and $\mathbf{x}_2 \notin S_{\alpha_g}$, $f(\mathbf{x}_1) < f(\mathbf{x}_2)$. If $S_{\alpha_g} \subset R_g$, then the probability that a spark lies in S_{α_g} is $v(S_{\alpha_g})/v(R_g) = \frac{\pi^{d/2}\alpha_g^d}{\Gamma(1+d/2)(2A_g)^d}$ where v is the volume measure. Therefore, a successful search is made (at least one spark lies in the hypersphere) if $\frac{\pi^{d/2}\alpha_g^d}{\Gamma(1+d/2)(2A_g)^d} \geq O\left(\frac{1}{n}\right)$, i.e., $\alpha_g \geq \frac{2A_g \sqrt[d]{\Gamma(d/2+1)/n}}{\sqrt{\pi}} := \beta A_g$. The sequence of the radii $\{\beta A_1, \beta C_r A_1, \beta C_r^2 A_1, \ldots\}$ converges to zero since $C_r < 1$. Therefore by Cantor's intersection theorem, the sequence of the fireworks converges to the minimum.

In order to make sure $S_{\alpha_g} \subset R_g$ holds for any g, by mathematical induction, a sufficient condition is that the explosion amplitude A_g is larger than $\alpha_{g-1} + \alpha_g$ (see Fig. 1), i.e., $C_r^{g-1}A_1 \geq \beta C_r^{g-1}A_1 + \beta C_r^{g-2}A_1$. □

This theorem actually applies to any $C_a < 1$. Remark that this condition is very difficult to meet because n needs to be unreasonably large to guarantee improvement in every generation. However,

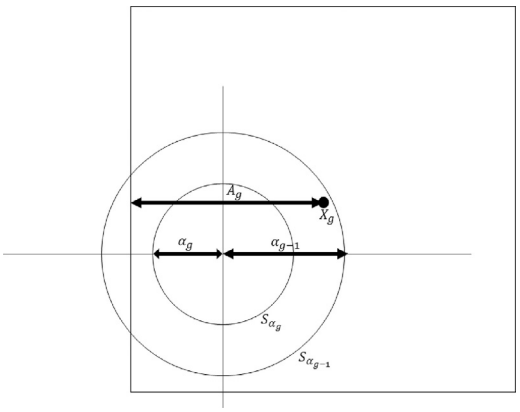

Fig. 1. An illustration of the sufficient condition in Theorem 1.

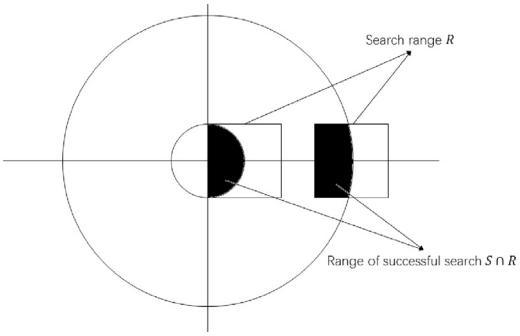

Fig. 2. Lower bound of successful probability in Theorem 2.

with the amplification coefficient $C_a > 1$, the condition can be relaxed. It is evident that $C_a > 1$ may slow down the convergence process (in exchange for better exploration), but in practice A_g converges to 0 no matter how large C_a is as long as $C_r < 1$ because the success rate converges to zero eventually given limited search range. On the contrary, large C_a is helpful to avoid premature convergence.

Theorem 2. *Suppose* $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$, $C_a > 1$, $C_r < 1$, *then* $\mathbf{x}$ *in Algorithm 2 converges to the minimum as long as* $1 - \left(1 - \frac{\pi^{d/2}}{2^{d+1}\Gamma(1+d/2)}\right)^n > \frac{-\log C_r}{\log C_a - \log C_r}$.

Proof. Let R be the search range in a certain generation. In each generation, there are two possible states: either the origin is within the search range $\mathbf{0} \in R$, or $\mathbf{0} \notin R$. In order to make sure $\mathbf{x}_g$ does not converge to any point other than the origin, a sufficient condition is the first state is recurrent. Let p be the probability of a successful search, i.e. at least one spark is better than the firework. If $C_a^p C_r^{1-p} > 1$ whenever the origin is outside of the search range, then it guarantees that the search range will be amplified whenever $\mathbf{0} \notin R$, then the first state is recurrent. Let S be a hypersphere such that its center is at the origin and the firework is on its hypersurface. When $\mathbf{0} \notin R$, $v(S \cap R)/v(R)$ has a lower bound $\frac{\pi^{d/2}}{2^{d+1}\Gamma(1+d/2)}$ which is achievable when the origin is at the center of a hypersurface of the hypercube (see Fig. 2), thus when $\mathbf{0} \notin R$, p has a lower bound $1 - \left(1 - \frac{\pi^{d/2}}{2^{d+1}\Gamma(1+d/2)}\right)^n$. □

With the amplification coefficient $C_a > 1$, the sufficient condition in Theorem 2 is looser than that in Theorem 1. For example, if $d = 2$, $C_a = 1.25$, $C_r = 0.8$, then $p \geq 1 - (1 - \pi/8)^n$, $\mathbf{x}_g$ converges to the minimum as long as $1 - (1 - \pi/8)^n \geq 1/2$, i.e., $n \geq 2$. With larger C_a and C_r, required n can be smaller. While if $d = 2$, $C_a = C_r = 0.8$, Theorem 1 requires $n \geq 7$.

Table 1
Test functions of CEC 2013 single objective optimization benchmark suite.

	No.	Name
Unimodal Functions	1	Sphere Function
	2	Rotated High Conditioned Elliptic Function
	3	Rotated Bent Cigar Function
	4	Rotated Discus Function
	5	Different Powers Function
Basic Multimodal Functions	6	Rotated Rosenbrocks Function
	7	Rotated Schaffers F7 Function
	8	Rotated Ackleys Function
	9	Rotated Weierstrass Function
	10	Rotated Griewanks Function
	11	Rastrigins Function
	12	Rotated Rastrigins Function
	13	Non-Continuous Rotated Rastrigins Function
	14	Schwefel's Function
	15	Rotated Schwefel's Function
	16	Rotated Katsuura Function
	17	Lunacek Bi_Rastrigin Function
	18	Rotated Lunacek Bi_Rastrigin Function
	19	Expanded Griewanks plus Rosenbrocks Function
	20	Expanded Scaffers F6 Function
Composition Functions	21	Composition Function 1 (Rotated)
	22	Composition Function 2 (Unrotated)
	23	Composition Function 3 (Rotated)
	24	Composition Function 4 (Rotated)
	25	Composition Function 5 (Rotated)
	26	Composition Function 6 (Rotated)
	27	Composition Function 7 (Rotated)
	28	Composition Function 8 (Rotated)

4. Experiments

4.1. The influence of parameters

There are only three parameters to be set up in the BBFWA: the number of sparks n, the amplification coefficient C_a and the reduction coefficient C_r.

Experiments are conducted on the 28 functions of CEC 2013 single objective optimization benchmark suite [37] including 5 uni-modal functions (1–5), 15 multi-modal functions (6–20) and 8 composition functions (21–28). The test functions are shown in Table 1. The dimensionality is 30 and the maximum number of function evaluations is 10000 times the dimensionality.

Different sets of parameters $\{C_a, C_r, n\}$ are evaluated for 51 independent trials. Based on $\{1.2, 0.9, 300\}$, we change one parameter each time to see the difference in the performance. The mean errors and standard deviations are shown in Table 2 and the smallest mean errors on each function are highlighted.

The comparison between the first column and the third column of Table 2 indicates that larger C_r is superior on multi-modal functions but inferior on uni-modal functions. The comparison between the second column and the third column of Table 2 indicates that larger C_a is superior on multi-modal functions but inferior on uni-modal functions. The comparison between the third column and the fourth column of Table 2 indicates that larger n is superior on multi-modal functions but inferior on uni-modal functions.

In practice, the parameters should be chosen according to the maximal number of function evaluations. Generally speaking, larger n is good at exploring and smaller n is good at exploiting because it leads to more generations. Larger C_a and C_r makes the explosion amplitude converge slowly, which is suitable when the maximal number of iterations is sufficient. Although the best values of the parameters always depend on the property of the objective function and the maximal number of function evaluations, practitioners can safely set any values that seem reasonable because from the above results it can be seen that the performance is stable in a very wide range of the parameters.

Table 2
Performances of different parameters.

F	{1.2, 0.8, 300}		{1.5, 0.9, 300}		{1.2, 0.9, 300}		{1.2, 0.9, 500}	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00E+00**	**0.00E+00**	3.50E−05	4.10E-05	**0.00E+00**	**0.00E+00**	6.20E-05	9.60E-05
2	**4.42E+05**	**1.64E+05**	1.35E+06	5.77E+05	6.28E+05	2.89E+05	1.26E+06	6.28E+05
3	7.15E+07	1.54E+08	3.61E+07	4.08E+07	**3.37E+07**	**4.91E+07**	4.17E+07	9.17E+07
4	**9.40E-05**	**1.03E-04**	3.18E-01	2.86E-01	1.23E-03	1.27E-03	1.40E-01	1.13E-01
5	**1.38E-03**	**1.54E-04**	3.91E-02	4.03E-02	2.51E-03	4.41E-04	2.51E-02	1.82E-02
6	3.50E+01	2.62E+01	4.21E+01	2.77E+01	**2.93E+01**	**2.16E+01**	4.17E+01	2.64E+01
7	8.42E+01	2.80E+01	7.13E+01	3.32E+01	7.94E+01	3.02E+01	**6.28E+01**	**2.75E+01**
8	2.10E+01	7.28E-02	**2.09E+01**	**8.33E-02**	2.09E+01	6.90E-02	2.09E+01	7.89E-02
9	2.20E+01	3.67E+00	1.92E+01	3.69E+00	1.94E+01	5.05E+00	**1.84E+01**	**3.94E+00**
10	2.69E-02	1.46E-02	2.78E-02	1.75E-02	**1.82E-02**	**1.55E-02**	2.05E-02	1.28E-02
11	1.39E+02	4.51E+01	1.24E+02	3.26E+01	1.22E+02	3.75E+01	**1.16E+02**	**3.67E+01**
12	1.30E+02	4.84E+01	**1.06E+02**	**2.90E+01**	1.20E+02	4.47E+01	1.22E+02	3.32E+01
13	2.32E+02	6.22E+01	2.12E+02	5.78E+01	2.07E+02	4.07E+01	**2.02E+02**	**4.68E+01**
14	3.66E+03	6.90E+02	3.68E+03	6.97E+02	**3.56E+03**	**6.50E+02**	3.64E+03	5.98E+02
15	3.75E+03	7.48E+02	3.49E+03	6.55E+02	3.54E+03	6.52E+02	**3.44E+03**	**6.22E+02**
16	4.13E-01	2.81E-01	**1.96E-01**	**1.27E-01**	2.68E-01	1.90E-01	2.19E-01	1.78E-01
17	1.83E+02	4.91E+01	1.70E+02	5.36E+01	**1.62E+02**	**3.86E+01**	1.70E+02	4.43E+01
18	1.90E+02	5.23E+01	1.67E+02	4.38E+01	1.74E+02	4.92E+01	**1.60E+02**	**4.70E+01**
19	6.70E+00	2.16E+00	6.70E+00	2.25E+00	6.30E+00	2.17E+00	**6.27E+00**	**1.69E+00**
20	1.36E+01	1.30E+00	1.26E+01	1.12E+00	1.29E+01	1.11E+00	**1.26E+01**	**1.33E+00**
21	3.05E+02	8.31E+01	**2.92E+02**	**7.50E+01**	2.98E+02	7.13E+01	2.95E+02	6.82E+01
22	4.64E+03	9.75E+02	4.47E+03	9.91E+02	**4.25E+03**	**7.52E+02**	4.31E+03	7.87E+02
23	4.38E+03	7.35E+02	4.11E+03	7.95E+02	4.37E+03	8.51E+02	**4.09E+03**	**7.12E+02**
24	2.65E+02	1.55E+01	**2.51E+02**	**1.36E+01**	2.58E+02	1.60E+01	2.52E+02	1.40E+01
25	2.91E+02	1.09E+01	2.81E+02	1.33E+01	2.81E+02	1.26E+01	**2.80E+02**	**1.18E+01**
26	2.03E+02	2.03E+01	2.03E+02	2.15E+01	2.03E+02	1.99E+01	**2.00E+02**	**2.47E-02**
27	8.78E+02	1.20E+02	**7.96E+02**	**9.46E+01**	8.32E+02	9.83E+01	8.02E+02	1.04E+02
28	3.00E+02	0.00E+00	**2.92E+02**	**3.92E+01**	3.39E+02	2.19E+02	3.22E+02	1.53E+02

It can be seen from Table 2 that the performances on multi-modal functions of the last three combinations are generally comparable, and the combination {1.2, 0.9, 300} is able to find the optimum on the first (sphere) test function. Therefore, its performance is considered balanced and will be compared with others in the next subsection.

4.2. Comparative performance

The results of the BBFWA with parameter {1.2, 0.9, 300} on the CEC13 benchmark suite are firstly compared with other fireworks algorithms, including the enhanced fireworks algorithm (EFWA) [16], the adaptive fireworks algorithm (AFWA) [38], the dynamic search fireworks algorithm (dynFWA) [19] and the cooperative framework fireworks algorithm (CoFFWA) [39].

The CoFFWA is also a newly proposed variant of the FWA. There are two main differences between the dynFWA and the CoFFWA: (1) In the dynFWA, the sparks generated by all fireworks are merged to form a single candidate pool in which the fireworks of the new generation will be selected. While in the CoFFWA, the successor of each firework is selected from the sparks generated by this firework. (2) There is a simple interactive mechanism called crowdedness-avoiding proposed in the CoFFWA. When a firework is too close to the CF, it will be reinitialized randomly in the search space.

All these algorithms are tested for 51 trials with dimensionality 30 and maximal evaluation number 300000. Pair-wise Wilcoxon rank sum tests with confidence level 5% are conducted between the BBFWA and each of the opponents. The numbers of functions on which the BBFWA performs significantly better (indicated by "win") and worse (indicated by "lose") than its opponent are shown in Fig. 3.

Compared with the EFWA, the AFWA and the dynFWA, the BBFWA shows significant advantages on all three kinds of objective functions. The performances of the CoFFWA and the BBFWA are comparable on multi-modal and composition functions, but the BBFWA performs better on uni-modal functions.

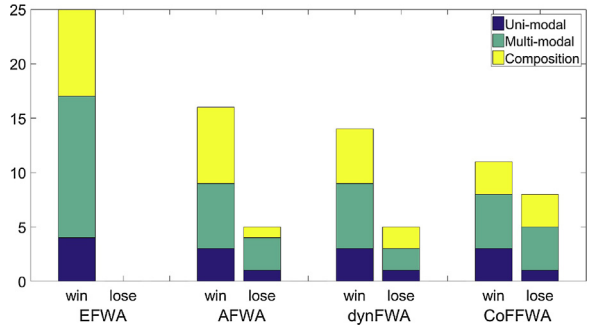

Fig. 3. Performance comparison with FWA variants.

The comparison against the dynFWA implies that the dimension selection mechanism and non-core fireworks are inefficient in the dynFWA. The dimension selection mechanism limits the diversity of explosion sparks, which is harmful to exploration. In the dynFWA, the non-core fireworks are typically not located in promising areas. Their contribution to exploration or exploitation is not comparable to the CF. Therefore, resources should not be wasted on them. On the contrary, in the BBFWA, all resources are concentrated on the CF.

The fact that the CoFFWA outperforms the dynFWA on multi-modal and composition functions indicates that the crowdedness-avoiding strategy [39] is effective in enhancing the capability of exploration. However, on the other hand, the fact that the CoFFWA does not outperform the BBFWA implies that the interaction among multiple fireworks is still not sufficient.

In summary, the simplest BBFWA achieves the state-of-the-art performance amongst FWA variants.

The results of the BBFWA are also compared with other typical meta-heuristics: standard particle swarm optimization (SPSO) 2011 [40], artificial bee colony (ABC) [41], differential evolution (DE) [5], covariance matrix adaptation evolution strategy (CMA-ES) [14], bare bones particle swarm optimization (BBPSO) [42] and

Table 3
Mean errors and standard deviations of the meta-heuristics.

	F	1	2	3	4	5	6	7
SPSO2011	Mean	0.00E+00	3.38E+05	2.88E+08	3.86E+04	5.42E-04	3.79E+01	8.79E+01
	Std.	1.88E-13	1.67E+05	5.24E+08	6.70E+03	4.91E-05	2.83E+01	2.11E+01
ABC	Mean	**0.00E+00**	6.20E+06	5.74E+08	8.75E+04	**0.00E+00**	1.46E+01	1.25E+02
	Std.	**0.00E+00**	1.62E+06	3.89E+08	1.17E+04	**0.00E+00**	4.39E+00	1.15E+01
DE	Mean	1.89E-03	5.52E+04	2.16E+06	1.32E-01	2.48E-03	7.82E+00	4.89E+01
	Std.	4.65E-04	2.70E+04	5.19E+06	1.02E-01	8.16E-04	1.65E+01	2.37E+01
CMA-ES	Mean	**0.00E+00**	**0.00E+00**	**1.41E+01**	**0.00E+00**	**0.00E+00**	**7.82E-02**	**1.91E+01**
	Std.	**0.00E+00**	**0.00E+00**	**9.96E+01**	**0.00E+00**	**0.00E+00**	**5.58E-01**	**1.18E+01**
BBPSO	Mean	**0.00E+00**	2.16E+06	3.11E+09	1.83E+04	**0.00E+00**	2.11E+01	2.28E+02
	Std.	**0.00E+00**	1.50E+06	4.76E+09	1.97E+04	**0.00E+00**	1.84E+01	8.39E+01
BBDE	Mean	2.30E+02	1.53E+07	1.06E+09	8.76E+03	4.42E+01	8.24E+01	3.93E+01
	Std.	3.44E+01	5.66E+06	5.54E+08	1.92E+03	1.11E+01	1.63E+01	1.33E+01
BBFWA	Mean	**0.00E+00**	6.28E+05	3.37E+07	1.23E-03	2.51E-03	2.93E+01	7.94E+01
	Std.	**0.00E+00**	2.89E+05	4.91E+07	1.27E-03	4.41E-04	2.16E+01	3.02E+01

	F	8	9	10	11	12	13	14
SPSO2011	Mean	**2.09E+01**	2.88E+01	3.40E-01	1.05E+02	1.04E+02	1.94E+02	3.99E+03
	Std.	**5.89E-02**	4.43E+00	1.48E-01	2.74E+01	3.54E+01	3.86E+01	6.19E+02
ABC	Mean	2.09E+01	3.01E+01	2.27E-01	**0.00E+00**	3.19E+02	3.29E+02	**3.58E-01**
	Std.	4.97E-02	2.02E+00	6.75E-02	**0.00E+00**	5.23E+01	3.91E+01	**3.91E-01**
DE	Mean	2.09E+01	**1.59E+01**	3.24E-02	7.88E+01	**8.14E+01**	**1.61E+02**	2.38E+03
	Std.	5.65E-02	**2.69E+00**	1.97E-02	2.51E+01	**3.00E+01**	**3.50E+01**	1.42E+03
CMA-ES	Mean	2.14E+01	4.81E+01	**1.78E-02**	4.00E+02	9.42E+02	1.08E+03	4.94E+03
	Std.	1.35E-01	2.48E+00	**1.11E-02**	2.49E+02	2.33E+01	6.28E+01	3.66E+02
BBPSO	Mean	2.10E+01	3.75E+01	1.82E-01	9.01E+01	1.63E+02	2.32E+02	2.04E+03
	Std.	4.71E-02	4.34E+00	1.11E-01	2.87E+01	8.20E+01	5.44E+01	4.99E+02
BBDE	Mean	2.09E+01	1.89E+01	8.30E+01	6.48E+01	1.82E+02	1.89E+02	1.74E+03
	Std.	4.95E-02	2.53E+00	3.26E+01	8.83E+00	1.37E+01	1.52E+01	2.50E+02
BBFWA	Mean	2.09E+01	1.94E+01	1.82E-02	1.22E+02	1.20E+02	2.07E+02	3.56E+03
	Std.	6.90E-02	5.05E+00	1.55E-02	3.75E+01	4.47E+01	4.07E+01	6.50E+02

	F	15	16	17	18	19	20	21
SPSO2011	Mean	3.81E+03	1.31E+00	1.16E+02	**1.21E+02**	9.51E+00	1.35E+01	3.09E+02
	Std.	6.94E+02	3.59E-01	2.02E+01	**2.46E+01**	4.42E+00	1.11E+00	6.80E+01
ABC	Mean	3.88E+03	1.07E+00	**3.04E+01**	3.04E+02	**2.62E-01**	1.44E+01	**1.65E+02**
	Std.	3.41E+02	1.96E-01	**5.15E-03**	3.52E+01	**5.99E-02**	4.60E-01	**3.97E+01**
DE	Mean	5.19E+03	1.97E+00	9.29E+01	2.34E+02	4.51E+00	1.43E+01	3.20E+02
	Std.	5.16E+02	2.59E-01	1.57E+01	2.56E+01	1.30E+00	1.19E+00	8.55E+01
CMA-ES	Mean	5.02E+03	**5.42E-02**	7.44E+02	5.17E+02	3.54E+00	1.49E+01	3.44E+02
	Std.	2.61E+02	**2.81E-02**	1.96E+02	3.52E+02	9.12E-01	3.96E-01	7.64E+01
BBPSO	Mean	7.43E+03	2.50E+00	1.13E+02	2.14E+02	6.77E+00	1.32E+01	3.18E+02
	Std.	1.28E+03	2.88E-01	2.94E+01	4.52E+01	3.36E+00	5.85E-01	7.88E+01
BBDE	Mean	6.77E+03	2.39E+00	1.00E+02	2.22E+02	1.20E+01	**1.19E+01**	4.74E+02
	Std.	3.99E+02	3.28E-01	8.90E+00	1.05E+01	1.45E+00	**4.06E-01**	4.48E+01
BBFWA	Mean	**3.54E+03**	2.68E-01	1.62E+02	1.74E+02	6.30E+00	1.29E+01	2.98E+02
	Std.	**6.52E+02**	1.90E-01	3.86E+01	4.92E+01	2.17E+00	1.11E+00	7.13E+01

	F	22	23	24	25	26	27	28
SPSO2011	Mean	4.30E+03	4.83E+03	2.67E+02	2.99E+02	2.86E+02	1.00E+03	4.01E+02
	Std.	7.67E+02	8.23E+02	1.25E+01	1.05E+01	8.24E+01	1.12E+02	4.76E+02
ABC	Mean	**2.41E+01**	4.95E+03	2.90E+02	3.06E+02	**2.01E+02**	**4.16E+02**	**2.58E+02**
	Std.	**2.81E+01**	5.13E+02	4.42E+00	6.49E+00	**1.93E-01**	**1.07E+02**	**7.78E+01**
DE	Mean	1.72E+03	5.28E+03	**2.47E+02**	2.80E+02	2.52E+02	7.64E+02	4.02E+02
	Std.	7.06E+02	6.14E+02	**1.54E+01**	1.57E+01	6.83E+01	1.00E+02	3.90E+02
CMA-ES	Mean	7.97E+03	6.95E+03	6.62E+02	4.41E+02	3.29E+02	5.39E+02	4.78E+03
	Std.	2.19E+02	3.27E+02	7.20E+01	4.00E+00	8.24E+00	7.64E+01	3.79E+02
BBPSO	Mean	2.05E+03	7.63E+03	2.99E+02	2.98E+02	3.94E+02	1.28E+03	4.75E+02
	Std.	4.33E+02	1.33E+03	4.58E+00	5.71E+00	1.24E+01	8.69E+01	5.66E+02
BBDE	Mean	1.53E+03	6.46E+03	2.53E+02	**2.72E+02**	3.00E+02	8.06E+02	7.55E+02
	Std.	3.85E+02	4.90E+02	9.66E+00	**5.97E+00**	6.83E+01	7.35E+01	2.41E+02
BBFWA	Mean	4.25E+03	**4.37E+03**	2.58E+02	2.81E+02	2.03E+02	8.32E+02	3.39E+02
	Std.	7.52E+02	**8.51E+02**	1.60E+01	1.26E+01	1.99E+01	9.83E+01	2.19E+02

bare bones differential evolution (BBDE) [43]. The results of the first three algorithms are reported in their competition papers [44–46], while the results of CMA-ES, BBPSO, and BBDE are obtained using the suggested parameters in their original papers. The mean errors[1] and standard deviations are shown in Table 3.

The mean errors of these algorithms are ranked on each test function respectively. The average ranks over 28 functions are shown in Fig. 4.

According to the average rankings, the performance of the BBFWA is the best among these algorithms, followed by DE and ABC, which is quite surprising because the mechanism of the BBFWA is the simplest. Although CMA-ES performs extremely well on uni-modal functions, apparently it suffers from premature convergence on multi-modal and composition functions. Pair-wise

[1] The mean errors of SPSO2011 are not reported in the competition paper, but its raw data can be downloaded via [47].

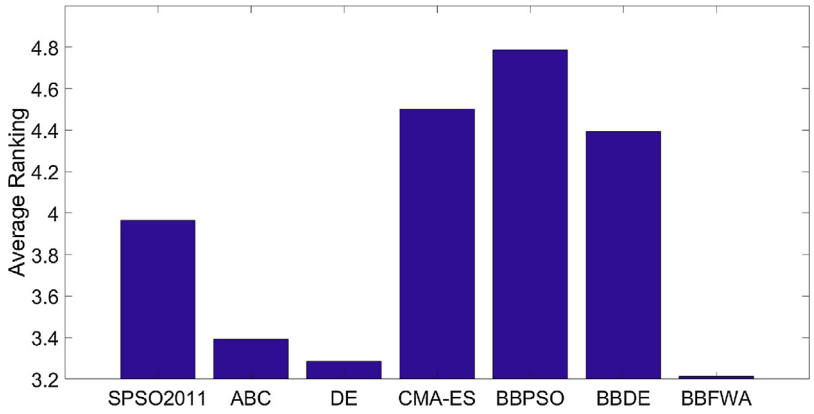

Fig. 4. Average rankings of the meta-heuristics.

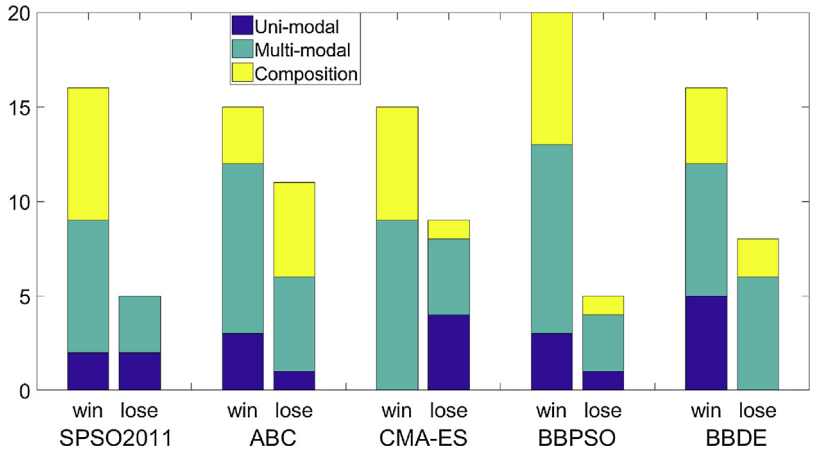

Fig. 5. Performance comparison with other meta-heuristics.

Wilcoxon rank sum tests with confidence level 5% are also conducted between the BBFWA and each of the other algorithms (except DE due to the lack of data [47]). The numbers of functions on which the BBFWA performs significantly better (indicated by "win") and worse (indicated by "lose") than its opponent are shown in Fig. 5. On uni-modal functions, the performance of the BBFWA is comparable to SPSO2011, and better than ABC but worse than CMA-ES. On multi-modal and composition functions, the BBFWA outperforms all these opponents. Generally speaking, the BBFWA shows a good balance between exploration and exploitation.

Remark that there exist some recent sophisticated variants of CMA-ES and DE that performs better than the BBFWA on the CEC13 benchmark suite. Interested readers are referred to [48] to find the results of other algorithms on this benchmark. Yet the above results indicate the performance of the BBFWA is serviceable. It should be considered as a new baseline for evolutionary and swarm algorithms.

4.3. *Efficiency*

It can be seen from Algorithm 2 that the BBFWA is a linear time algorithm. The running time is linear with both the dimensionality and the population size. Hence, the BBFWA is one of the fastest optimization algorithms in terms of the order.

To fairly compare the running time of different algorithms on different platforms, Liang et al. proposed a metric [37] as described in the following.

Let $T0$ be the time consumed by the hardware for running a set of certain calculations, which reflects the performance of the hardware platform. Let $T1$ be the time consumed for the evaluations of F14, and $\hat{T2}$ be the average time consumed by the algorithm on optimizing F14. Therefore $\hat{T2}$–$T1$ represents the pure time consumed by the algorithm itself (regardless of the objective function), and $(\hat{T2} - T1)/T0$ represents the time complexity of the algorithm (regardless of the hardware platform). Note that although this metric hardly varies with different objective functions or different platforms, it does depend on how the implementation of the algorithm is optimized.

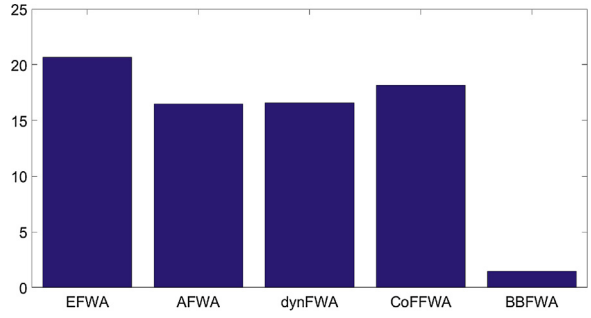

Fig. 6. Algorithm complexities of FWA variants.

Fig. 6 shows the comparison of complexities among different FWA variants. The AFWA and the dynFWA are faster than the EFWA because they replace the minimal explosion amplitude check mechanism in the EFWA with the adaptive/dynamic explosion amplitude control. The CoFFWA is slower than the dynFWA because it introduces a new interaction mechanism among fireworks. The

Table 4
Algorithm complexities of the meta-heuristics.

	$T0$	$T1$	$\hat{T2}$	$(\hat{T2} - T1)/T0$
SPSO	3.43	172.91	192.97	5.84
DE	0.11	1.68	2.47	7.15
CMA-ES	0.22	1.04	5.90	21.91
BBPSO	0.22	1.04	1.57	2.37
BBDE	0.22	1.04	2.64	7.19
BBFWA	0.22	1.04	1.37	**1.46**

Table 5
The bifunctional catalyst blend optimal control (BCBOC) problem.

	Best	Worst	Mean	Std.
GA-MPC	1.1515E-05	1.1515E-05	1.1515E-05	0.0000E+00
BBFWA	1.1515E-05	1.1515E-05	1.1515E-05	0.0000E+00

Table 6
Optimal control of a non-linear stirred tank reactor (OCNSTR).

	Best	Worst	Mean	Std.
GA-MPC	1.3771E+01	1.4329E+01	1.3815E+01	1.5460E-01
BBFWA	1.3771E+01	1.4329E+01	1.4061E+01	2.8433E-01

Table 7
Transmission network expansion planning (TNEP) problem.

	Best	Worst	Mean	Std.
GA-MPC	5.0000E-01	9.3343E-01	7.4841E-01	1.2491E-01
BBFWA	5.0000E-01	9.4332E-01	7.3546E-01	1.0651E-01

BBFWA is by far the fastest fireworks algorithm because it removes the non-core fireworks, the dimension selection mechanism and the mutation operator.

Table 4 shows the time complexity comparison with other algorithms.[2]

PSO, DE and the BBFWA are all linear time algorithms, while CMA-ES is a quadratic time algorithm which will become very slow when the dimensionality increases [14]. According to this metric, the BBFWA runs faster than PSO and DE.

4.4. Real-world problems

In this section, three real-world optimization problems in [49] are employed to test the utility of the BBFWA. Following the instructions of [49], the algorithm is run 25 times independently and the maximal number of evaluations in each run is set to 150000. The parameters of the BBFWA are identical with in the previous section, except that C_r is set to 0.97 because the number of function evaluations is relatively more sufficient here in regard with the problem dimensionalities. The results of the winner of CEC2011 competition on real-world numerical optimization problems, GA with a New Multi-Parent Crossover (GA-MPC) [50], are adopted here for comparisons. The losses after 150000 evaluations on the three problems are shown in Tables 5–7.

On the BCBOC problem, both algorithms succeeded in finding the optimal solution. On the OCNSTR problem, the GA-MPC performs slightly better than the BBFWA in terms of the mean loss. On the TNEP problem, the mean loss of the BBFWA is smaller, but the worst loss is larger. Generally speaking, the performance of the BBFWA is comparable to state-of-the-art methods on these real-world problems.

[2] This index of ABC is not given in its report.

5. Conclusion

Some researchers have pointed out the embarrassment of the field of evolutionary computation [51]: the gap between the theory and the practice has been growing. Many sophisticated algorithms have been proposed or invented in these years, but in contrast, their applications are quite limited. We believe one main reason is that these sophisticated algorithms are not friendly to practitioners. Most practitioners are not interested in the metaphors or mathematical foundations of the algorithms. On the opposite, they may find these algorithms hard to understand and hard to implement.

Thus, in this paper, we introduce an extremely simple but efficient meta-heuristic algorithm for continuous global optimization, called the bare bones fireworks algorithm (BBFWA). It adopts only the essential operators of the fireworks algorithm and it has only three parameters to set up. The mechanism of this algorithm is easy to understand. The time complexity of this algorithm is linear. It can be easily implemented on a variety of platforms, which is convenient for real-world applications. Experimental results on a standard benchmark and real-world problems indicate that the performance of the BBFWA is serviceable and stable.

"The more generic an algorithm, the more its utility and more research efforts should be spent on such a procedure to make it better [52]". We believe the BBFWA can be a useful tool for practitioners who are not familiar with meta-heuristic optimization algorithms. It can also serve as a new start point for designing new metaheuristics.

Acknowledgements

The authors would like to thank Shaoqiu Zheng, Jie Li and Manxin Li for valuable discussions, and Professor Mahamed G.H. Omran for providing the source code of BBDE.

This work was supported by the Natural Science Foundation of China (NSFC) under grant nos. 61375119 and 61673025 and supported by Beijing Natural Science Foundation (4162029), and partially supported by National Key Basic Research Development Plan (973 Plan) Project of China under grant no. 2015CB352302.

References

[1] H.G. Beyer, H.P. Schwefel, Evolution strategies – a comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52.
[2] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 760–766.
[3] C.-R. Hwang, Simulated annealing: theory and applications, Acta Appl. Math. 12 (1) (1988) 108–111.
[4] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99.
[5] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[6] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolut. Comput. 1 (1) (1997) 53–66, http://dx.doi.org/10.1109/4235.585892, ISSN 1089-778X.
[7] R. Oftadeh, M. Mahjoob, M. Shariatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search, Comput. Math. Appl. 60 (7) (2010) 2087–2098, http://dx.doi.org/10.1016/j.camwa.2010.07.049, ISSN 0898-1221, http://www.sciencedirect.com/science/article/pii/S0898122110005419.
[8] Y.-J. Zheng, Water wave optimization: a new nature-inspired metaheuristic, Comput. Oper. Res. 55 (2015) 1–11, http://dx.doi.org/10.1016/j.cor.2014.10.008, ISSN 0305-0548, http://www.sciencedirect.com/science/article/pii/S0305054814002652.
[9] S. Cheng, Q. Qin, J. Chen, Y. Shi, Brain storm optimization algorithm: a review, Artif. Intell. Rev. 46 (4) (2016) 445–458.
[10] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in: Advances in Swarm Intelligence, Springer, 2010, pp. 355–364.
[11] C.W. Cleghorn, A.P. Engelbrecht, Particle swarm convergence: an empirical investigation, 2014 IEEE Congress on Evolutionary Computation (CEC) (2014) 2524–2530, http://dx.doi.org/10.1109/CEC.2014.6900439, ISSN 1089-778X.
[12] P. Hennig, C.J. Schuler, Entropy search for information-efficient global optimization, J. Mach. Learn. Res. 13 (9) (2012) 1809–1837.

[13] R. Mallipeddi, M. Lee, An evolving surrogate model-based differential evolution algorithm, Appl. Soft Comput. 34 (Supplement C) (2015) 770–787, http://dx.doi.org/10.1016/j.asoc.2015.06.010, ISSN 1568-4946, http://www.sciencedirect.com/science/article/pii/S1568494615003592.
[14] N. Hansen, The CMA evolution strategy: a comparing review, Stud. Fuzziness Soft Comput. 192 (2006) 75–102.
[15] J. Liu, S. Zheng, Y. Tan, The improvement on controlling exploration and exploitation of firework algorithm, in: Advances in Swarm Intelligence, Springer, 2013, pp. 11–23.
[16] S. Zheng, A. Janecek, Y. Tan, Enhanced fireworks algorithm, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 2069–2077.
[17] B. Zhang, M. Zhang, Y.-J. Zheng, Improving enhanced fireworks algorithm with new Gaussian explosion and population selection strategies, in: Advances in Swarm Intelligence, Springer, 2014, pp. 53–63.
[18] C. Yu, L. Kelley, S. Zheng, Y. Tan, Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 3238–3245.
[19] S. Zheng, A. Janecek, J. Li, Y. Tan, Dynamic search in fireworks algorithm, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 3222–3229.
[20] Y.-J. Zheng, Q. Song, S.-Y. Chen, Multiobjective fireworks optimization for variable-rate fertilization in oil crop production, Appl. Soft Comput. 13 (11) (2013) 4253–4263.
[21] L. Liu, S. Zheng, Y. Tan, S-metric based multi-objective fireworks algorithm, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 1257–1264.
[22] K.S. Reddy, L.K. Panwar, R. Kumar, B.K. Panigrahi, Distributed resource scheduling in smart grid with electric vehicle deployment using fireworks algorithm, J. Mod. Power Syst. Clean Energy 4 (2) (2016) 188–199.
[23] A.L. Bolaji, A.A. Ahmad, P.B. Shola, Training of neural network for pattern classification using fireworks algorithm, Int. J. Syst. Assur. Eng. Manag. (2016) 1–8, http://dx.doi.org/10.1007/s13198-016-0526-z, ISSN 0976-4348.
[24] K.S. Reddy, A. Mandal, K.K. Verma, G. Rajamohan, Fitting of Bezier surfaces using the fireworks algorithm, Int. J. Adv. Eng. Technol. 9 (3) (2016) 421.
[25] T.S. Babu, J.P. Ram, K. Sangeetha, A. Laudani, N. Rajasekar, Parameter extraction of two diode solar PV model using fireworks algorithm, Sol. Energy 140 (2016) 265–276.
[26] T.-J. Su, S.-M. Wang, T.-Y. Li, S.-T. Shih, V.-M. Hoang, Design of hybrid sliding mode controller based on fireworks algorithm for nonlinear inverted pendulum systems, Adv. Mech. Eng. 9 (1) (2016), 1687814016684273.
[27] R.K. Dutta, N.K. Karmakar, T. Si, Artificial neural network training using fireworks algorithm in medical data mining, Int. J. Comput. Appl. 137 (1) (2016) 1–5, published by Foundation of Computer Science (FCS), NY, USA.
[28] K.T. Tung, N.T.B. Loan, et al., Applying artificial neural network optimized by fireworks algorithm for stock price estimation, ICTACT J. Soft Comput. 6 (3) (2016).
[29] K. Ding, S. Zheng, Y. Tan, A GPU-based parallel fireworks algorithm for optimization, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, ACM, 2013, pp. 9–16.
[30] K. Ding, Y. Tan, Attract-repulse fireworks algorithm and its CUDA implementation using dynamic parallelism, Int. J. Swarm Intell. Res. 6 (2) (2015) 1–31.
[31] S.A. Ludwig, D. Dawar, Parallelization of enhanced firework algorithm using MapReduce, Int. J. Swarm Intell. Res. 6 (2) (2015) 32–51.
[32] J. Liu, S. Zheng, Y. Tan, Analysis on global convergence and time complexity of fireworks algorithm, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 3207–3213.
[33] Y. Tan, Fireworks Algorithm (FWA), Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 17–35, http://dx.doi.org/10.1007/978-3-662-46353-6_2, ISBN 978-3-662-46353-6.
[34] A. Watkins, J. Timmis, L. Boggess, Artificial immune recognition system (AIRS): an immune-inspired supervised learning algorithm, Genet. Progr. Evol. Mach. 5 (3) (2004) 291–317, http://dx.doi.org/10.1023/B:GENP.0000030197.83685.94, ISSN 1573-7632.
[35] P. Wolfe, Methods of nonlinear programming, Recent Adv. Math. Progr. (1963) 67–86.
[36] G. Gopalakrishnan Nair, On the convergence of the LJ search method, J. Optim. Theory Appl. 28 (3) (1979) 429–434.
[37] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212, 2013.
[38] J. Li, S. Zheng, Y. Tan, Adaptive fireworks algorithm, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 3214–3221.
[39] S. Zheng, J. Li, A. Janecek, Y. Tan, A cooperative framework for fireworks algorithm, IEEE/ACM Trans. Comput. Biol. Bioinform. 14 (1) (2017) 27–41, http://dx.doi.org/10.1109/TCBB.2015.2497227, ISSN 1545-5963.
[40] M. Clerc, Standard Particle Swarm Optimization, from 2006 to 2011, Particle Swarm Central, 2011.
[41] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 39 (3) (2007) 459–471.
[42] J. Kennedy, Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, SIS'03, IEEE, 2003, pp. 80–87.
[43] M.G. Omran, A.P. Engelbrecht, A. Salman, Bare bones differential evolution, Eur. J. Oper. Res. 196 (1) (2009) 128–139.
[44] M. Zambrano-Bigiarini, M. Clerc, R. Rojas, Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 2337–2344.
[45] M. El-Abd, Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 2215–2220.
[46] N. Padhye, P. Mittal, K. Deb, Differential evolution: performances and analyses, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 1960–1967.
[47] P.N. Suganthan, Results of 22 Papers, 2017 (accessed 27.02.17) http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Sharedcuments/CEC2013/Results-of-22-papers.zip.
[48] P.N. Suganthan, Special Session & Competition on Real-Parameter Single Objective Optimization at CEC-2013, 2017 (accessed 30.03.17) http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm.
[49] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Jadavpur University, Nanyang Technological University, Kolkata, 2010.
[50] S.M. Elsayed, R.A. Sarker, D.L. Essam, GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems, in: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2011, pp. 1034–1040.
[51] Z. Michalewicz, Quo vadis, evolutionary computation? in: IEEE World Congress on Computational Intelligence, Springer, 2012, pp. 98–121.
[52] A. Saha, T. Ray, How does the good old genetic algorithm fare at real world optimization? in: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2011, pp. 1049–1056.