This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

Loser-out Tournament Based Fireworks Algorithm for Multi-modal Function Optimization

Junzhi Li and Ying Tan, *Senior Member, IEEE*

***Abstract*—Real-world optimization problems are usually multi-modal which require optimization algorithms to keep a balance between exploration and exploitation. Therefore, multi-modal optimization is one of the main opportunities as well as one of the main challenges for evolutionary algorithms. In this paper, a loser-out tournament based fireworks algorithm is proposed for solving multi-modal optimization problems. The search manner of the conventional fireworks algorithm is based on the cooperation of several fireworks. While in the loser-out tournament based fireworks algorithm, we propose competition as a new manner of interaction, in which the fireworks are compared with each other not only according to their current status but also according to their progress rate. If the fitness of a certain firework cannot catch up with the best one with its current progress rate, it is considered a loser in the competition. The losers will be eliminated and reinitialized because it is vain to continue their search processes. Reinitializing these fireworks would greatly reduce the probability of being trapped in local minima for the algorithm. Experimental results show that the proposed algorithm is very powerful in optimizing multi-modal functions. It not only outperforms previous versions of the fireworks algorithm, but also outperforms several famous evolutionary algorithms.**

***Index Terms*—Fireworks Algorithm, Multi-modal Optimization, Swarm Intelligence, Evolutionary Algorithm, Loser-out Tournament**

I. Introduction

CONVEX optimization problems can be solved easily by mathematical methods, while multi-modal global optimization is more challenging because it requires algorithms to keep a balance between exploration and exploitation, which is quite complicated. Inspired from natural phenomena, many different swarm and evolutionary algorithms have been proposed to solve multi-modal optimization problems.

The fireworks algorithm (FWA) [1] is a newly proposed optimization algorithm inspired by the phenomenon of fireworks explosion. The FWA has proven serviceable in many real-world applications and shown strong capability for global optimization. The FWA conducts heuristic search mainly by iterating the explosion operation and the selection operation. In the explosion operation, numerous sparks are generated around the fireworks within certain explosion amplitudes. After that, the fireworks of a new generation are selected from these sparks.

The core idea of the FWA is that the fireworks interchange information to dynamically control the resource allocation and the search manner. In previous works [2]–[4], cooperation is considered as the main manner of information interaction, like other swarm intelligence algorithms. While in this paper, we propose a competitive framework as an alternative manner of information interaction.

Competition is a common phenomenon in the nature and the human society. It represents a state of conflict between individuals or populations. In the nature, biological individuals or populations fight with each other for territory or resource. The losers are often exiled or even eliminated. Although competition is often a disaster for the losers, it does have positive effects on the nature. This mechanism guarantees resource will not be wasted on unfit populations or species. Land and food resources are spared for other creatures to develop. Such a paradigm has also proven efficient in multi-modal optimization problems because resource allocation is crucial to exploration.

In the proposed loser-out tournament based fireworks algorithm, fireworks compete with each other and the losers will be forced to restart from a new location. The competitive mechanism is based on the anticipation of fireworks' fitness. If the fitness of a firework cannot catch up with the best one with its current progress rate, then this firework is considered a loser and will be reinitialized to avoid wasting resources on searching unpromising areas. As an algorithmic improvement, the proposed algorithm has by far the best performance on several multi-modal objective functions among all the variants of the fireworks algorithm. It can also outperform several other typical swarm and evolutionary algorithms.

The remainder of this paper is organized as follows. Section II introduces some related works in the field of multi-modal optimization and fireworks algorithms. Section III introduces the framework of the fireworks algorithm and its operators. Our proposed loser-out tournament mechanism is presented and discussed in Section IV. Experimental results are shown in Section V to illustrate the performance of the proposed algorithm. Section VI concludes this paper.

II. Related Works

A. Multi-modal Optimization

Continuous optimization is very important in many real-world applications. Convex optimization problems [5] with known gradient can be solved by direct mathematical methods such as the gradient descent. However, many of the

Junzhi Li and Ying Tan are with the Key Laboratory of Machine Perception (Ministry of Education), Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China. Prof. Y. Tan is the corresponding author. This work was supported by the Natural Science Foundation of China (NSFC) under grant no. 61673025 and 61375119 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. Email:{ljz,ytan}@pku.edu.cn

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

landscapes of the objective functions are more complex and sometimes the gradient cannot be easily derived. Therefore, many nature-inspired evolutionary and swarm algorithms have been proposed for multi-modal global optimization, such as evolution strategy (ES) [6], particle swarm optimization (PSO) [7], simulated annealing (SA) [8], the genetic algorithm (GA) [9], differential evolution (DE) [10], ant colony optimization (ACO) [11], the culture algorithm (CA) [12], the memetic algorithm (MA) [13], etc. Although global optima are not guaranteed to be found, these algorithms are usually capable of finding high-quality solutions at reasonable costs.

Most evolutionary and swarm algorithms utilize a population which iteratively evolves (by mutation, selection or movement) to achieve better and better fitness. However, a single population often suffers from the problem of premature convergence, which is disadvantageous in multi-modal optimization [14], [15]. There are three main manners to solve this problem.

a) The most popular way is to adapt the parameters in these algorithms to control the progress [16]. These techniques include deterministic control, adaptive control and self-adaptive control.
b) Restarting the population is a very intuitive and effective way to reduce the probability of being trapped in local optima if the exploitation capability of the algorithm is very powerful [17], [18]. However, restarting a single population is sometimes not the most efficient way, because it is possible that the population searches the same local area for multiple times [19].
c) Inspired by ecological phenomena, niching techniques are proposed to find multiple local optima of the objective function [19]. The main issue of niching methods is to divide a single population into multiple sub-populations and maintain the divergence of these sub-populations.

Besides, a few researchers proposed multi-population algorithms to solve multi-modal optimization problems [20]–[23]. Parallel multi-population frameworks are equivalent to restarting a single population if there is no interaction among these populations [24]. Therefore, designing interactive mechanisms is necessary in such algorithms.

B. Fireworks Algorithm

The fireworks algorithm has attracted much research interest since it was proposed in 2010 [1].

Practitioners have found it useful in many real-world applications including clustering [25], regional seismic waveform inversion [26], constrained portfolio optimization [27], web information retrieval [28], privacy preserving [29], etc.

A theoretical analysis has proven that the FWA has some global convergence properties [30].

Several algorithmic research works have pointed out some drawbacks of the FWA and proposed more advanced versions, such as the enhanced FWA [31], the improved FWA [32], the adaptive FWA and the dynamic search FWA [33], [34]. Thanks to these works, the operators in the FWA have been significantly improved.

Still, there is much room for this young algorithm [35]. For example, the interaction among the fireworks might be improving including interactive mechanisms [2].

III. The Framework of the Fireworks Algorithm

In this section, we introduce the framework of the fireworks algorithm. This framework basically follows the independent framework proposed in [2], but there are also several important adaptations to make it clearer and more efficient.

The FWA searches for better solutions by iterating the explosion and selection operation. In the explosion operation, sparks are generated around the locations of the fireworks. In the selection operation, fireworks of the new generation are selected from the generated sparks. In the previous versions of the FWA [1], [31], [34], there is a common candidate pool (including all the generated sparks) from which all the fireworks in the next generation are selected together. A recent work [2] has defined an independent framework for the FWA. In the independent framework, each firework in the next generation is selected from its own candidate pool, which includes only its own sparks. Therefore, each firework and its explosion sparks form an independent population. Each population updates itself by generating new sparks and updating the location of the firework. These populations are independent except when the interactive mechanism is triggered. The only other relationship among these populations is that they have to share the total resource, i.e., in each generation, the total number of explosion sparks to be generated is fixed, which is allocated to the fireworks according to their fitness. Except for the interactive mechanism (which will be introduced in the next section), the roles of the fireworks are identical. The ways the populations develop are also identical. So this framework is quite suitable for parallelization.

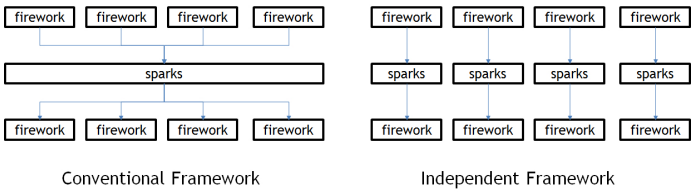

Fig. 1: Conventional Framework vs. Independent Framework

Throughout this paper, without loss of generality, we consider the following continuous minimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a vector in the d dimensional Euclidean space. The object is to find the optimal $\mathbf{x}$ with the minimal evaluation (fitness) value $f(\mathbf{x})$.

A. Explosion Operation

The explosion operation is the key operation in the FWA. In each generation, certain numbers of explosion sparks are generated around the fireworks within certain explosion amplitudes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

1) Number of Explosion Sparks: In most of the previous FWA versions [1], [31], [33], the number of explosion sparks for each firework i is calculated by the following formula:

$$\lambda_i = \hat{\lambda} \cdot \frac{\max_j\{f(\mathbf{X}_j)\} - f(\mathbf{X}_i)}{\sum_k (\max_j\{f(\mathbf{X}_j)\} - f(\mathbf{X}_k))}, \tag{2}$$

where $\mathbf{X}_i$ is the position of the ith firework, $\hat{\lambda}$ is a constant parameter which controls the total number of explosion sparks in one generation. The idea behind this formula was to make fireworks with better fitness have more explosion sparks to search the local areas more thoroughly.

However, there are some problems of this formula:

a) A previous study [32] has shown by experiments that the numbers of explosion sparks are not stable in different iterations. The fitness values fluctuate fiercely with different objective functions and different positions in the search space. As a result, there is no regularity in the number of explosion sparks according to Eq. (2).
b) Relevantly, the number of explosion sparks generated by the best firework cannot be controlled. The best firework is the most important firework in the search process because it generates the most explosion sparks and conducts exploitation to make sure the solution found by the algorithm is acceptable. However, such a design is not stable.

Proposition 1. *If μ is the number of fireworks, λ_i are calculated according to Eq.* (2)*, then*

$$\frac{\hat{\lambda}}{\mu - 1} \leq \max_i\{\lambda_i\} \leq \hat{\lambda}. \tag{3}$$

The above lower and upper bounds are both tight. Suppose the worst fitness among these fireworks $\max_j\{f(\mathbf{X}_j)\}$ is large, then the resource of the best firework is in the worst case only $\frac{1}{\mu-1}$ of the total resource. While if $\min_j\{f(\mathbf{X}_j)\}$ is small, then the best firework may occupy nearly all the resource.

c) The number of sparks of the firework with the worst fitness is zero. This was fixed in the original paper by setting thresholds [1], but it is not elegant to do so. Such a phenomenon reveals the irrationality of this formula.

Thus, in our opinion, the number of explosion sparks for each firework should depend on the ranking of its fitness value rather than the fitness value itself.

In this paper, we adopt the power law distribution [36], which is simple and very common in the nature and the human society, to determine the number of sparks for each firework. The famous 80-20 rule [37] implies that the power law distribution is a natural and efficient way to allocate the resource. That is,

$$\lambda_r = \hat{\lambda} \cdot \frac{r^{-\alpha}}{\sum_{r=1}^{\mu} r^{-\alpha}}, \tag{4}$$

where r is the fitness ranking of this firework, μ is the total number of the fireworks, and α is a parameter to control the shape of the distribution. The larger α is, the more explosion sparks good fireworks generate.

Fig. 2 shows the distributions of the numbers of explosion sparks with different values of α.

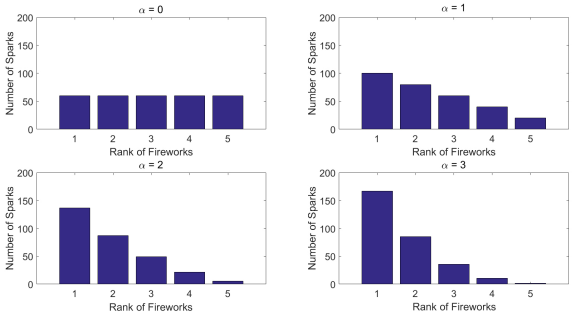

Fig. 2: The Influence of Different Values of α on the Distribution of Explosion Sparks

The parameter α has a significant influence on the performance. Generally speaking, for uni-modal functions α should be larger (elitism) while for multi-modal functions α should be smaller (equalitarianism). See Section V-A for more information.

2) Explosion Amplitude: In previous works [2], [34], only the best firework's explosion amplitude is dynamically controlled. Here, all the explosion amplitudes of the fireworks are controlled in a dynamic manner, reads

$$A_i^g = \begin{cases} A_i^1 & g = 1 \\ C_r A_i^{g-1} & f(\mathbf{X}_i^g) \geq f(\mathbf{X}_i^{g-1}) \\ C_a A_i^{g-1} & f(\mathbf{X}_i^g) < f(\mathbf{X}_i^{g-1}) \end{cases}, \tag{5}$$

where A_i^g is the explosion amplitude of the ith firework in generation g. In the first generation, the amplitude is preset to a constant number which is usually the diameter of the search space. After that, if in generation $g-1$ a firework found a better solution than the best in generation $g-2$, the amplitude 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 new firework, 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. Otherwise it may be too short (conservative) and cannot make the largest progress and thus need to be amplified. With the dynamic control, the algorithm is able to keep the amplitudes proper for the search. That is, the dynamic explosion amplitude is long in the earlier phases to perform exploration, and is short in the later phases to perform exploitation.

3) Generating Explosion Sparks: Algorithm 1 shows how the explosion sparks are generated for each firework (where the superscript g representing the generation number is omitted), which is simplified compared to previous versions [1], [31]. [1]

The explosion sparks are generated uniformly within a hypercube. The radius of the hypercube is the explosion

[1]There was a dimension selection mechanism in the explosion operation, but it is eliminated here because it is not effective and it costs some extra time to generate random numbers [38], [39].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

amplitude and the center of the hypercube is the position of the firework.

Algorithm 1 Generating the Explosion Sparks for the ith. Firework

Require: $\mathbf{X}_i$, A_i and λ_i
1: **for** $j = 1$ **to** λ_i **do**
2: **for** each dimension $k = 1, 2, ..., d$ **do**
3: sample η from $\mathcal{U}(-1, 1)$
4: $\mathbf{s}_{ij}^{(k)} \leftarrow \mathbf{X}_i{}^{(k)} + \eta \cdot A_i$
5: **end for**
6: **end for**
7: **return** all the $\mathbf{s}_{ij}$

B. Mutation Operation

The mutation operation is optional in the FWA, in which the positions of the fireworks are mutated to generate mutation sparks. Several different types of mutation operators [4], [31], [40], [41] have been proposed to enhance the diversity of the population. Here, we adopt a recently proposed guiding spark [38] which is simple but efficient. It helps both to exploration and to exploitation.

Algorithm 2 shows how the guiding sparks are generated for each firework.

Algorithm 2 Generating the Guiding Spark for the ith firework

Require: $\mathbf{X}_i$, $\mathbf{s}_{ij}$, λ_i and σ
1: Sort the sparks by their fitness values $f(\mathbf{s}_{ij})$ in the ascending order.
2: $\mathbf{\Delta}_i \leftarrow \frac{1}{\sigma\lambda_i}(\sum_{j=1}^{\sigma\lambda_i} \mathbf{s}_{ij} - \sum_{j=\lambda_i-\sigma\lambda_i+1}^{\lambda_i} \mathbf{s}_{ij})$
3: $\mathbf{M}_i \leftarrow \mathbf{X}_i + \mathbf{\Delta}_i$
4: **return** $\mathbf{M}_i$

Generating guiding sparks requires the information acquired by explosion sparks. σ is a parameter to control the proportion of adopted explosion sparks. Note that only one guiding spark is generated for each firework.

C. Mapping Rule

For constrained optimization problems, the sparks which are out of boundaries need to be mapped into the feasible space. In this paper, we follow the uniform random mapping operator introduced in the EFWA [31], that is, if a spark is located out of the boundaries, it will be replaced by a new one which is uniformly randomly chosen from the feasible space. We point out that if the optimal point is located near the boundaries, such kind of mapping operator may reduce the convergence speed. However, it is good for exploration.

D. Selection Mechanism

In the independent framework, each firework in the next generation is selected from its own candidate pool, which includes only its own offspring. Hence, each firework along with its sparks forms a separate population. Different populations can be located remotely. Each population in the algorithm searches independently in the same manner, except when the interactive mechanism is triggered. There are many different ways to select the fireworks of the next generation [1], [4], [31]. In this paper, we follow the elitism selection mechanism [2] and greedily select the best individual in each population to be the firework of the next generation (survival of the fittest). That is,

$$\mathbf{X}_i^{g+1} = \arg\min\{f(\mathbf{X}_i^g), f(\mathbf{s}_{ij}), f(\mathbf{M}_i)\}. \quad (6)$$

Although initially designed as a swarm intelligence algorithm, the FWA also follows the same framework of evolutionary algorithms. Compared with typical swarm intelligence algorithms like PSO or ACO, the selection operation in the FWA makes the population inherit and utilize information in a totally different manner. The main difference between the FWA and estimation of distribution algorithms [42] is the *implicit* sampling distribution in the FWA, which is more flexible but less theoretically clear. Compared with typical evolutionary algorithms like ES or GA, the framework of the FWA allows interactive behaviors among fireworks as we will see in the rest of this paper.

IV. Loser-out Tournament Based Fireworks Algorithm

A. Motivation

The key to multi-modal optimization is to balance between exploitation and exploration [43]. However, the word "balance" may be misleading in the following sense: exploitation and exploration are not just opposites. Sometimes they can be achieved simultaneously and sometimes they even help each other. Promoting the exploitation does not necessarily harm the exploration, and vice versa.

Utilizing multiple populations instead of a single population can be advantageous for exploration if efficient interactive mechanisms are adopted. The mechanisms of interaction can be roughly divided into cooperation and competition [44]. As far as artificial algorithms are concerned, the difference between cooperation and competition is not substantial but only conceptional.

Cooperation and competition at the individual level are both commonly used in nature-inspired heuristics. Most swarm algorithms prefer cooperation [7], [11], in which centralized control is not necessary. While evolutionary algorithms use competition more frequently [6], [45], because selection is the most natural way of competition. Some algorithms propose combining these two manners [10], [46], [47].

Cooperation and competition at the population level are more complex and more rarely used in nature-inspired heuristics. In the nature, the relationship among different species is usually called coevolution [48]. Among different populations of the same species, competition is typically more common than cooperation because otherwise these populations may be united as a larger one. The phenomenon of cooperation among multiple populations has been applied in PSO [49], GA [50], CA [51], etc. The phenomenon of competition among multiple

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

populations has been applied in ABC [52], GA [50], MA [53], etc.

As the most fierce way of competition, war is known for its cruelty. The losers in a war are usually exiled or eliminated. However, the positive influence of the wars is that 1) the defeated populations may restart in a new environment and 2) resources (land, food, etc.) can be spared for new populations to develop and evolve. These influences are important in the natural evolution process. Therefore, we believe such a mechanism can also help the exploration in multi-modal optimization.

Different from other swarm intelligence algorithms, there are naturally multiple populations in the fireworks algorithm, because each firework and its own sparks are geographically close and therefore share similar properties. Based on such a framework, a competitive interaction mechanism can be implemented. The following questions arise: 1) how to design the competitive mechanism among the populations; 2) how to improve exploration based on the competition results.

1) Competition: One of the most simple ways of competition in the nature is called single combat [54], which is a duel between two warriors from the two armies. These two warriors are usually champions of the two sides. A single combat can sometimes reduce the casualty of the war and save time. Especially for designing an optimization algorithm, it is computationally expensive and not necessary to compare the qualities of all the individuals in these populations because at the end of the optimization process, we usually only care about the best solution found.

In the fireworks algorithm, the fireworks represent the best individuals of the populations (see Eq.(6)). Therefore, the competitive mechanism in our proposed algorithm is based on the fitness comparison among the fireworks. However, a currently inferior firework is not necessarily inferior in the future. The information we need to obtain from the comparison is not the fitness of the current position of a firework, but the quality of the local area. Hence, we make an anticipation for each firework by predicting the fitness of this firework at the end of the optimization process. These anticipations reflect whether these fireworks are promising. If the anticipation of a certain firework is worse than the current fitness of the best firework, this firework is considered as a loser.

2) Loser-out: In the nature, the losers in a competition are usually exiled or eliminated. In order to keep the exploration capability of the algorithm, eliminating all the losers is not an option because the number of populations would quickly reduce to one. In the loser-out mechanism, the losers are forced to remove from their current positions. This is actually a hopeful mechanism for the losers because they may develop and evolve in a different place without the direct danger from the winners. It is quite frequent both in the nature and in the human society that a certain population retakes its territory or achieves great success after it is exiled. Loser-out can be considered as a mechanism for "avoiding local optima" and "maintaining diversity" in the nature.

As for multi-modal optimization, the loser-out mechanism is conducted by reinitialization. That is, randomly re-choose a position for the firework to begin its search process and reset all its parameters. In this way, the probability of finding the global optimum is improved. For example, if this probability of a single trial is 0.1, then the probability of ten independent trials is $1-(1-0.1)^{10} \approx 0.65$.

Of course, how to set a competitive mechanism and what to do with the results should be considered together. For example, if the competition happens too frequently, the loser-out mechanism may not help the exploration but harm the exploitation severely, because the local search of a loser firework may be wrongly interrupted.

B. Loser-out Tournament (LoT)

To predict the final fitness of a firework can be considered as a problem of time series forecasting [55]. There are numerous approaches for this task. But to make predictions for each firework in each generation can be time consuming using these methods. In this regard, we cannot make strong assumptions on the series and we do not need extreme accuracy in the prediction. In addition, in this certain task, the risk of underestimation is much higher than overestimation. In other words, we would rather give a firework more time to search around its current local area than arbitrarily exile it because some resource has already been used in searching this area. Taking these factors into consideration, we think a linear prediction is appropriate for this task.

Define the improvement of the ith firework $\mathbf{X}_i^g$ in generation g as

$$\delta_i^g = f(\mathbf{X}_i^{g-1}) - f(\mathbf{X}_i^g) \geq 0. \tag{7}$$

δ_i^g indicates the extent of improvement of this population. It is the difference between the best individual generated in this generation and the best individual generated in the last generation. If it is large, it means the population is improving quickly and this local area is still of much potential. It becomes smaller when the population gradually approaches the local optimum.

Then we make the prediction of its fitness in the final generation g_{max} as

$$\widehat{f(\mathbf{X}_i^{g_{max}})} = f(\mathbf{X}_i^g) - (g_{max} - g)\delta_i^g. \tag{8}$$

The ith firework is considered as a loser and will be reinitialized if the prediction is worse than the current best one, i.e., $\widehat{f(\mathbf{X}_i^{g_{max}})} > \min_j\{f(\mathbf{X}_j^g)\}$. By reinitialization, we mean its position will be randomly chosen in the search space and its explosion amplitude will be set to the initial value.

Algorithm 3 shows how the loser-out tournament mechanism works in every generation.

1) Caution: Note that if $f(\mathbf{X}_i^g) = f(\mathbf{X}_i^{g-1})$, i.e., no improvement is made, this mechanism will not be triggered. If we delete line 2 and line 4 in Algorithm 3, then the fireworks will be reinitialized too frequently due to a property of the dynamic explosion amplitude and the elitism selection: $f(\mathbf{X}_i^g) = f(\mathbf{X}_i^{g-1})$ is a regular event in the search process. With the dynamic explosion amplitude, the frequency of improvement is nearly a constant, which is almost irrelevant of the property of the objective function. To see this, consider a time period of t generations: $g+1$ to $g+t$. Suppose the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

Algorithm 3 Loser-out Tournament

Require: maximal generation number g_{max}, fireworks number μ

1: **for** $i = 1$ **to** μ **do**
2: **if** $f(\mathbf{X}_i^g) < f(\mathbf{X}_i^{g-1})$ **then**
3: $\delta_i^g \leftarrow f(\mathbf{X}_i^{g-1}) - f(\mathbf{X}_i^g)$
4: **end if**
5: **if** $\delta_i^g \cdot (g_{max} - g) < f(\mathbf{X}_i^g) - \min_j\{f(\mathbf{X}_j^g)\}$ **then**
6: reinitialize the ith firework.
7: **end if**
8: **end for**

average frequency of improvements in this period is p, then according to Eq.(5),

$$A^{g+t} = C_a^{tp} C_r^{t(1-p)} A^{g+1}. \tag{9}$$

If t is large,

$$C_a^p C_r^{(1-p)} = \sqrt[t]{\frac{A^{g+t}}{A^{g+1}}} \approx 1. \tag{10}$$

Thus,

$$p \approx \frac{\log C_r}{\log C_r - \log C_a}. \tag{11}$$

It means, the dynamic explosion amplitude usually adapts itself to keep the frequency of improvements stable. The fixed point of p depends only on the two parameters C_r and C_a. Hence, a single failure does not mean this local area is unpromising because it happens regularly.

When the loser-out tournament is adopted by other heuristics, line 2 and line 4 in Algorithm 3 are also necessary if the heuristic does not guarantee improvement in every iteration, just like the FWA.

2) Safety and Efficiency: In the following, we explain why our condition of reinitialization is safe. If a firework could never surpass the best one, then it is right and effective to reinitialize it. On the contrary, if this firework could have surpassed the best one if it is not reinitialized, i.e., $f(\widetilde{\mathbf{X}_i^{g_{max}}}) < \min_j\{f(\mathbf{X}_j^{g_{max}})\}$, where $f(\widetilde{\mathbf{X}_i^{g_{max}}})$ is what the fitness of this firework would be if it is not reinitialized, then reinitializing it is not only ineffective but also harmful to the search process.

But when it satisfies the condition in Algorithm 3, the conditional probability

$$\Pr(f(\widetilde{\mathbf{X}_i^{g_{max}}}) < \min_j\{f(\mathbf{X}_j^{g_{max}})\} | f(\widehat{\mathbf{X}_i^{g_{max}}}) > \min_j\{f(\mathbf{X}_j^g)\})$$

becomes very low because:

a) The fitness of this firework does not improve in every generation. As we explained, p is nearly a constant. During the following $g_{max} - g$ generations, improvements happen only in about $(g_{max} - g)p$ generations. In other generations, the fitness does not improve. While, we assume it improves in every generation in Eq. (8).

b) The fitness of the best one will never suffer. On the contrary, it may also improve.

Proposition 2. $\min_j\{f(\mathbf{X}_j^{g_{max}})\} \le \min_j\{f(\mathbf{X}_j^g)\}$.

Proof: Define $k_g = \arg\min_j\{f(\mathbf{X}_j^g)\}$. If $k_{g+1} = k_g$, by the elitism selection (Eq. (6)), the fitness of a certain firework is monotonically nonincreasing, $f(\mathbf{X}_{k_g}^{g+1}) \le f(\mathbf{X}_{k_g}^g)$. If $k_{g+1} \ne k_g$, by the definition of k_{g+1}, $f(\mathbf{X}_{k_{g+1}}^{g+1}) \le f(\mathbf{X}_{k_{g+1}}^{g+1}) \le f(\mathbf{X}_{k_g}^g)$. By mathematical induction, $\min_j\{f(\mathbf{X}_j^{g_{max}})\} = f(\mathbf{X}_{k_{g_{max}}}^{g_{max}}) \le f(\mathbf{X}_{k_g}^g) = \min_j\{f(\mathbf{X}_j^g)\}$. ■

c) The improvement generally reduces with the search process. If we admit the positive correlation between the step size and the extent of improvement, then the improvement generally reduces because the explosion amplitude reduces. For the ith firework (thus i is omitted), let $C = [X_1^g - A^g, X_1^g + A^g] \times ... \times [X_d^g - A^g, X_d^g + A^g]$ and $B = \{x \in \mathbb{R}^d | f(x) < f(\mathbf{X}^g)\}$ where the subscript indicates the dimensionality, then

$$p \approx \Pr\{f(\mathbf{X}^{g+1}) < f(\mathbf{X}^g)\} = \frac{|C \cap B|}{|C|}. \tag{12}$$

$|B|$ reduces with the search process because the fitness of the firework is monotonically nonincreasing. Therefore $|C \cap B| \le |B|$ also reduces. As we explained, p can be considered as a constant, so $|C| = (2A^g)^d$ reduces with the search process. Therefore A reduces with the search process (though slowly). That is, in order to maintain the frequency of improvements, the extent of improvements reduces. (See also the discussion in the Appendix A.3 of [56]).

Although unlikely, it is still possible that a firework which is actually searching a promising area is wrongly reinitialized due to its very bad luck, but this is considered as an affordable risk because the robustness of a swarm algorithm is based on the contribution of all individuals.

On the other hand, our proposed mechanism is efficient, because it does not need the actual final fitness of the fireworks. Once a firework is considered as unpromising, it will be reinitialized instantly, which gives these fireworks many times of retrying. For example, if a firework is unfortunately located in the same area with the best one, we do not need to let it search this area further, because it is almost certain that this firework may never catch up with the best one.

At the earlier phases, the improvements of the fireworks are relatively large, so that they will not be frequently reinitialized and can exploit deeper in a local area. While at the later phases, the improvements of the fireworks become relatively small, and the remaining number of generations becomes lower, so they will begin to be reinitialized to search new areas if the current one is considered unpromising or hopeless by this mechanism. In this way, the algorithm not only avoids searching the same area with multiple fireworks but also avoids searching unpromising areas.

3) Comparison with Single Population: The loser-out tournament is designed for the framework of multiple populations. In the following, we analyze its advantage over iterating a single population. Consider an ideal model. Suppose there are m local optima and among them only one is the global optimum. By iterating only a single population, the probability of finding

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

the global optimum with exact k trials is $(\frac{m-1}{m})^{k-1}\frac{1}{m}$. Then the expected number of trials is

$$\sum_{k=1}^{\infty} k(\frac{m-1}{m})^{k-1}\frac{1}{m} = m. \quad (13)$$

While the proposed mechanism can avoid visiting the same local optima for multiple times. Therefore, the probability of finding the global optimum with exact k trials is

$$\frac{m-1}{m}\cdot\frac{m-2}{m-1}\cdot\ldots\cdot\frac{m-k+1}{m-k+2}\cdot\frac{1}{m-k+1} = \frac{1}{m}. \quad (14)$$

Then the expected number of trials is

$$\sum_{k=1}^{m}\frac{k}{m} = \frac{m+1}{2}. \quad (15)$$

Actual optimization problems are more complex than this model. It is sometimes impossible to find even one local optimum within the time limit because a so-called local search is sometimes not simple at all. In this case, the advantage of our proposed mechanism may become even more obvious for multi-modal optimization. If there is only one population, it will not be reinitialized before it approaches a local optimum. While, in the loser-out tournament, the other populations do not need to wait for the best population to exhaust all the resource in local exploitation. Instead, they can explore other local areas and meanwhile the best population is still searching around its position until a better local area is found.

More empirical analyses of this mechanism are given in Section V-B.

C. Loser-out Tournament Based Fireworks Algorithm

Algorithm 4 shows the complete loser-out tournament based fireworks algorithm (LoTFWA) proposed in this paper. In the initialization phase, the location of the fireworks are randomly set in the whole feasible space and the explosion amplitudes are set to the diameter of the search space. There are two parts in the iteration phase. In the first part, sparks are generated by each firework, and the position of each firework is updated according to the selection operator. In the second part, fireworks compete with the best one and the loser fireworks are reinitialized. The iteration continues until the termination criterion (running time, precision, number of evaluations, etc.) is met.

Algorithm 4 Loser-out Tournament Based Fireworks Algorithm

```
Randomly initialize μ fireworks in the search space.
Evaluate the fireworks' fitness.
repeat
    for i = 1 to μ do
        Calculate λ_i according to Eq.(4).
        Calculate A_i according to Eq.(5).
        Generate explosion sparks according to Alg.1.
        Generate guiding sparks according to Alg.2.
        Evaluate all the fitness of the sparks.
        Select the best individual (including the ith. firework,
        its explosion sparks and guiding sparks) as the ith.
        firework of next generation.
    end for
    Perform the loser-out tournament according to Alg.3.
until termination criterion is met.
return the position and the fitness of the best individual.
```

V. Experiments

A. Parameter Setting

In this subsection, we give principles for setting the parameters in the LoTFWA. The main parameters include: the number of fireworks μ, the total number of explosion sparks λ, the dynamic amplitude coefficients C_a and C_r, the mutation parameter σ and the power law distribution parameter α.

Basically, smaller μ makes the algorithm good at exploiting because each firework can generate more sparks. While larger μ makes the algorithm able to explore more areas but in that case each firework generates fewer sparks. In this paper, we follow the suggestion in [1] and set $\mu = 5$.

C_a and C_r are two important parameters for the dynamic amplitude control. The larger C_a and C_r are, the stronger the capability of exploration is. In this paper, we follow the suggestions in [31] and use $C_a = 1.2$ and $C_r = 0.9$.

The influence of the mutation parameter σ is complicated. Larger σ makes the direction of the mutation vector accurate but also makes the step size conservative. According to the experimental results in [38], $\sigma = 0.2$ is usually the best choice.

The parameter α is introduced in this paper, we will show the influence of this parameter by experiments. Besides, the influence of the parameter λ is also important in the proposed algorithm. We conduct experiments to compare the 16 sets of parameters, i.e., $\{\alpha, \lambda\} \in \{0, 1, 2, 3\} \times \{100, 200, 300, 400\}$. The parameters are evaluated on the CEC 2013 single objective optimization benchmark suite [57] including 5 uni-modal functions and 23 multi-modal functions (shown in Table I). The dimensionality of these functions is $d = 30$. According to the instructions of this benchmark suite, all the algorithms are run 51 times for each function and the maximal number of function evaluations in each run is $10000d$. The mean errors of the 16 sets of parameters are ranked, and the ranks are averaged over the 28 test functions, as shown in Fig. 3 (the lower, the better).

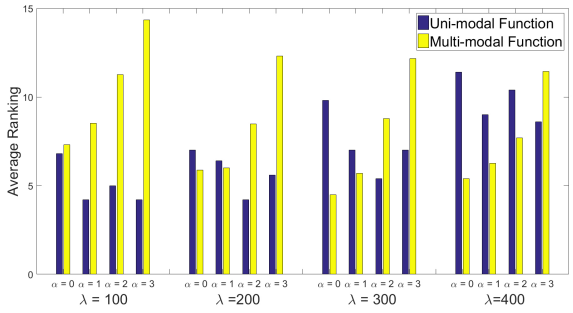

Fig. 3: Average Ranks of the 16 Sets of Parameters

According to the experimental results, we have the follow-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

TABLE I: 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)

ing observations.

a) For uni-modal functions, λ should be small. It implies that for uni-modal functions, the number of generations is more important than the population size. Smaller population and more generations are good for uni-modal functions.
b) For uni-modal functions, α should be large (the performances of 1, 2 and 3 are comparable). It implies that for uni-modal functions, the resource should be concentrated on elite fireworks. The purpose of other fireworks is exploration, which is not important for uni-modal functions.
c) For multi-modal functions, α should be small. For multi-modal functions, equalitarianism is the best policy because it divides the resource equally and enables all the populations to search effectively.
d) For multi-modal functions, $\alpha = 0, \lambda = 300$ performs the best. For multi-modal functions, the population size and the number of generations are both important. They should be balanced according to the maximal number of function evaluations and the complexity of the problem. For this benchmark, the maximal number of function evaluations is comparatively sufficient (10000 times the dimensionality), so the population size can be large. For real-world applications, smaller λ is suggested.

B. The Significance of the LoT

The LoT is a reinitialization strategy based on the competitive interaction. In order to illustrate the significance of the LoT, two questions have to be answered:

a) Does the LoTFWA outperform the FWA without reinitialization strategy?
b) Does the LoTFWA outperform the FWA with a simple reinitialization strategy?

In the following, the FWA with LoT is denoted as LoTFWA, the FWA without reinitialization strategy is denoted as NRS, and the FWA with a simple reinitialization strategy is denoted as SRS. In the SRS, the same restart criterion is set for all the fireworks, where if the improvement of a firework is less than 1e-10 for more than 5 times in a row, it will be reinitialized. The parameters of the three algorithms are all set to $\mu = 5, \lambda = 300, C_a = 1.2, C_r = 0.9, \sigma = 0.2$ and $\alpha = 0$.

The mean errors on the 28 functions are shown in Table II. The mean errors are ranked on each function and the averaged rankings (AR.) over all functions for each algorithm are calculated, shown in the last row in Table II. Note that by the central limit theorem, $AR \sim \mathcal{N}(\frac{k+1}{2}, \frac{k^2-1}{12N})$, where k is the number of algorithms, N is the number of test functions. Here $k = 3$ and $N = 28$, therefore $AR \sim \mathcal{N}(2, 0.15^2)$. Compared with the prior standard deviation 0.15, $2 - 1.5 = 0.5$ is a very large difference between the average rankings of the SRS and the LoTFWA.

The best mean errors are highlighted. In addition, two-sided Wilcoxon rank sum tests (with confidence level 95%) are conducted between the LoTFWA and the NRS, and between the LoTFWA and the SRS. In the last two columns of Table II, the '1' indicates that the LoTFWA performs significantly better, the '-1' indicates its opponent performs significantly better, while the '0' indicates that their performances are not significantly different.

TABLE II: Comparison among LoTFWA, NRS and SRS

F.	NRS	SRS	LoTFWA	vs. NRS	vs. SRS
1	**0.00E+00**	**0.00E+00**	**0.00E+00**	0	0
2	**1.08E+06**	1.13E+06	1.19E+06	0	0
3	**1.43E+07**	1.94E+07	2.23E+07	0	-1
4	1.73E+03	**1.67E+03**	2.13E+03	-1	-1
5	**3.23E-03**	3.24E-03	3.55E-03	-1	-1
6	1.60E+01	**1.33E+01**	1.45E+01	0	0
7	6.75E+01	6.51E+01	**5.05E+01**	1	1
8	2.09E+01	2.09E+01	**2.09E+01**	1	1
9	1.66E+01	1.68E+01	**1.45E+01**	1	1
10	2.69E-02	**2.59E-02**	4.52E-02	-1	-1
11	8.17E+01	7.14E+01	**6.39E+01**	1	1
12	8.43E+01	7.70E+01	**6.82E+01**	1	1
13	1.71E+02	1.46E+02	**1.36E+02**	0	1
14	2.89E+03	2.84E+03	**2.38E+03**	1	1
15	3.15E+03	3.02E+03	**2.58E+03**	1	1
16	8.82E-02	7.05E-02	**5.74E-02**	1	1
17	7.12E+01	7.39E+01	**6.20E+01**	1	1
18	7.32E+01	7.40E+01	**6.12E+01**	1	1
19	3.55E+00	3.66E+00	**3.05E+00**	1	1
20	1.31E+01	**1.24E+01**	1.33E+01	-1	0
21	2.14E+02	2.04E+02	**2.00E+02**	0	0
22	3.56E+03	3.46E+03	**3.12E+03**	1	1
23	3.79E+03	3.74E+03	**3.11E+03**	1	1
24	2.45E+02	2.48E+02	**2.37E+02**	1	1
25	2.83E+02	2.83E+02	**2.71E+02**	1	1
26	2.00E+02	**2.00E+02**	2.00E+02	0	0
27	7.90E+02	7.85E+02	**6.84E+02**	1	1
28	2.80E+02	2.84E+02	**2.65E+02**	0	0
AR.	2.39	2	**1.5**	16:4	17:4

The performance of the LoTFWA is slightly worse than both the NRS and the SRS on uni-modal functions (1-5) because in this case reinitializing the fireworks is not only meaningless but also wasteful. In the LoT mechanism, basically only one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

firework is conducting the local search, while other fireworks keep being reinitialized. Although using five fireworks to search the same local area is not efficient, they can find the local optimum slightly faster than only one firework.

On the other hand, judging from the performances on multi-modal and composition functions, the LoT mechanism greatly enhances the capability of exploration. The SRS is also effective compared with the NRS, but it is not as effective as the LoT. In the SRS, each firework only utilizes its own information. Thus, it can only detect whether it has approached the local optima, but cannot obtain the knowledge about the relative quality of this local area. On the contrary, the LoT enables a firework to acquire information from others and give up unpromising areas immediately. Therefore, the LoTFWA is able to more thoroughly explore the whole search space instead of being trapped in bad local areas.

Fig. 4 shows an example of how the LoT works. At the beginning, Firework3 found the best local area among all the fireworks, and other fireworks started to be reinitialized because they could not catch up with the evaluation value of Firework3. Fortunately, after the reinitialization, Firework4 found a better local area. Then Firework3 gave up its current location immediately and began to search other areas in the search space. From then on, other fireworks kept being reinitialized because they could not find better areas.

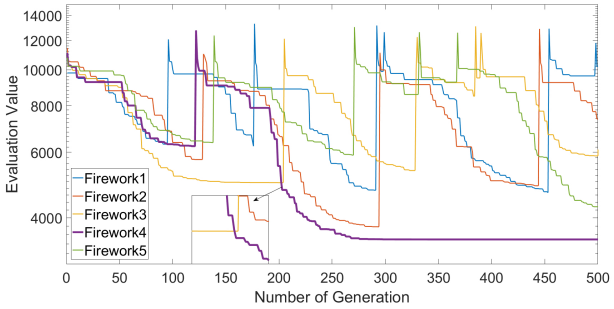

Fig. 4: Curves of Evaluation Values on Function 23

C. Comparison with Other FWA variants

In this subsection we compare the LoTFWA with other FWA variants, including the enhanced fireworks algorithm (EFWA) [31], the adaptive fireworks algorithm (AFWA) [33], the dynamic search firework algorithm (dynFWA) [34], the cooperative framework fireworks algorithm (CoFFWA) [2] and the guided fireworks algorithm (GFWA) [38]. The parameters of these algorithms are set to the suggested values in their original papers. All these algorithms are tested under the same conditions as the LoTFWA on the CEC13 benchmark. Their mean errors and standard deviations are shown in Table III. Their rankings are averaged separately over uni-modal (1-5) and multi-modal (6-28) functions, also shown in Table III. (The prior standard deviation of ARs on uni-modal functions is 0.76, while on multi-modal functions is 0.36.) The minimal mean errors on each function are highlighted. According to the ARs, GFWA is the best FWA version for uni-modal optimization, while LoTFWA is by far the best FWA version for multi-modal optimization.

Also, two-sided Wilcoxon rank sum tests (with confidence level 95%) are conducted between the LoTFWA and each of the other FWA variants. The numbers of significantly better results of the LoTFWA (indicated by "win") and of its opponents (indicated by "lose") are shown in Fig. 5. It can be seen that the advantage of the LoTFWA is overwhelming on multi-modal functions.

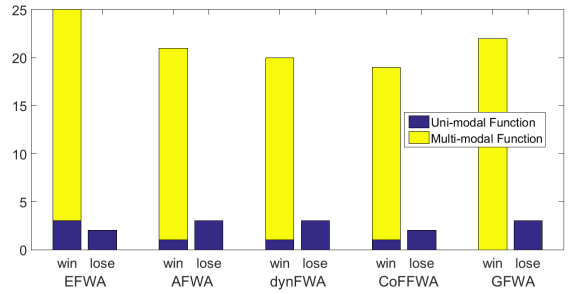

Fig. 5: Rank Sum Test Results between LoTFWA and Other FWA Variants

The LoTFWA can be considered as an extension of the GFWA, in which only one firework is adopted. In the GFWA, the resource is concentrated only on the single firework, and hence it can achieve a good exploitation capability. While, by introducing multiple fireworks and an efficient interactive mechanism among these fireworks, the LoTFWA has a much greater exploration capability.

The comparisons also indicate that the crowdedness-avoiding reinitialization strategy in the CoFFWA is not as efficient as the LoT. In fact, the crowdedness-avoiding strategy only considers a special case of the LoT, i.e., another firework is located in the same local area as the best firework. In addition, in the crowdedness-avoiding reinitialization strategy, with the explosion amplitude of the best firework reducing, the odds of triggering the criterion tend to decrease. While the LoT can be triggered even if they are located remotely. Moreover, the LoT can be parameter-free, while there is a preset parameter in the crowdedness-avoiding strategy.

It can be seen from Table II and Fig. 5 that the LoTFWA sacrifices a part of its exploitation capability for a stronger exploration capability, which brings a significant elevation of performance on multi-modal test functions.

D. Comparison with Other Typical Evolutionary Algorithms

In this subsection, the LoTFWA is compared with the artificial bee colony algorithm (ABC), the standard particle swarm optimization 2011 (SPSO2011) [58], the restart CMA-ES with increasing population size (IPOP-CMA-ES) [18], and the differential evolution (DE) algorithm [10]. All these algorithms are tested under the same conditions as the LoTFWA. The parameter setting and the results of ABC, SPSO2011, IPOP-CMA-ES and DE are reported in [59]–[62]. Some raw data can be downloaded from [63]. The mean errors and standard deviations are shown in Table IV where the minimal mean

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

TABLE III: Mean Errors, Standard Deviations and Average Ranks of FWA Variants

	EFWA		AFWA		dynFWA		COFFWA		GFWA		LoTFWA	
F.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	7.82E-02	1.31E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
2	**5.43E+05**	**2.04E+05**	8.92E+05	3.92E+05	7.87E+05	3.56E+05	8.80E+05	4.18E+05	6.96E+05	2.66E+05	1.19E+06	4.27E+05
3	1.26E+08	2.15E+08	1.26E+08	1.54E+08	1.57E+08	2.21E+08	8.04E+07	8.88E+07	3.74E+07	8.65E+07	**2.23E+07**	**1.91E+07**
4	1.09E+00	3.53E-01	1.14E+01	6.83E+00	1.28E+01	8.06E+00	2.01E+03	1.37E+03	**5.00E-05**	**6.17E-05**	2.13E+03	8.11E+02
5	7.90E-02	1.01E-02	6.04E-04	9.24E-05	**5.42E-04**	**7.98E-05**	7.41E-04	9.82E-05	1.55E-03	1.82E-04	3.55E-03	5.01E-04
AR.	4.00		3.00		3.00		3.20		**2.00**		3.80	
6	3.49E+01	2.71E+01	2.99E+01	2.63E+01	3.15E+01	2.62E+01	2.47E+01	2.08E+01	3.49E+01	2.74E+01	**1.45E+01**	**6.84E+00**
7	1.33E+02	4.34E+01	9.19E+01	2.63E+01	1.03E+02	2.95E+01	8.99E+01	1.78E+01	7.58E+01	2.98E+01	**5.05E+01**	**9.69E+00**
8	2.10E+01	4.82E-02	2.09E+01	7.85E-02	2.09E+01	7.59E-02	2.09E+01	9.79E-02	2.09E+01	9.11E-02	**2.09E+01**	**6.14E-02**
9	3.19E+01	3.48E+00	2.48E+01	4.89E+00	2.56E+01	3.95E+00	2.40E+01	4.04E+00	1.83E+01	4.61E+00	**1.45E+01**	**2.07E+00**
10	8.29E-01	8.42E-02	4.73E-02	3.44E-02	4.20E-02	2.76E-02	**4.10E-02**	**2.69E-02**	6.08E-02	3.36E-02	4.52E-02	2.47E-02
11	4.22E+02	9.26E+01	1.05E+02	3.43E+01	1.07E+02	3.23E+01	9.90E+01	2.36E+01	7.50E+01	2.59E+01	**6.39E+01**	**1.04E+01**
12	6.33E+02	1.38E+02	1.52E+02	4.43E+01	1.56E+02	5.57E+01	1.40E+02	4.06E+01	9.41E+01	3.28E+01	**6.82E+01**	**1.45E+01**
13	4.51E+02	7.45E+01	2.36E+02	6.06E+01	2.44E+02	5.35E+01	2.50E+02	5.93E+01	1.61E+02	4.74E+01	**1.36E+02**	**2.30E+01**
14	4.16E+03	6.16E+02	2.97E+03	5.70E+02	2.95E+03	5.51E+02	2.70E+03	4.95E+02	3.49E+03	8.30E+02	**2.38E+03**	**3.13E+02**
15	4.13E+03	5.61E+02	3.81E+03	5.03E+02	3.71E+03	7.57E+02	3.37E+03	5.01E+02	3.67E+03	6.35E+02	**2.58E+03**	**3.83E+02**
16	5.92E-01	2.30E-01	4.97E-01	2.56E-01	4.77E-01	3.34E-01	4.56E-01	3.15E-01	1.00E-01	7.13E-02	**5.74E-02**	**2.13E-02**
17	3.10E+02	6.52E+01	1.45E+02	2.55E+01	1.48E+02	3.74E+01	1.10E+02	2.16E+01	8.49E+01	2.10E+01	**6.20E+01**	**9.45E+00**
18	1.75E+02	3.81E+01	1.75E+02	4.92E+01	1.89E+02	6.04E+01	1.80E+02	4.04E+01	8.60E+01	2.33E+01	**6.12E+01**	**9.56E+00**
19	1.23E+01	3.68E+00	6.92E+00	2.37E+00	6.87E+00	1.93E+00	6.51E+00	2.08E+00	5.08E+00	1.88E+00	**3.05E+00**	**6.43E-01**
20	1.46E+01	1.73E-01	**1.30E+01**	**9.72E-01**	1.30E+01	1.01E+00	1.32E+01	1.01E+00	1.31E+01	1.09E+00	1.33E+01	1.02E+00
21	3.24E+02	9.67E+01	3.16E+02	9.33E+01	2.92E+02	8.39E+01	2.06E+02	6.14E+01	2.59E+02	8.58E+01	**2.00E+02**	**2.80E-03**
22	5.75E+03	1.08E+03	3.45E+03	7.44E+02	3.41E+03	5.82E+02	3.32E+03	6.31E+02	4.27E+03	8.90E+02	**3.12E+03**	**3.79E+02**
23	5.74E+03	7.59E+02	4.70E+03	8.98E+02	4.55E+03	8.63E+02	4.47E+03	7.90E+02	4.32E+03	7.69E+02	**3.11E+03**	**5.16E+02**
24	3.37E+02	7.33E+01	2.70E+02	1.31E+01	2.72E+02	1.29E+01	2.68E+02	2.19E+01	2.56E+02	1.75E+01	**2.37E+02**	**1.20E+01**
25	3.56E+02	2.80E+01	2.99E+02	1.24E+01	2.97E+02	1.07E+01	2.94E+02	1.28E+01	2.89E+02	1.34E+01	**2.71E+02**	**1.97E+01**
26	3.21E+02	9.04E+01	2.73E+02	8.51E+01	2.62E+02	8.11E+01	2.13E+02	4.16E+01	2.05E+02	2.71E+01	**2.00E+02**	**1.76E-02**
27	1.28E+03	1.10E+02	9.72E+02	1.33E+02	9.92E+02	1.22E+02	8.71E+02	2.10E+02	8.15E+02	1.22E+02	**6.84E+02**	**9.77E+01**
28	4.34E+03	2.08E+03	4.37E+02	4.67E+02	3.40E+02	2.43E+02	2.84E+02	5.41E+01	3.60E+02	2.60E+02	**2.65E+02**	**7.58E+01**
AR.	5.87		4.13		4.04		2.83		2.87		**1.26**	

errors on each function are highlighted. The rankings of the mean errors of the five algorithms are averaged over uni-modal and multi-modal functions separately. (The prior standard deviation of ARs on uni-modal functions is 0.63, while on multi-modal functions is 0.29.) Their statistical information is shown in Fig. 6 except that of DE due to the lack of data.

On unimodal functions, IPOP-CMA-ES performs by far the best, while the other four algorithms perform closely.

On multi-modal and composition functions, judging from the ARs, the LoTFWA performs the best. The ARs of ABC and DE are comparable, which are better than that of SPSO2011. ABC, the LoTFWA and IPOP-CMA-ES achieve 8, 8 and 7 of the minimal mean errors on multi-modal and composition functions respectively, while SPSO2011 and DE achieve none. Although ABC performs very well on eight functions, its AR implies that it ranks lowly on the other functions.

CMA-ES is a very powerful evolutionary algorithm for uni-modal optimization, but does not necessarily perform well on multi-modal functions when it suffers from the problem of premature convergence [38]. Thus, restart mechanisms have been introduced in CMA-ES to enhance the probability of finding the global optimum. The population in CMA-ES actually has many chances to restart due to its very fast convergence. However, IPOP-CMA-ES does not outperform the LoTFWA on multi-modal functions generally even though the exploitation capability of the FWA is just ordinary (comparable to ABC, PSO and DE). In particular, on composition functions where local search becomes more difficult, the LoTFWA performs better (see F21, F24, F25, F26, F27, F28). Based on these experimental results, we think using the competitive interaction among multiple populations may be a more promising methodology for multi-modal optimization than restarting a single population. In fact, the LoT proposed in this paper can be transplanted to almost any population-based algorithms.

CMA-ES is a quadratic time algorithm while the others are linear time algorithms. It takes about 2.2 hours for the LoTFWA to optimize all the 28 functions for 51 times when $d = 30$ on the Intel I7-6700HQ CPU (2.6GHz) and MATLAB R2016a.

VI. Conclusion

This paper proposes a loser-out tournament based fireworks algorithm for multi-modal optimization. Experimental results on the CEC 2013 single objective benchmark suite indicate that the proposed algorithm is powerful in multi-modal function optimization even though its local search capability is not outstanding. Therefore, we believe the proposed framework and the interactive mechanism are efficient for multi-modal optimization. In addition, it can be easily embedded within other population-based algorithms.

The LoTFWA should be considered as a new baseline for the development of the fireworks algorithm. There are a lot of works to be done to enhance the local search capability. Besides, other kinds of interactive mechanisms are also worth further investigation.

References

[1] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Advances in Swarm Intelligence*. Springer, 2010, pp. 355–364.

[2] S. Zheng, J. Li, A. Janecek, and Y. Tan, "A cooperative framework for fireworks algorithm," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 27–41, Jan 2017.

[3] Y. Tan, "Cooperative fireworks algorithm," in *Fireworks Algorithm*. Springer, 2015, pp. 133–149.

[4] B. Zhang, Y. J. Zheng, M. X. Zhang, and S. Y. Chen, "Fireworks algorithm with enhanced fireworks interaction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 42–55, Jan 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2787042, IEEE Transactions on Evolutionary Computation

Fig. 6: Boxplots of the Four Algorithms

TABLE IV: Mean Errors, Standard Deviations and Average Ranks of the Five Evolutionary Algorithms

	ABC		SPSO2011		IPOP-CMAES		DE		LoTFWA	
F.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**1.88E-13**	**0.00E+00**	**0.00E+00**	1.89E-03	4.65E-04	**0.00E+00**	**0.00E+00**
2	6.20E+06	1.62E+06	3.38E+05	1.67E+05	**0.00E+00**	**0.00E+00**	5.52E+04	2.70E+04	1.19E+06	4.27E+05
3	5.74E+08	3.89E+08	2.88E+08	5.24E+08	**1.73E+00**	**9.30E+00**	2.16E+06	5.19E+06	2.23E+07	1.91E+07
4	8.75E+04	1.17E+04	3.86E+04	6.70E+03	**0.00E+00**	**0.00E+00**	1.32E-01	1.02E-01	2.13E+03	8.11E+02
5	**0.00E+00**	**0.00E+00**	5.42E-04	4.91E-05	**0.00E+00**	**0.00E+00**	2.48E-03	8.16E-04	3.55E-03	5.01E-04
AR.	3.4		3		**1**		3		3.2	
6	1.46E+01	4.39E+00	3.79E+01	2.83E+01	**0.00E+00**	**0.00E+00**	7.82E+00	1.65E+01	1.45E+01	6.84E+00
7	1.25E+02	1.15E+01	8.79E+01	2.11E+01	**1.68E+01**	**1.96E+01**	4.89E+01	2.37E+01	5.05E+01	9.69E+00
8	2.09E+01	4.97E-02	2.09E+01	5.89E-02	2.09E+01	5.90E-02	2.09E+01	5.65E-02	**2.09E+01**	**6.14E-02**
9	3.01E+01	2.02E+00	2.88E+01	4.43E+00	2.45E+01	1.61E+01	1.59E+01	2.69E+00	**1.45E+01**	**2.07E+00**
10	2.27E-01	6.75E-02	3.40E-01	1.48E-01	**0.00E+00**	**0.00E+00**	3.24E-02	1.97E-02	4.52E-02	2.47E-02
11	**0.00E+00**	**0.00E+00**	1.05E+02	2.74E+01	2.29E+00	1.45E+00	7.88E+01	2.51E+01	6.39E+01	1.04E+01
12	3.19E+02	5.23E+01	1.04E+02	3.54E+01	**1.85E+00**	**1.16E+00**	8.14E+01	3.00E+01	6.82E+01	1.45E+01
13	3.29E+02	3.91E+01	1.94E+02	3.86E+01	**2.41E+00**	**2.27E+00**	1.61E+02	3.50E+01	1.36E+02	2.30E+01
14	**3.58E-01**	**3.91E-01**	3.99E+03	6.19E+02	2.87E+02	2.72E+02	2.38E+03	1.42E+03	2.38E+03	3.13E+02
15	3.88E+03	3.41E+02	3.81E+03	6.94E+02	**3.38E+02**	**2.42E+02**	5.19E+03	5.16E+02	2.58E+03	3.83E+02
16	1.07E+00	1.96E-01	1.31E+00	3.59E-01	2.53E+00	2.73E-01	1.97E+00	2.59E-01	**5.74E-02**	**2.13E-02**
17	**3.04E+01**	**5.15E-03**	1.16E+02	2.02E+01	3.41E+01	1.36E+00	9.29E+01	1.57E+01	6.20E+01	9.45E+00
18	3.04E+02	3.52E+01	1.21E+02	2.46E+01	8.17E+01	6.13E+01	2.34E+02	2.56E+01	**6.12E+01**	**9.56E+00**
19	**2.62E-01**	**5.99E-02**	9.51E+00	4.42E+00	2.48E+00	4.02E-01	4.51E+00	1.30E+00	3.05E+00	6.43E-01
20	1.44E+01	4.60E-01	1.35E+01	1.11E+00	1.46E+01	3.49E-01	1.43E+01	1.19E+00	**1.33E+01**	**1.02E+00**
21	**1.65E+02**	**3.97E+01**	3.09E+02	6.80E+01	2.55E+02	5.03E+01	3.20E+02	8.55E+01	2.00E+02	2.80E-03
22	**2.41E+01**	**2.81E+01**	4.30E+03	7.67E+02	5.02E+02	3.09E+02	1.72E+03	7.06E+02	3.12E+03	3.79E+02
23	4.95E+03	5.13E+02	4.83E+03	8.23E+02	**5.76E+02**	**3.50E+02**	5.28E+03	6.14E+02	3.11E+03	5.16E+02
24	2.90E+02	4.42E+00	2.67E+02	1.25E+01	2.86E+02	3.02E+01	2.47E+02	1.54E+01	**2.37E+02**	**1.20E+01**
25	3.06E+02	6.49E+00	2.99E+02	1.05E+01	2.87E+02	2.85E+01	2.80E+02	1.57E+01	**2.71E+02**	**1.97E+01**
26	2.01E+02	1.93E-01	2.86E+02	8.24E+01	3.15E+02	8.14E+01	2.52E+02	6.83E+01	**2.00E+02**	**1.76E-02**
27	**4.16E+02**	**1.07E+02**	1.00E+03	1.12E+02	1.14E+03	2.90E+02	7.64E+02	1.00E+02	6.84E+02	9.77E+01
28	**2.58E+02**	**7.78E+01**	4.01E+02	4.76E+02	3.00E+02	0.00E+00	4.02E+02	3.90E+02	2.65E+02	7.58E+01
AR.	3.13		3.96		2.57		3.30		**2.04**	

[5] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[6] I. Rechenberg, "Evolution strategy," *Computational intelligence: Imitating life*, vol. 1, 1994.

[7] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[8] C.-R. Hwang, "Simulated annealing: theory and applications," *Acta Applicandae Mathematicae*, vol. 12, no. 1, pp. 108–111, 1988.

[9] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[10] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[11] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr 1997.

[12] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the third annual conference on evolutionary programming*, vol. 131139. Singapore, 1994.

[13] P. Moscato *et al.*, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[14] T. Bäck, "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, Jun 1994, pp. 57–62 vol.1.

[15] ——, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[16] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, April 2015.

[17] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search: Framework and Applications*. Boston, MA: Springer US, 2010, pp. 363–397.

[18] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1769–1776.

[19] O. M. Shir, "Niching in evolutionary algorithms," in *Handbook of Natural Computing*. Springer, 2012, pp. 1035–1069.

[20] D. Zaharie, "A multipopulation differential evolution algorithm for multimodal optimization," in *Proceedings of Mendel*, 2004, pp. 17–22.

[21] B. Niu, Y. Zhu, and X. He, "Multi-population cooperative particle swarm optimization," in *Advances in Artificial Life*. Springer, 2005, pp. 874–883.

[22] T. Blackwell and J. Branke, *Multi-swarm Optimization in Dynamic Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 489–500.

[23] J. Branke, T. Kaussler, C. Smidt, and H. Schmeck, *A Multi-population Approach to Dynamic Optimization Problems*. London: Springer London, 2000, pp. 299–307.

[24] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, June 1993.

[25] X. Yang and Y. Tan, "Sample index based encoding for clustering using evolutionary computation," in *Advances in Swarm Intelligence*. Springer, 2014, pp. 489–498.

[26] K. Ding, Y. Chen, Y. Wang, and Y. Tan, "Regional seismic waveform inversion using swarm intelligence algorithms," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 1235–1241.

[27] N. Bacanin and M. Tuba, "Fireworks algorithm applied to constrained portfolio optimization problem," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 1242–1249.

[28] H. A. Bouarara, R. M. Hamou, A. Amine, and A. Rahmani, "A fireworks algorithm for modern web information retrieval with visual results mining," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 6, no. 3, pp. 1–23, 2015.

[29] A. Rahmani, A. Amine, R. M. Hamou, M. E. Rahmani, and H. A. Bouarara, "Privacy preserving through fireworks algorithm based model for image perturbation in big data," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 6, no. 3, pp. 41–58, 2015.

[30] J. Liu, S. Zheng, and Y. Tan, "Analysis on global convergence and time complexity of fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3207–3213.

[31] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2069–2077.

[32] J. Liu, S. Zheng, and Y. Tan, "The improvement on controlling exploration and exploitation of firework algorithm," in *Advances in swarm intelligence*. Springer, 2013, pp. 11–23.

[33] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3214–3221.

[34] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3222–3229.

[35] Y. Tan, C. Yu, S. Zheng, and K. Ding, "Introduction to fireworks algorithm," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 4, no. 4, pp. 39–70, 2013.

[36] Wikipedia, "Power law distribution," 2017, [Accessed 27-February-2017]. [Online]. Available: https://en.wikipedia.org/wiki/Power_law#Power-law_probability_distributions

[37] ——, "80/20 rule," 2017, [Accessed 27-February-2017]. [Online]. Available: https://en.wikipedia.org/wiki/Pareto_principle

[38] J. Li, S. Zheng, and Y. Tan, "The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 153–166, Feb 2017.

[39] J. Li and Y. Tan, "The bare bones fireworks algorithm: A minimalist global optimizer," *Applied Soft Computing*, vol. 62, no. Supplement C, pp. 454 – 462, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494617306609

[40] Y.-J. Zheng, X.-L. Xu, H.-F. Ling, and S.-Y. Chen, "A hybrid fireworks optimization method with differential evolution operators," *Neurocomputing*, vol. 148, pp. 75 – 82, 2015.

[41] C. Yu, L. Kelley, S. Zheng, and Y. Tan, "Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3238–3245.

[42] P. Larranaga, "A review on estimation of distribution algorithms," in *Estimation of distribution algorithms*. Springer, 2002, pp. 57–100.

[43] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 35:1–35:33, Jul. 2013.

[44] J. Horn, "The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations," Ph.D. dissertation, Citeseer, 1997.

[45] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[46] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 869–880, Jul 1996.

[47] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 103–127, 2009.

[48] J. N. Thompson, "The coevolutionary process," *The Quarterly Review of Biology*, vol. 64, no. Volume 71, Number 3, 1996.

[49] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2238–2251, Oct 2016.

[50] S. W. Mahfoud, "Niching methods for genetic algorithms," *Urbana*, vol. 51, no. 95001, pp. 62–94, 1995.

[51] M. R. R. N. and Z. Kobti, "Heterogeneous multi-population cultural algorithm," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 292–299.

[52] S. Biswas, S. Kundu, D. Bose, S. Das, P. N. Suganthan, and B. K. Panigrahi, "Migrating forager population in a multi-population artificial bee colony algorithm with modified perturbation schemes," in *2013 IEEE Symposium on Swarm Intelligence (SIS)*, April 2013, pp. 248–255.

[53] W. Sheng, S. Chen, M. Sheng, G. Xiao, J. Mao, and Y. Zheng, "Adaptive multisubpopulation competition and multiniche crowding-based memetic algorithm for automatic data clustering," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 838–858, Dec 2016.

[54] Wikipedia, "Single combat," 2017, [Accessed 27-February-2017]. [Online]. Available: https://en.wikipedia.org/wiki/Single_combat

[55] C. Chatfield, *Time-series forecasting*. CRC Press, 2000.

[56] M. E. H. Pedersen, "Tuning & simplifying heuristical optimization," 2010.

[57] J. Liang, B. Qu, P. Suganthan, and 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*, vol. 201212, 2013.

[58] M. Clerc, "Standard particle swarm optimization, from 2006 to 2011," *Particle Swarm Central*, 2011.

[59] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2337–2344.

[60] M. El-Abd, "Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2215–2220.

[61] I. Loshchilov, "CMA-ES with restarts for solving CEC 2013 benchmark problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 369–376.

[62] N. Padhye, P. Mittal, and K. Deb, "Differential evolution: Performances and analyses," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 1960–1967.

[63] P. N. Suganthan, "Results of 22 papers," 2017, [Accessed 27-February-2017]. [Online]. Available: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2013/Results-of-22-papers.zip

Junzhi Li received the B.S. degree from the Department of Machine Intelligence and a minor degree in philosophy from the Department of Philosophy and Religious Studies from Peking University, Beijing, China, in 2013, where he is currently pursuing the Ph.D. degree with the Key Laboratory of Machine Perception (Ministry of Education) and the Department of Machine Intelligence, School of Electronics Engineering and Computer Science.

His current research interests include evolutionary computation, artificial neural networks, and machine learning.

Ying Tan is a full professor and PhD advisor at the School of Electronics Engineering and Computer Science of Peking University, and director of Computational Intelligence Laboratory at Peking University. He received his BEng, MS, and PhD from Southeast Univ., in 1985, 1988, and 1997, respectively. He worked at Chinese University of Hong Kong in 1999 and 2004-2005, and visited many universities including Columbia University, Kyushu University, Auckland University of Technology, etc. He is the inventor of Fireworks Algorithm (FWA). He serves as the Editor-in-Chief of International Journal of Computational Intelligence and Pattern Recognition (IJCIPR), the Associate Editor of IEEE Transactions on Evolutionary Computation (TEC), IEEE Transactions on Cybernetics (CYB), IEEE Transactions on Neural Networks and Learning Systems (NNLS), etc. He also served as an Editor of Springers Lecture Notes on Computer Science (LNCS) for 28+ volumes, and Guest Editors of several referred Journals, including IEEE/ACM Transactions on Computational Biology and Bioinformatics, Information Science, Neurocomputing, Natural Computing, Softcomputing, etc. He is a member of Emergent Technologies Technical Committee (ETTC) of IEEE Computational Intelligence Society since 2010. He is the founder and chair of the ICSI International Conference series. He was the general chair of joint general chair of 1st&2nd BRICS Congress of Computational Intelligence, program committee co-chair of IEEE WCCI 2014, etc. He won the 2nd-Class Natural Science Award of China in 2009. His research interests include computational intelligence, swarm intelligence, data mining, intelligent information processing for information security etc. He has published more than 280 papers in refereed journals and conferences in these areas, and authored/co-authored 11 books and 16 chapters in book, and received 4 invention patents.