Orthogonal Learning Firefly Algorithm

Tomas Kadavy, Roman Senkerik, Michal Pluhacek and Adam Viktorin

Tomas Bata University in Zlin, T.G.Masaryka 5555, 760 01 Zlin, Czech Republic
{kadavy, senkerik, pluhacek, aviktorin}@utb.cz

Abstract. The primary aim of this original work is to provide a more in-depth insight into the relations between control parameters adjustments, learning techniques, inner swarm dynamics and possible hybridization strategies for popular swarm metaheuristic Firefly Algorithm (FA). In this paper, a proven method, orthogonal learning, is fused with FA, specifically, with its hybrid modification Firefly Particle Swarm Optimization (FFPSO). The parameters of the proposed Orthogonal Learning Firefly Algorithm (OLFA) are also initially thoroughly explored and tuned. The performance of the developed algorithm is examined and compared with canonical FA and above mentioned FFPSO. Comparisons have been conducted on well-known CEC 2017 benchmark functions, and the results have been evaluated for statistical significance using the Friedman rank test.

Keywords: Firefly Algorithm, Particle Swarm Optimization, Orthogonal learning, Friedman rank

1 Introduction

The swarm-based algorithms, besides the other algorithms like nature-inspired or non-swarm algorithms, are quite popular among researchers interested in heuristic optimization. The most popular amongst them is probably the Particle Swarm Optimization (PSO) algorithm. The PSO was developed in 1995 by Eberhart and Kennedy [1]. Since the PSO introduction, it still attracts the attention of many researchers. Many extensive studies have been performed so far, for example studying the impact of several possible configurations or parameter settings of PSO [2, 3]. Even more complex studies have been carried out investigating velocity clamping [4], population restart [5, 6], various hybridizations [7 - 9], and boundary strategies [10, 11]. All the studied features have some performance impact on the PSO in general. As one could imagine, similar studies are provided for other swarm-based algorithms. Because of the mutual similarity of particular groups of swarm algorithms, some findings or conclusions for one algorithm could be re-applied to others. One of the possible and popular adjustment is based on orthogonal learning [12]. The basic idea behind orthogonal learning lies in the identification either of combinations or inherent parts of solutions in particular dimensions offering the best results and using them in the next optimization steps. The principle of orthogonal learning will be explained in details within the relevant section of this paper. For several years, another swarm-based algorithm proves its applicability and is becoming quite popular among researchers. The Firefly Algorithm (FA) was introduced

adfa, p. 1, 2011.

in 2008 [13]. Since then, like for the PSO, many extensions and modifications were proposed for this effective optimization algorithm [14 - 19]. The FA was successfully used for various optimization problems; for example, the design of antenna [20], job scheduling [21], and solving the traveling salesman problem [22]. The relative novelty of the FA, compared to the PSO, is the reason that there are still only a few studies published about its various features and inner dynamic [23] (again compared to the PSO). The popularity and versatility of both algorithms, PSO and FA, has led to the creation of a hybrid algorithm, which is called Firefly Particle Swarm Optimization (FFPSO) [24]. The overall popularity of FA and PSO can be derived based on the number of recent papers that use one of the algorithms [38 - 42].

Nowadays [25], instead of developing new heuristic optimization algorithms, the smart improvements of existing ones are more favorable. Several modern techniques were designed to improve overall performance. For example, the ensemble method [26, 27], the above-mentioned hybridization [28, 29], or adaptive control [30, 31] were already adopted. Generally speaking, all the mentioned methods are aimed at either the analyses and improvements of the inner dynamic of a particular algorithm or on the combination of the various strategies borrowed from different optimization algorithms. The basic idea behind the hybridization of several different metaheuristic algorithms is that the resulting hybrid approach should combine the advantages of all donor algorithms, hence to eliminate their disadvantages. However, achieving this ideal state is a difficult task, and it is a clear motivation behind the presented research.

The effective technique already used with PSO algorithm, the orthogonal learning [32], could secure a possible performance improvement of FA. However, the enhancement of the canonical FA may be a difficult task, due to its nature. Nevertheless, thanks to the above-mentioned hybrid FFPSO, the advantages of orthogonal learning could be accessible, and the application is much more simplified.

The motivation and the originality of the presented research can be summarized as follows:

- To present a comprehensive review of the control parameter adjustments schemes, possible hybridization of FA and orthogonal learning technique, so that the readers can easily navigate between different parameter settings/hybridization strategies, and to see the direct comparisons of performances and deeper insight into swarm dynamics.
- The proposed optimization algorithm is tested and statistically evaluated on well-known benchmark CEC 2017 [33].
- The extensive parameters tuning for the new proposed algorithm are also present here, evaluated on the very same benchmark suite and conclusions are supported by Friedman rank test [34].
- The comparisons with original hybridized algorithm FFPSO and with the original version of the FA, PSO and Orthogonal Learning Particle Swarm Optimization (OLPSO) are presented here and statistically evaluated as well.

In this extended version of the original paper [43], the comprehensive motivation of proposed research is presented with more detailed results of the proposed algorithm's performance. The section with tunning of algorithm parameters is also included. The

part, parameter tunning, contains explanations of how each of the settings influences the overall performance of the proposed algorithm. Finally, this extended paper revised the description of the covered algorithms, and the pseudocodes are added.
The rest of the paper is structured as follows; the brief backgrounds of PSO, OLPSO, FA, and FFPSO are given in sections II and III. These sections also cover a description of the proposed novel hybrid approach based on orthogonal learning together with the results of the influence of its parameters on the overall performance. In section IV, the experiment design is given, i.e., CEC benchmark setup, and the parameter adjustment for all compared algorithms. The results and conclusion sections follow afterward.

2 Particle Swarm Optimization

The PSO is one of the most prominent representatives on the field of swarm intelligence based algorithms. It was first published by Eberhart and Kennedy in 1995 [1]. The algorithm mimics the social behavior and movement patterns of swarming animals in nature. In this work, two variants of PSO are utilized, the original (canonical) PSO and the Orthogonal learning PSO.

2.1 Canonical Particle Swarm Optimization

The algorithm simulates a movement of a swarm of artificial particles in n-dimensional space. The position of each particle represents a tentative solution for the optimized problem. The iterative movement of the particles is driven by a velocity vector, labeled as $\boldsymbol{v}$. Each particle remembers its best position (solution to the problem) obtained so far. This solution is tagged as the ***pBest***, personal best solution. Further, each particle has access to the global best solution, ***gBest***, which is selected from all ***pBest*s**.

In every iteration of the algorithm, the new position of each particle is calculated based on the previous position and new velocity.

The new position of the particle is calculated according to the formula (1).

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \boldsymbol{v}_i' \tag{1}$$

Where $\boldsymbol{x}_i'$ is a new position of particle i, $\boldsymbol{x}_i$ is the previous old position of a particle and $\boldsymbol{v}_i'$ is the new velocity of a particle, calculated according to (2).

$$\boldsymbol{v}_i' = w \cdot \boldsymbol{v}_i + c_1 \cdot \boldsymbol{r_1} \cdot (\boldsymbol{pBest}_i - \boldsymbol{x}_i) + c_2 \cdot \boldsymbol{r_2} \cdot (\boldsymbol{gBest} - \boldsymbol{x}_i) \tag{2}$$

Where w is inertia weight [35], $\boldsymbol{v}_i$ is the old velocity, c_1 and c_2 are learning factors, and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are vectors of random numbers drawn from unimodal distribution in the range <0,1>.

2.2 Orthogonal Learning Particle Swarm Optimization

The Orthogonal Experimental Design (OED) mechanism was introduced to PSO to improve its learning strategy. This improvement led to the creation of Orthogonal

Learning PSO (OLPSO) [12]. As the name suggests, the traditional learning mechanism of PSO was replaced by novel Orthogonal Learning, which should help construct an efficient and promising exemplar for a particle to learn from. Each corresponding dimension of an optimized problem is regarded as a factor. The factor means that the OLPSO combines information of the particle's ***pBest*** and the ***pBest*s** of its neighborhoods to construct a guidance vector. The velocity equation (2) is then changed to (3).

$$\boldsymbol{v}_i' = w \cdot \boldsymbol{v}_i + c \cdot r \cdot (\boldsymbol{gVector}_i - \boldsymbol{x}_i) \tag{3}$$

Where ***gVector*** is the guidance vector, the values of ***gVector*** are just pointers to particular ***pBest's*** components that should be used in corresponding dimensions. The guidance vector (learning exemplar) is used until it cannot improve the particles ***pBest*** solution for a certain number of iterations that is called reconstruction gap *G*.

The brief process of the construction of the *gVector* is as follows:

1. An orthogonal array (OA) $L_M(2^D)$, where $\boldsymbol{M} = 2^{\lfloor \log_2(d+1) \rfloor}$ is created using the procedure which is clearly and more detailed described in [12].
2. According to the OA, the ***M*** trial solutions are created by selecting the corresponding value from ***pBest*s**. Own ***pBest*** or ***pBest*** of a different particle can be selected for this operation.
3. Each trial solution is evaluated, and the best solution is recorded as $\boldsymbol{X_b}$.
4. Calculate the effect of each level on each factor and determine the best level for each factor. Based on these levels, the predictive solution $\boldsymbol{X_P}$ is created and evaluated.
5. The solution ($\boldsymbol{X_b}$ or $\boldsymbol{X_P}$) is selected as the guidance vector ***gVector*** based on the objective function value.

3 Firefly Algorithm

This nature-based optimization algorithm was developed and introduced by Yang in 2008 [13]. The fundamental principle of this algorithm lies in simulating the mating behavior of fireflies at night when fireflies emit light to attract a suitable partner. The FA was also successfully used for many optimization problems [20 - 22].

3.1 Canonical Firefly Algorithm

The movement of one firefly towards another one is defined by equation (4), where $\boldsymbol{x}_{ik}^{t+1}$ is a new position of firefly *i* for dimension *k*, $\boldsymbol{x}_{ik}^{t}$ is the current position of firefly *i* and $\boldsymbol{x}_{jk}^{t}$ is a selected brighter firefly (with better objective function value). Parameter α is a randomization parameter ($\alpha \in \langle 0, 1 \rangle$). The original FA uses the random value drawn from the uniform distribution. Finally, *sign* provides random direction -1 or 1 to ensure that the firefly could travel in both directions.

$$\boldsymbol{x}_{ik}^{t+1} = \boldsymbol{x}_{ik}^{t} + \beta_i \cdot \left(\boldsymbol{x}_{jk}^{t} - \boldsymbol{x}_{ik}^{t}\right) + \alpha \cdot sign \tag{4}$$

The brightness I_i of a firefly is computed by the equation (5) where $(\boldsymbol{x}_i)$ is the CF value of corresponding *i*-firefly, γ stands for the light absorption parameter of a media in which fireflies are and, m is another user-defined coefficient and, it should be set $m \geq 1$. The variable r_{ij} is the Euclidian distance (6) between the two compared fireflies (*d* stands for the dimension size of the optimized problem). The firefly $\boldsymbol{x}_i$ could only fly towards the $\boldsymbol{x}_j$ firefly if $I_j < I_i$.

$$I_i = \frac{f(\boldsymbol{x}_i)}{1+\gamma r_{ij}^m} \tag{5}$$

$$r_{ij} = \sqrt{\sum_{k=1}^{d} \left(\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}\right)^2} \tag{6}$$

$$\beta_i = \frac{\beta_i'}{1+\gamma r_{ij}^m} \tag{7}$$

The attractiveness β_i (7) is proportional to the brightness I_i as mentioned in the rules above and, so these equations are quite similar to each other. The β_i' is the initial attractiveness defined by the user, the γ is again the light absorption parameter and the r_{ij} is once more the Euclidian distance.

3.2 Hybrid of Firefly and Particle Swarm Optimization Algorithms

Another more advanced version of the FA could be created by its hybridization with other successful metaheuristic algorithms. The basic idea behind such an approach is that the new hybrid strategy can share advantages from both algorithms and possibly eliminate their disadvantages.

The typical example is a hybrid of the FA and PSO algorithms, the FFPSO [23] introduced in late 2015 by Padmavathi Kora and K. Sri Rama Krishna. The central principle remains the same as in the standard FA, but the equation for firefly motion (4) is slightly changed according to PSO movement and is newly computed as (8).

$$\boldsymbol{x}_i' = w\boldsymbol{x}_i + c_1 e^{-r_{px}^2}(\boldsymbol{pBest}_i - \boldsymbol{x}_i) + c_2 e^{-r_{gx}^2}(\boldsymbol{gBest} - \boldsymbol{x}_i) + \alpha \cdot sign \tag{8}$$

Where *w*, c_1, and c_2 are control parameters transferred from PSO and their values often depends on the user. Similarly, the ***pBest*** and ***gBest*** are variables formerly belonging to the PSO algorithm. They both represent the memory of the best position where ***pBest*** is best position of each particle and ***gBest*** is globally achieved best position so far. The remaining variables r_{px} (9) and r_{gx} (10) are distances between particle x_i and both $\boldsymbol{pBest}_i$ and ***gBest***.

$$r_{px} = \sqrt{\sum_{k=1}^{d} \left(pBest_{i,k} - x_{i,k}\right)^2} \tag{9}$$

$$r_{gx} = \sqrt{\sum_{k=1}^{d} \left(gBest_k - x_{i,k}\right)^2} \tag{10}$$

3.3 Orthogonal Learning Firefly Algorithm

Our proposed algorithm, OLFA, is based on FFPSO mentioned above and it uses the orthogonal learning technique. Our application of orthogonal learning is similar to the Orthogonal Learning Particle Swarm Optimization (OLPSO). The OLFA also generates the promising learning exemplar by adopting an orthogonal learning strategy for each particle to learn from. This means that the equation of firefly movement (8) is slightly changed and does not contain the ***pBest*** and ***gBest*** any longer. The new equation (11) includes only the trial exemplar ***gVector***.

$$\boldsymbol{x_i'} = w\boldsymbol{x_i} + c \cdot e^{-r^2}(\boldsymbol{gVector_i} - \boldsymbol{x_i}) + \alpha \cdot sign \tag{11}$$

Where Euclidian distance r is computed as (12) between firefly $\boldsymbol{x_i}$ and trial ***gVector***.

$$r = \sqrt{\sum_{k=1}^{d}\left(gVector_{i,k} - x_{i,k}\right)^2} \tag{12}$$

The ***gVector*** is used as the guide for each particle until it cannot improve the solution quality for a certain number of iterations, which is called refreshing gap G. When the number of non-improved iterations reaches the refreshing gap limit, the learning ***gVector*** is reconstructed. The (re)construction process of ***gVector*** is the same as described in Section 2.2. The randomization parameter α is no longer a fixed value but computed by (13) [24]. This ensures that the random step will be proportional to a given parameter range of the optimized problem. The $U(0,1)$ represents a random value in a range <0, 1> from a uniform distribution. The b^u and b^l are upper and lower boundary limits of optimized parameters of a given problem.

$$\alpha = 0.5 \cdot [U(0,1) - 0.5] \cdot |b^u - b^l| \tag{13}$$

The pseudocode 1 below shows the fundamentals of OLFA operations.

Algorithm 1 OLFA

```
OLFA initialization
for i = 1 to NP do
    generate gVector_i
    counter_i = 0
end for
while iteration < max_iteration do
    for i = 1 to NP do
          calculate r by (12)
          move x_i by (11)
          if x_i not improved then
                counter_i++
                if counter_i > gap then
                      counter_i = 0
                      generate gVector_i
                end if
          end if
    end for
    record the best solution
end while
```

4 Experiment Setup

The performance of the newly proposed hybrid algorithm depends on the setting of its control parameters. Although OLFA is a combination of several algorithms, it utilizes only three user-defined parameters (inertia weight *w*, learning coefficient *c*, and a refreshing gap *G*). It is unlikely that the values of these parameters recommended for original versions could still be applied without further analysis. Therefore, a tuning experiment of the control parameters of OLFA was performed.

4.1 Tuning experiment

Firstly, the influence of a learning parameter *c*, which is adapted from original PSO, was tested. The tested values of *c* were selected as {0.8, 1.0, 1.2, 1.4, 1.7, 2}. These values are similar as typically set for PSO [4]. Other parameters were fixed on original values: $G = 5$, and $w = \langle 0.9 - 0.4 \rangle$. Note that inertia weight *w* can be a single numeric variable or a linear decreasing over iterations of an algorithm by (14). Where w_S and w_E are max and min values of *w. iteration* is the current iteration of the algorithm, and the *max_iterations* represents a maximum number of iterations defined by the user. This test used the linear decreasing inertia weight.

$$w = w_S - \frac{(w_S - w_E) \cdot iteration}{\max\_iterations} \tag{14}$$

Second test case tested the influence of the refreshing gap *G*. This time the fixed parameters were set as $c = 2$ and $w = \langle 0.9 - 0.4 \rangle$. The refreshing gap *G* was tested for values {2, 5, 7, 11, 13, 17, 20, 30, 40, 50, 60, 70, 80, 90, 100}.

Last test case studied the influence of inertia weight *w* setting. The parameters refreshing gap and learning coefficient were fixed on values $G = 5$ and $c = 2$. The constant values of *w* were tested for {0.7, 0.8, 0.9, 1.0, 1.1, 1.2}. The linear decreasing inertia weight was also tested, the w_E was fixed on 0.4 and the w_S values were {0.7, 0.8, 0.9, 1.0, 1.1, 1.2}.

The results were tested for statistical significance using the Friedman Rank test [33]. The null hypothesis that the means are equal is rejected at the 5% level based on the Friedman Rank test. The corresponding p-values of Friedman Rank tests are presented in Table 1.

Table 1. *p*-values of performed Friedman rank tests.

Test case (tested parameter)	*p*-value
c	0.26
w	$1.97 \cdot 10^{-56}$
G	$2.03 \cdot 10^{-152}$

If the p-value is lower than 0.05, the further Friedman rankings are relevant. In Figure 1, the Friedman rankings for the different tested parameter influence for dimension size

d = 10 are shown. The lower the rank is, the better is the performance of the labeled setting. Furthermore, the presented Friedman ranks are accompanied by critical distance evaluated according to the Nemenyi Critical Distance post-hoc test for multiple comparisons. The dashed line represents the critical distance from the best setting (the lowest mean rank).

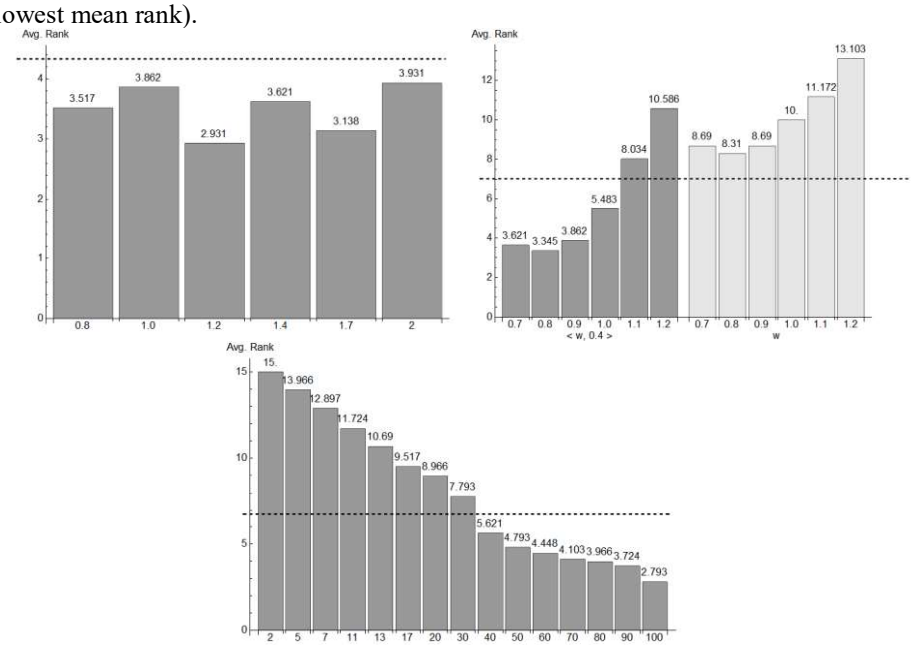


Fig. 1. Friedman rank tests for different parameter tests: learning coefficient c (top left), inertia weight w (top right) and a refreshing gap G (bottom).

According to results presented in Figure 1 (top left), the value of a learning coefficient c has no significant influence on the performance of the OLFA. The provided p-value in Table 1 support this finding because the null hypothesis is rejected (p-value is lower than 0.05). However, this is only valid for the selected range of values. Different values of c may change the quality of the results. Even the different combination of fixed parameters may interfere with this conclusion.

As for the gap value G, based on the obtained results depicted in Figure 1 (bottom) the higher the gap value, the better is the overall performance. The higher values are likely beneficial because fireflies have a longer time to explore possible solutions of a guidance vector. Another reason may be that when a new guidance vector has to be created, several evaluations of potential solutions have to be made (without any improvement of an obtained best solution) thus less objective function evaluations can be used to explore the possible parameters. However, it is possible that there is some upper limit for this refreshing gap value, where there is no more space for development. This limit could be connected to the number of fireflies (NP) or the maximum number of iteration of the algorithm. If the ratio among these settings is low, then a firefly has no time to examine the selected guidance vector properly and has no time to create a new one.

Finally, from the obtained Friedman rank test depicted in Figure 1 (top right), the linear decreasing w is more beneficial to the overall performance of OLFA. Particularly the setting where $w = <0.8 - 0.4>$. It is worth mention that if a $w > 1$ then the behavior of OLFA may not convert to an optimal solution because the newly generated position will lie most likely behind the guidance vector.
Based on the performed tests, the parameter values for refreshing gap $G = 100$, learning coefficient $c = 1.2$ and inertia weight $w = <0.8 - 0.4>$ give the average best results.

4.2 Performance test

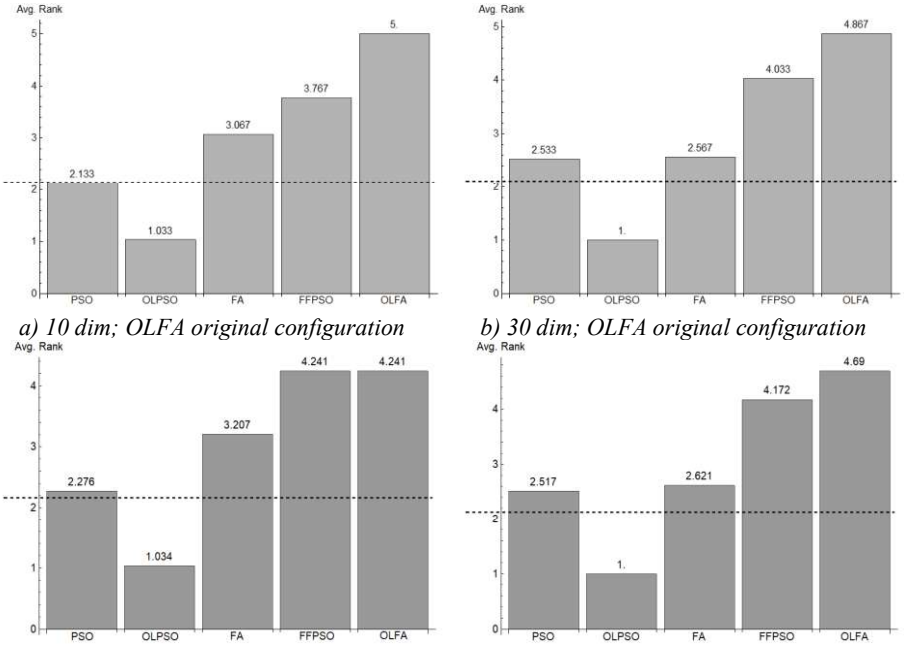
The performance tests of proposed OLFA were performed on a set of well-known benchmark functions CEC'17 that are detailly described in [33]. The tested dimensions d were 10 and 30. The maximal number of function evaluation was set as $10\,000 \cdot d$ (dimension size). The lower and upper boundaries of the search space were set as $b^l = -100$ and $b^u = 100$ according to the CEC'17 definition. The number of particles (fireflies) was set to 40 for all dimension sizes. Every test function was tested for 51 independent runs, and the results were statistically evaluated. The benchmark includes 30 test functions in four categories: unimodal, multimodal, hybrid and composite. However, due to some technical difficulties, the authors of bencthe hmark recommend to skip test function f_2. The global minimum of each function is easy to determine as it is $f_i \cdot 100$ where i is an order of test function.
The control parameters settings for all tested and compared algorithms are given in Table 2. Parameters were set to optimal values according to literature. The proposed OLFA algorithm was tested with the parameters determined by tests performed in previous *Section 4.1*.

Table 2. Parameters of tested algorithms

Name	Parameters
PSO	$w = 0.729, c_1 = c_2 = 1.49445$
OLPSO	$w = 0.9 - 0.4, c = 2, G = 5$
FA	$\alpha = 0.5, \beta_0 = 0.2, \gamma = 1,$
FFPSO	$\alpha = 0.5, \beta_0 = 0.2, \gamma = 1$ $w = 0.729, c = 1.49445$
OLFA	$w = 0.8 - 0.4, c = 1.2, G = 100$

5 Results

The results of the performance testing are reported in this section. The overall performances of all tested algorithms on dimension sizes 10 and 30 are evaluated and compared using Friedman ranks with critical distance assessed according to the Nemenyi Critical Distance post-hoc test for multiple comparisons. The visual outputs of various comparisons with rankings are given in Figure 2. All Friedman rank test hypotheses are relevant with a p-value lower than 0.05 as presented in Table 3. Figure 2 depicts four different comparisons. The first two parts *a)* and *b)* are for dimension sizes 10 and 30 with OLFA parameters set to default values inherited from the original algorithms, on which OLFA is based. The parts *c)* and *d)* are again for dimension sizes 10 and 30 but this time for tuned parameters of OLFA.



a) 10 dim; OLFA original configuration

b) 30 dim; OLFA original configuration

c) 10 dim; OLFA tuned parameters

d) 30 dim; OLFA tuned parameters

Fig. 2. Friedman rank test results.

Table 3. p-values of performed Friedman rank tests presented in Fig. 2.

	a)	*b)*	*c)*	*d)*
p-value	$7.78 \cdot 10^{-35}$	$5.83 \cdot 10^{-56}$	$6.27 \cdot 10^{-33}$	$5.80 \cdot 10^{-47}$

The overall best performing algorithm is OLPSO, on which idea is OLFA mainly based. The classical versions of PSO and FA have almost the same results (mostly for higher dimension size). The FFPSO, the ancestor of OLFA, has the same results as OLFA for

dimension size 10 and slightly better for dimension size 30. This finding is interesting because in OLFA the firefly does not move towards other fireflies like classical FA, but in direction estimated by the guidance vector created using orthogonal learning. FFPSO, on the other hand, using the PSO method of having access to *pBest* and *gBest* memories. The idea behind OLFA lies somewhere between these two principles.

6 Conclusion

The primary aim of this original work is to provide a more in-depth insight into the relations between control parameters adjustments, learning techniques, inner swarm dynamics and possible hybridization strategies for FA based on the PSO. Both algorithms are popular and have already many applications in several real-world tasks. Specifically, the focus here is to experimentally investigate the influence of modern technique, which is orthogonal learning, to the performance of the hybrid FFPSO algorithm. Thus novel proposed algorithm orthogonal learning firefly algorithm – OLFA is thoroughly tested here.

Our proposed algorithm has been tested on CEC 2017 benchmark suite. The results were statistically evaluated and compared with other algorithms using the Friedman rank test. The algorithms in comparisons are canonical FA, FFPSO, PSO, and OLPSO. This paper presents two different scenarios in all comparisons. Firstly, the control parameters of OLFA were initially set to values recommended for its ancestors, and secondly, an extensive tuning of OLFA parameters have been carried out to improve the overall performance of the OLFA algorithm. Thus OLFA had become competitive against its hybridized predecessor FFPSO.

The analyzed data suggest that the orthogonal learning technique performs better with the PSO algorithm than applied to FFPSO. This may be affected by the nature of FA behavior, which seems to be less suitable for implementation of orthogonal learning. However, the proposed approach can be further improved by better parameter settings, e.g., refreshing gap G, where performed tests lend weight to the theory, that this parameter may be somewhat adaptive and related to the dimensionality of the optimization problem being solved.

Despite the fact, that the ongoing research has brought many powerful and robust metaheuristic algorithms, the researchers have to deal with a well-known phenomenon so-called *no free lunch theorem* [36] forcing them to test various methods, techniques, adaptations and parameter settings leading to the acceptable results.

The presented results indicate that the original FFPSO hybrid algorithm offers a noticeable potential to many improvements. Even though the results do not show any significant performance improvement, it is necessary to emphasize the fact that, like most of the evolutionary/swarm-based algorithms, they are inspired by natural evolution, and their development can be considered as a form of evolution. Such a fact is mentioned in the article [37] that even incremental steps in algorithm development, including failures, may be the inspiration for the development of robust and powerful metaheuristics.

Finally, extensive research including even more optimal parameter tuning is required for increasing the understandability of the algorithm behavior and its performance.

Also, a design of the proposed algorithm offers several possible changes that could affect its performance. Majority of them are based on the improvements of the “donor" algorithms FA and PSO, for example, Lévy flights, adaptive control parameters, comprehensive learning and more.

Acknowledgments. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2020/001. This work is also based upon support by COST (European Cooperation in Science \& Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

References

1. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. 1995.
2. Shi Y, Eberhart R C. Parameter selection in particle swarm optimization. In: International conference on evolutionary programming. Springer, Berlin, Heidelberg, 1998. p. 591-600.
3. Cleghorn, CH. W., and Engelbrecht A., "Particle swarm optimizer: The impact of unstable particles on performance." Computational Intelligence (SSCI), 2016 IEEE Symposium Series on. IEEE, 2016.
4. Shi Y, Eberhart R C. Empirical study of particle swarm optimization. In: Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on. IEEE, 1999. p. 1945-1950.
5. Hu X, Eberhart R C. Adaptive particle swarm optimization: detection and response to dynamic systems. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. IEEE, 2002. p. 1666-1670.
6. Pluhacek M, et al. PSO with partial population restart based on complex network analysis. In: International Conference on Hybrid Artificial Intelligence Systems. Springer, Cham, 2017. p. 183-192.
7. Juang C. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004, 34.2: 997-1006.
8. Esmin A AA, Lambert-torres G, De Souza AC Z. A hybrid particle swarm optimization applied to loss power minimization. IEEE Transactions on power systems, 2005, 20.2: 859-866.
9. Kadavy T, et al. Multi-swarm Optimization Algorithm Based on Firefly and Particle Swarm Optimization Techniques. In: International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2018. p. 405-416.
10. Helwig S, Branke J, Mostaghim S. Experimental analysis of bound handling techniques in particle swarm optimization. IEEE Transactions on Evolutionary computation, 2013, 17.2: 259-271.

11. Kadavy T, et al. Comparing strategies for search space boundaries violation in PSO. In: International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2017. p. 655-664.
12. Zhan, Z. H., Zhang, J., Li, Y., & Shi, Y. H. (2011). Orthogonal learning particle swarm optimization. IEEE transactions on evolutionary computation, 15(6), 832-847.
13. Yang X. Nature-inspired metaheuristic algorithms. Luniver press; 2010.
14. Oosumi R, Tamura K, Yasuda K. Novel single-objective optimization problem and firefly algorithm-based optimization method. In: Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on. IEEE, 2016. p. 001011-001015.
15. Yang XS. Multiobjective firefly algorithm for continuous optimization. Engineering with Computers, 2013, 29.2: 175-184.
16. Yang XS. Firefly algorithm, Levy flights and global optimization. In: Research and development in intelligent systems XXVI. Springer, London, 2010. p. 209-218.
17. dos santos Coelho L, Mariani V C. Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Computers & Mathematics with Applications, 2012, 64.8: 2371-2382.
18. Fister, I., Fister Jr, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, *13*, 34-46.
19. Beltran, A., Hughes, C., & Das, S. (2019, June). Improved Maximum Power Point Tracking of Partially Shaded PV Arrays Using Particle Swarm Optimization with Zone Initialization. In 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC) (pp. 0663-0667). IEEE.
20. Zaman M A, et al. Nonuniformly spaced linear antenna array design using firefly algorithm. International Journal of Microwave Science and Technology, 2012, 2012.
21. Yousif A, Abdullah A H, Nor S M. Scheduling jobs on grid computing using firefly algorithm. 2011.
22. Jati G K, et al. Evolutionary discrete firefly algorithm for travelling salesman problem. In: Adaptive and intelligent systems. Springer, Berlin, Heidelberg, 2011. p. 393-403.
23. Cleghorn, Christopher W., and Andries P. Engelbrecht. "Firefly optimization: A study on frame invariance." Computational Intelligence (SSCI), 2017 IEEE Symposium Series on. IEEE, 2017.
24. Kora P, Rama Krishna KS. Hybrid firefly and Particle Swarm Optimization algorithm for the detection of Bundle Branch Block. International Journal of the Cardiovascular Academy. 2016 March 1,2(1):44-8.
25. Fister Jr, I., Mlakar, U., Brest, J., & Fister, I. (2016). A new population-based nature-inspired algorithm every month: is the current era coming to the end. In Proceedings of the 3rd Student Computer Science Research Conference (pp. 33-37). University of Primorska Press.
26. Du W, Li B. Multi-strategy ensemble particle swarm optimization for dynamic optimization. Information Sciences. 2008 August 1,178(15):3096-109.
27. Wang H, Wu Z, Rahnamayan S, Sun H, Liu Y, Pan J. Multi-strategy ensemble artificial bee colony algorithm. Information Sciences. 2014 September 20,279:587-603.
28. Shelokar, P. S., Siarry, P., Jayaraman, V. K., & Kulkarni, B. D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied mathematics and computation*, *188*(1), 129-142.
29. Das, S., Abraham, A., & Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In *Advances of computational intelligence in industrial systems* (pp. 1-38). Springer, Berlin, Heidelberg.

30. Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 3, pp. 1951-1957). IEEE.
31. Shi, Y., & Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on* (Vol. 1, pp. 101-106). IEEE.
32. Zhan, Z. H., Zhang, J., Li, Y., & Shi, Y. H. (2011). Orthogonal learning particle swarm optimization. IEEE transactions on evolutionary computation, 15(6), 832-847.
33. N. H. Awad, et al. Problem Definitions and Evaluation Criteria for CEC 2017 Special Session and Competition on Single-Objective Real-Parameter Numerical Optimization, 2016.
34. Demsar J. Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research, 2006, 7.Jan: 1-30.
35. J. Kennedy, "The particle swarm: social adaptation of knowledge," in Proceedings of the IEEE International Conference on Evolutionary Computation, 1997, pp. 303–308.
36. Droste, S., Jansen, T., & Wegener, I. (1999, July). Perhaps not a free lunch but at least a free appetizer. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1 (pp. 833-839). Morgan Kaufmann Publishers Inc.
37. Piotrowski, A. P., & Napiorkowski, J. J. (2018). Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?. Swarm and Evolutionary Computation.
38. Zhou, J., Nekouie, A., Arslan, C. A., Pham, B. T., & Hasanipanah, M. (2019). Novel approach for forecasting the blast-induced AOp using a hybrid fuzzy system and firefly algorithm. Engineering with Computers, 1-10.
39. Hassan, M. K., El Desouky, A. I., Badawy, M. M., Sarhan, A. M., Elhoseny, M., & Gunasekaran, M. (2019). EoT-driven hybrid ambient assisted living framework with naïve Bayes–firefly algorithm. Neural Computing and Applications, 31(5), 1275-1300.
40. Altabeeb, A. M., Mohsen, A. M., & Ghallab, A. (2019). An improved hybrid firefly algorithm for capacitated vehicle routing problem. Applied Soft Computing, 84, 105728.
41. Tharwat, A., Elhoseny, M., Hassanien, A. E., Gabel, T., & Kumar, A. (2019). Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. Cluster Computing, 22(2), 4745-4766.
42. Wang, K., Liu, J., Kong, S., Cai, P., Xu, H., & Wan, S. (2019). Quantitative Recognition of Granary Heat Source Using Quantum-Behaved Particle Swarm Optimization Method. Journal of Thermophysics and Heat Transfer, 1-6.
43. Tomas, K., Michal, P., Adam, V., & Roman, S. (2018, June). Orthogonal Learning Firefly Algorithm. In International Conference on Hybrid Artificial Intelligence Systems (pp. 315-326). Springer, Cham.