EAI Endorsed Transactions
on Energy Web

Research Article EAI.EU

A new heuristic with a multi-threaded implementation of a modified Firefly Algorithm

Alfonso Murillo-Suarez, Felix Martinez-Rios*

Universidad Panamericana, Facultad de Ingeniería, Ciudad de México, México

Abstract

In this article, we present a modified version of the Firefly Algorithm implemented in a multi-threaded model to improve the results obtained by the original algorithm significantly. This multi-threaded algorithm allows the threads to obtain different results by the independent execution of the heuristic method in each of them, although for keeping all the threads with significant executions, the algorithm performs some crossover techniques, explained in detail in this article, for the threads to learn between them while maintaining its independence. For testing the new algorithm, we use the six benchmark functions used in the literature for testing the original Firefly Algorithm, and to prove that the improved results are significant, we perform the Wilcoxon test to the results obtained. The results obtained with this new heuristic proved to be significantly better while taking advantage of today's commercial processors.

Received on 29 February 2020; accepted on 06 April 2020; published on 15 April 2020

Keywords: Firefly Algorithm, heuristics, optimization, exploration, exploitation, multi-threading

doi:10.4108/eai.13-7-2018.163984

1. Introduction

As optimization becomes more complex, the need for algorithms capable of delivering reliable results has increased. What makes optimization problems harder to solve is the number of variables (dimensions) involved in the formulation of the problem and that the values for these variables are englobed in a continuous domain [1].

Traditional methods and algorithms fail when trying to find the optimal values for specific functions, or have really long execution times. When optimization is so complex that the optimal values cannot be computed by regular algorithms in a considerable amount of time, heuristics are used for trying to approach the best solution [2]. Although heuristic methods are able to get a very reliable approach, they do not get the exact values of the parameters for getting the optimum. Researchers have exhaustively developed new heuristic algorithms that improve the results and execution times.

A widely developed category of heuristic methods is nature-inspired algorithms [3, 4]. For these algorithms, researchers try to formulate algorithms that simulate natural processes, especially from biology, physics, and chemistry. The major number of nature-inspired algorithms are bio-inspired due to the successful characteristics of these behaviors. Nature-inspired algorithms are usually classified as Swarm-Intelligent (SI) based, bio-inspired not SI based, and physics and chemistry-based [5]. Figure 1 shows examples of algorithms within these classifications.

Swarm-Intelligent based algorithms are inspired in the collective behavior of insects and animals that, despite individually they are not good on finding objectives, the whole swarm has an organization to achieve their objective. These algorithms are the most used because of the ability of agents to learn of their own experience and between them through iterations.

Bio-inspired algorithms contain SI algorithms as a subset since not all bio-inspired algorithms use the swarm collective behavior. These kinds of algorithms are based on biological individual behaviors like flowers pollination or genetic processes.

Authors have also inspired from physics or chemistry laws. As these phenomena also is a product of nature, they are englobed in the nature-inspired algorithms, but they are not based on biological processes. Some

*Corresponding author. Email: alfonso.murillosuarez@up.edu.mx

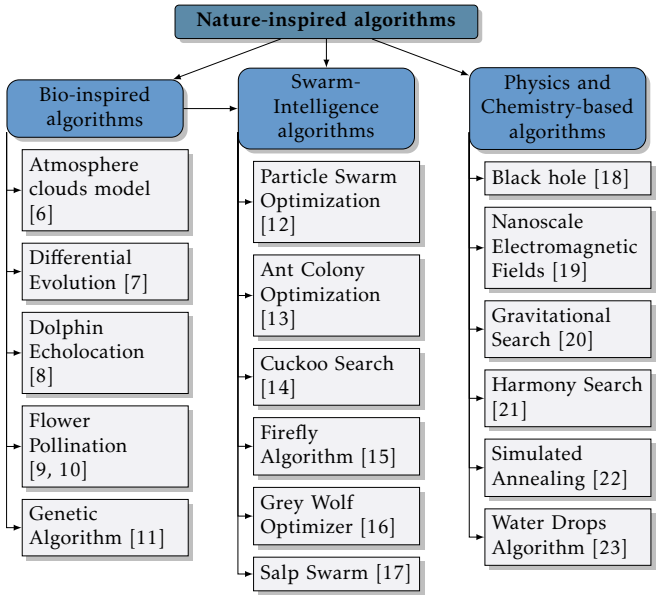


Figure 1. Nature-inspired algorithms classification and examples

sources of inspiration for this category of algorithms are electromagnetic fields, gravity processes, etc.

2. Firefly Algorithm

2.1. Considerations for the Firefly Algorithm

The Firefly Algorithm, presented by Yang in 2009 [15], is a Swarm-Intelligence based algorithm that uses the fireflies' characteristic flashing as inspiration. This flashing helps the fireflies to find mating partners (communication) and to attract prey. As the characteristic of SI algorithms is the cluster's communication and interaction, the first purpose of the flashing is used. In some species, fireflies get attracted to each other thanks to the flashing patterns of male fireflies; for other species of fireflies, this is used as a technique to confuse male fireflies (they believe there is a potential mate) and eat them. In any of these variants, the flashing clearly represents a form of organization between the members of the swarm.

The light intensity from the fireflies' flashings I is less visible as the distance r between the observer and the light source increases, obeying the inverse square law: $I \propto 1/r^2$. Additional to this, light from the environment also decreases the flashing perception.

The distance and the light absorption factors are considered while implementing the algorithm but also, for simplifying the description of the algorithm, three rules are set:

1. Fireflies are unisex, so they can be attracted between them regardless of their sex.

2. The attractiveness of a firefly is proportional to their brightness, so between two fireflies, the less bright will move towards the brighter one. This also implies that the farther a firefly is from another, the less attractive it is.

3. The brightness of a firefly is determined by the objective function. For minimization problems (as used in this paper) the brightness or intensity of the firefly can be represented as $I = -f(x)$, so that the minimum of the objective function is the highest intensity.

Algorithm 1: Pseudocode of the Firefly Algorithm

```
Data: N, α, β0, γ, Max_Iterations
Result: Best solution
Generate initial population of fireflies
  xi(i = 1, 2, ..., N)
Evalute the objective function for all N fireflies
  by f(xi)
Formulate light intensity as I = −f(xi)
while t < Max_Iterations do
    for i = 1 to N do
        for j = 1 to N do
            if Ij > Ii then
                Move firefly i towards firefly j
                  according to Equation 4
            end
        end
        Evaluate new solutions and update light
          intensity
    end
    Rank the fireflies and update the best solution
      found so far
    t ← t + 1
end
Return the best solution
```

Considering the above rules, the Firefly Algorithm is formulated as shown in Algorithm 1.

The input parameters for Algorithm 1 are the N number of fireflies that will make the swarm, the randomization parameter α, the attractiveness β_0 when the distance r from the source is 0, the environment light absorption coefficient γ, and the maximum number of iterations of the algorithm.

2.2. Mathematical formulation of the Firefly Algorithm

The attractiveness of a firefly and the light intensity are two very important factors for the algorithm. As we said before, to simplify the process the attractiveness of the firefly is related to its brightness, which is determined by the fitness value of the firefly's position.

For simplicity, we determine the brightness (or light intensity) as $I(x) \propto -f(x)$, but the attractiveness β is

more complicated since it is relative to the observant distance from the firefly (distance r_{ij} between fireflies i and j). As the light intensity $I(r)$ varies according to the inverse square law we can say that $I(r) = I_s/r^2$, where I_s is the light intensity at the source. But distance is not the only factor affecting the attractiveness, we said before that the light is lost in the environment with a degree of absorption, given by the absorption coefficient γ. The effect of both factors can be expressed with the following Gaussian form:

$$I(r) = I_0 e^{-\gamma r^2} \tag{1}$$

Remembering that the attractiveness β of a firefly is related to how other fireflies perceive the light intensity, the attractiveness of a firefly i perceived by a firefly j can be formulated as follows:

$$\beta(r) = \beta_0 e^{-\gamma r_{ij}^2}, \tag{2}$$

where β_0 is the attractiveness of firefly i at $r = 0$ and r_{ij} is the distance between both fireflies. This distance is calculated as the Cartesian distance:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} \left(x_i^k - x_j^k\right)^2}, \tag{3}$$

where x_i^k is the $k-th$ component of $i-th$ firefly's position x_i.

If a firefly i finds an attractive (more brightening) firefly j, the movement of firefly i towards firefly j is defined as follows:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha\left(rand - \frac{1}{2}\right), \tag{4}$$

where the constant β_0 is usually set to 1 and the random factor $\alpha \in (0, 1)$.

3. Multi-threaded Firefly Algorithm

Nowadays, parallel architectures are accessible at low costs and the implementation of parallel processes has been made easier with today's tools. This has made researchers develop multi-threaded algorithms to solve different problems [24].

The main purpose of the multi-threading implementations is to improve the execution times by running a process simultaneously in various processors instead of using a single one [25].

3.1. Modified Firefly Algorithm

Our work is based on the original Firefly Algorithm but we made some modifications to improve the results even if it is not executed in the multi-threaded implementation. Especially for improving the abilities of exploration and exploitation of the algorithm.

When we say exploration, we mean that the algorithm is capable of searching variously, with flexibility and taking risks through the objective function's search space, so that no subspace is unexplored. Exploitation consists of, once having a potential subspace, refine the results [26].

To achieve this, we propose a modification of the equation movement in which the randomization coefficient is smaller for the first fireflies in the swarm to give them the possibility to exploit and increases towards the last ones for them to explore. This is made because the fireflies are sorted after each iteration, so the first fireflies are the ones with better results, and the final ones have the worst results so, with a bigger randomization coefficient they can keep exploring to improve their results. With this consideration, our proposed movement equation is as follows:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha^{N+1-i}\left(rand - \frac{1}{2}\right), \tag{5}$$

where the randomization coefficient for each firefly is modified according to the number of fireflies in the swarm (N) and its position in the ordered cluster (i).

At the beginning of the algorithm, the fireflies have random positions so there is no guarantee that the first ones have good results, so we use the original movement equation (Equation 4) at the first iterations and our new movement equation (Equation 5) for the last ones. We use a percentage of iterations (POI) to determine how many iterations should be executed before we start using the new movement equation.

In addition to the improvements before, we allow the fireflies to explore new solutions but we only accept them if they improve, so that all fireflies have temporal values (which are the ones changing during the iteration) and actual values, that are only changed if the temporal ones got better results.

The pseudocode in Algorithm 2 details this modified version of the Firefly Algorithm.

3.2. Multi-threaded implementation

The modified version of the Firefly Algorithm (Algorithm 2) is the one to be executed simultaneously by the threads, but for taking more advantage of this multi-threaded implementation, we use some crossover techniques that allow the independent executions of the algorithm to communicate between them periodically so that they can learn from each other.

The crossover techniques are executed after a determined number of iterations (CNI) is reached by each of the independent executions. All the processes are paused when they reach the CNI iterations and their results are compared and modified according to the selected crossover technique. After completing

Algorithm 2: Pseudocode of the modified Firefly Algorithm

Data: N, α, β_0, γ, Max_Iterations, POI
Result: Best solution
Generate initial population of fireflies $x_i (i = 1, 2, ..., N)$
Evalute the objective function for all N fireflies by $f(x_i)$
Formulate light intensity as $I_i = -f(x_i)$
while $t < Max\_Iterations$ **do**
  Copy the parameters of all fireflies x_i to a temporal swarm $temp\_x_i$
  Copy the intensity of all fireflies I_i to a temporal array $temp\_I_i$
  for $i = 1$ *to* N **do**
    for $j = 1$ *to* N **do**
      if $temp\_I_j > yrmp\_I_i$ **then**
        if $t <$ POI% of Max_Iterations **then**
          Move temporal firefly i towards temporal firefly j according to Equation 4
        end
        else
          Move temporal firefly i towards temporal firefly j according to Equation 5
        end
      end
    end
    Evaluate temporal solutions and update temporal light intensity
    foreach *Firefly i in* x_i **do**
      if $temp\_x_i$ *is better than* x_i **then**
        $x_i \leftarrow temp\_x_i$
        $I_i \leftarrow temp\_I_i$
      end
    end
  end
  Rank the fireflies and update the best solution found so far
  $t \leftarrow t + 1$
end
Return the best solution

the modifications of the crossover, the independent executions are restarted for another CNI iterations, after which the crossover technique is executed again. This process repeats until de maximum number of iterations is reached.

We developed two crossover techniques: *Crossover with the K best threads* and *Crossover with Simulated Annealing with the K best threads*.

- **Crossover with the K best threads.** This crossover technique consists of sorting the threads according to their best fitness obtained for the objective function and selecting the k_best number of best threads. For each of the remaining threads, a random k_best_i thread is selected to copy its values for the parameters, so when the thread restarts the execution of the algorithm it starts with the same values for the parameters as the k_best_i thread assigned.

- **Crossover with Simulated Annealing with the K best threads.** This technique also sorts the threads and selects the k_best ones but uses the Simulated Annealing process [22] to determine if a thread is going to receive the parameters of a random thread from the k_best ones or not. In order to determine this, the following expression is calculated as presented in [27]:

$$g(t) = e^{\left\{\frac{-1}{log\left[\frac{Max\_Iterations}{t}\right]}\right\}}, \tag{6}$$

  where t is the current iterations and $Max\_Iterations$ is the maximum number of iterations for the algorithm.

  A random number in the interval $[0, 1)$ is generated for each of the threads out of the set of k_best threads. The thread is going to receive the parameters of the assigned k_best_i thread only if $g(t) < random[0, 1)$.

Once explained this, we can present how the Multi-threaded implementation of a modified Firefly Algorithm works. The pseudocode in Algorithm 3 describes the process.

Algorithm 3 receives the following parameters:

- NT**:** the number of threads that will execute the algorithm simultaneously.

- CNI**:** the number of iterations that each of the individual processes will perform before pausing for the implementation of the crossover technique.

- k_best**:** the number of threads that will be selected to be used to copy their parameters to the other threads.

- N**:** the number of fireflies that will make up the swarm of each individual process.

- α, β_0, γ**:** the randomization factor, the attractiveness at the source, and the light absorption coefficient used for the Firefly Algorithm (Algorithm 2).

Algorithm 3: Pseudocode of the Multi-threaded implementation of a modified Firefly Algorithm

Data: NT, CNI, k_best, N, α, β_0, γ, Max_Iterations, POI
Result: S^{best}

Initialize the Υ set with NT threads
Initialize S^{best}
foreach τ *in* Υ **do**
  Initialize S_τ^{best}
end
$t \leftarrow 0$
while $t < Max\_Iterations$ **do**
  foreach τ *in* Υ **do**
    Execute the modified Firefly Algorithm as shown in Algorithm 2 until CNI iterations are reached
    $S_\tau^{best} \leftarrow$ the best solution obtained so far by τ
    if S_{tau}^{best} *is better than* S^{best} **then**
      $S^{best} \leftarrow S_{tau}^{best}$
    end
  end
  Sort the threads in Υ according to S_τ^{best}
  Select the k_best threads in Υ
  foreach τ *in* Υ *and* $\tau \notin k\_best$ **do**
    Perform the selected crossover technique
  end
  $t \leftarrow t + CNI$
end
Return S^{best}

- $Max\_Iterations$: the maximum number of iterations that all the processes will perform.
- POI: the percentage of iterations that will be executed before the new movement equation (Equation 5) is used as shown in Algorithm 2.

4. Experiments

For testing our algorithm we based on the results published by Hashmi et al. in [28].

4.1. Benchmark functions

We used the six benchmark functions shown in Table 1 for testing our algorithm since are the ones used in [28] so that we can compare the results obtained. Table 1 has all de details about the functions (number of function, name, dimensions, and the mathematical equation).

4.2. Experimental setup

We evaluated the six benchmark functions with different combinations of the number of threads NT and the k_best parameters, and varying the number of fireflies N as in [28]. The remaining parameters were set as:

- $CNI = 2000$
- $\alpha = 0.2$
- $\beta = 1.0$
- $\gamma = 0.2$
- $Max\_Iterations = 10000$
- $POI = 50\%$

We executed the algorithm 20 times for being able to obtain an average result, this average result is compared with the best values obtained by [28]. Hashmi et al. vary the number of fireflies for their experiment so we did the same, but we also vary the NT and k_best

Table 1. Benchmark functions

$f1$	Sphere	
D=10	Range=$[-100, 100]$	$f(x)_{min} = 0$
$f(x) = \sum_{i=1}^{d} x_i^2$		
$f2$	Sum Square	
D=10	Range=$[-10, 10]$	$f(x)_{min} = 0$
$f(x) = \sum_{i=1}^{d} i x_i^2$		
$f3$	Step 2	
D=10	Range=$[-100, 100]$	$f(x)_{min} = 0$
$f(x) = \sum_{i=1}^{d} \lvert x_i + 0.5\rvert^2$		
$f4$	Trid 10	
D=10	Range=$[-10, 10]$	$f(x)_{min} = -210^*$
$f(x) = \sum_{i=1}^{d} (x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}$		
$f5$	Zakharov	
D=10	Range=$[-0.5, 10]$	$f(x)_{min} = 0$
$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^4$		
$f6$	Rosenbrock	
D=10	Range=$[-30, 30]$	$f(x)_{min} = 0$
$f(x) = \sum_{i=1}^{d} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$		

** The minimum of the Trid 10 function is obtained by:* $f(x)_{min} = (-d(d+4)(d-1))/6$

Table 2. Results: 10 fireflies and Crossover with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	1.23E-06	1.43E-15	1.56E-15	1.41E-15	1.52E-15	1.55E-15	**1.26E-15**	1.52E-15	1.42E-15	1.31E-15
2	3.74E-06	1.17E-14	1.13E-14	**9.19E-15**	1.34E-14	1.16E-14	1.10E-14	1.07E-14	1.16E-14	1.04E-14
3	4.56E-07	1.62E-15	1.50E-15	1.38E-15	1.59E-15	1.57E-15	1.49E-15	1.36E-15	**1.24E-15**	1.43E-15
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	4.19E-15	4.13E-15	3.33E-15	4.57E-15	**3.29E-15**	3.90E-15	4.04E-15	4.00E-15	3.93E-15
6	-9.94E+03	5.35E-01	5.48E-01	1.94E-01	1.54E+01	**1.73E-01**	2.59E-01	6.32E-01	1.34E+01	1.75E-01

Table 3. Results: 10 fireflies and Crossover with Simulated Annealing with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	1.23E-06	1.45E-15	**1.32E-15**	1.34E-15	1.47E-15	1.41E-15	1.50E-15	1.49E-15	1.58E-15	1.58E-15
2	3.74E-06	1.07E-14	1.04E-14	**8.97E-15**	1.08E-14	9.64E-15	9.61E-15	1.06E-14	1.12E-14	9.38E-15
3	4.56E-07	1.67E-15	1.53E-15	1.51E-15	1.57E-15	**1.43E-15**	1.47E-15	1.61E-15	1.55E-15	1.53E-15
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	4.14E-15	3.77E-15	**3.20E-15**	4.09E-15	3.42E-15	3.73E-15	4.07E-15	4.46E-15	3.78E-15
6	-9.94E+03	2.13E+01	1.69E+01	4.13E-01	1.33E+01	1.92E-01	2.04E-01	1.09E+01	5.56E-01	**1.73E-01**

parameters. We finished with six different experiments to test the objective functions:

1. 10 fireflies using the crossover with the K best technique (Table 2)
2. 10 fireflies using the crossover with Simulated Annealing with the K best technique (Table 3)
3. 20 fireflies using the crossover with the K best technique (Table 4)
4. 20 fireflies using the crossover with Simulated Annealing with the K best technique (Table 5)
5. 40 fireflies using the crossover with the K best technique (Table 6)
6. 40 fireflies using the crossover with Simulated Annealing with the K best technique (Table 7)

Each of these experiments varies the number of threads NT in 7, 8, and 10; and the k_best value in 3 and 4.

The execution of the experiments was made in a Dell Precision M6800 Laptop with Intel Core i7-4900MQ (Quad Core 2.80GHz, 3.8GHz Turbo, 8MB) with 32 GB in RAM.

4.3. Results of the experiments

In this subsection, we show the results obtained for the different experiments (Tables 2, 3, 4, 5, 6, and 7). The tables headings indicate the amount of threads used and the value for the k_best parameter, for example, the heading 8T-3K indicates that de number of threads $NT = 8$ and the value of K threads $k\_best = 3$.

The best results obtained for each function are marked in bold. We applied the Wilcoxon test [29] to validate that our results were significantly better than the ones obtained by the original algorithm.

Table 4. Results: 20 fireflies and Crossover with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	6.14E-07	1.69E-29	2.08E-29	1.83E-29	2.04E-29	**1.46E-29**	1.73E-29	1.69E-29	1.72E-29	1.68E-29
2	5.51E-06	5.98E-27	**1.74E-27**	3.55E-27	7.07E-27	6.86E-27	3.21E-27	7.26E-27	4.31E-27	2.97E-27
3	4.67E-07	1.84E-29	1.81E-29	**1.65E-29**	1.85E-29	1.78E-29	1.77E-29	1.80E-29	1.71E-29	**1.65E-29**
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	1.38E-26	4.16E-27	5.52E-27	9.72E-27	1.28E-26	4.70E-27	7.72E-27	7.26E-27	**3.19E-27**
6	-9.96E+03	6.13E+00	1.12E+01	3.88E+01	2.76E+01	1.62E+01	2.19E+01	**6.11E-01**	2.70E+00	9.58E-01

Table 5. Results: 20 fireflies and Crossover with Simulated Annealing with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	6.14E-07	1.89E-29	1.79E-29	1.64E-29	1.92E-29	1.90E-29	1.83E-29	2.11E-29	**1.52E-29**	1.62E-29
2	5.51E-06	1.16E-26	3.11E-27	2.99E-27	5.67E-27	2.07E-27	4.53E-27	1.20E-26	**1.88E-27**	6.95E-27
3	4.67E-07	1.72E-29	1.77E-29	1.51E-29	1.78E-29	1.45E-29	1.54E-29	1.71E-29	1.73E-29	**1.44E-29**
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	3.97E-27	5.43E-27	6.37E-27	6.21E-27	5.58E-27	8.16E-27	**2.75E-27**	6.24E-27	5.18E-27
6	-9.96E+03	2.29E+01	1.35E+01	2.02E+01	3.99E+01	2.36E+01	2.55E+01	1.66E+01	2.19E+01	**6.70E-01**

Table 6. Results: 40 fireflies and Crossover with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	5.46E-07	7.84E-57	6.61E-57	1.15E-56	1.23E-56	8.86E-57	6.52E-57	8.89E-57	6.86E-57	**6.46E-57**
2	3.83E-06	7.06E-34	4.70E-34	4.63E-35	9.93E-33	7.76E-35	8.47E-36	**5.50E-36**	2.04E-35	2.94E-35
3	4.22E-07	9.86E-33	6.16E-33	**2.47E-33**	7.40E-33	1.02E-32	8.63E-33	8.63E-33	9.86E-33	7.40E-33
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	1.03E-35	1.49E-34	**4.33E-36**	4.21E-35	4.59E-35	5.88E-35	8.69E-35	1.21E-34	8.77E-35
6	-9.99E+03	4.60E+01	6.60E+01	5.41E+01	7.99E+01	7.43E+01	5.02E+01	**3.66E+00**	8.84E+01	4.10E+01

Table 7. Results: 40 fireflies and Crossover with Simulated Annealing with the K best threads

f	FFA	7T-2K	8T-2K	10T-2K	7T-3K	8T-3K	10T-3K	7T-4K	8T-4K	10T-4K
1	5.46E-07	8.23E-57	6.75E-57	6.95E-57	9.26E-57	8.01E-57	7.06E-57	**6.36E-57**	7.66E-57	8.24E-57
2	3.83E-06	6.46E-34	8.14E-34	9.60E-34	1.75E-34	4.88E-35	4.40E-35	6.92E-35	**4.91E-36**	2.20E-35
3	4.22E-07	7.40E-33	9.86E-33	7.40E-33	1.60E-32	1.05E-32	8.63E-33	7.40E-33	1.11E-32	**3.70E-33**
4	-2.10E+02	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**	**-2.10E+02**
5	-9.00E-02	2.14E-35	5.05E-35	**8.50E-36**	5.85E-35	9.32E-35	2.37E-35	8.44E-35	2.79E-35	2.17E-35
6	-9.99E+03	5.37E+01	6.13E+01	6.49E+01	8.48E+01	1.89E+01	5.29E+01	6.51E+01	3.36E+01	**1.26E+01**

4.4. Results analysis

To show the behavior of our algorithm we prepared Figure 2 which shows the convergence curve of the algorithm using 10 fireflies, 10 threads and the k_best parameter set as 2. The vertical axis shows the fitness of the objective function while the horizontal axis represents the iterations (from 1 to 10000).

The algorithm converges fastly to small values, this is why in further works we will propose modifications to start exploiting the search space when the algorithm converges, although this can be managed with the POI parameter. Tests varying this parameter are also contemplated for further work.

We also made some trajectory curves of the firefly that obtained the best result in 500 iterations with a CNI of 100 and the remaining parameters stayed the same. The red dot shows the last value obtained (the best value).

Table 3 shows the trajectory of the best firefly with the crossover to the best K technique while Table 4 uses the crossover with annealing with the best K technique. In these graphs, we are able to see sudden changes in the position that are consequences of the crossover between the threads.

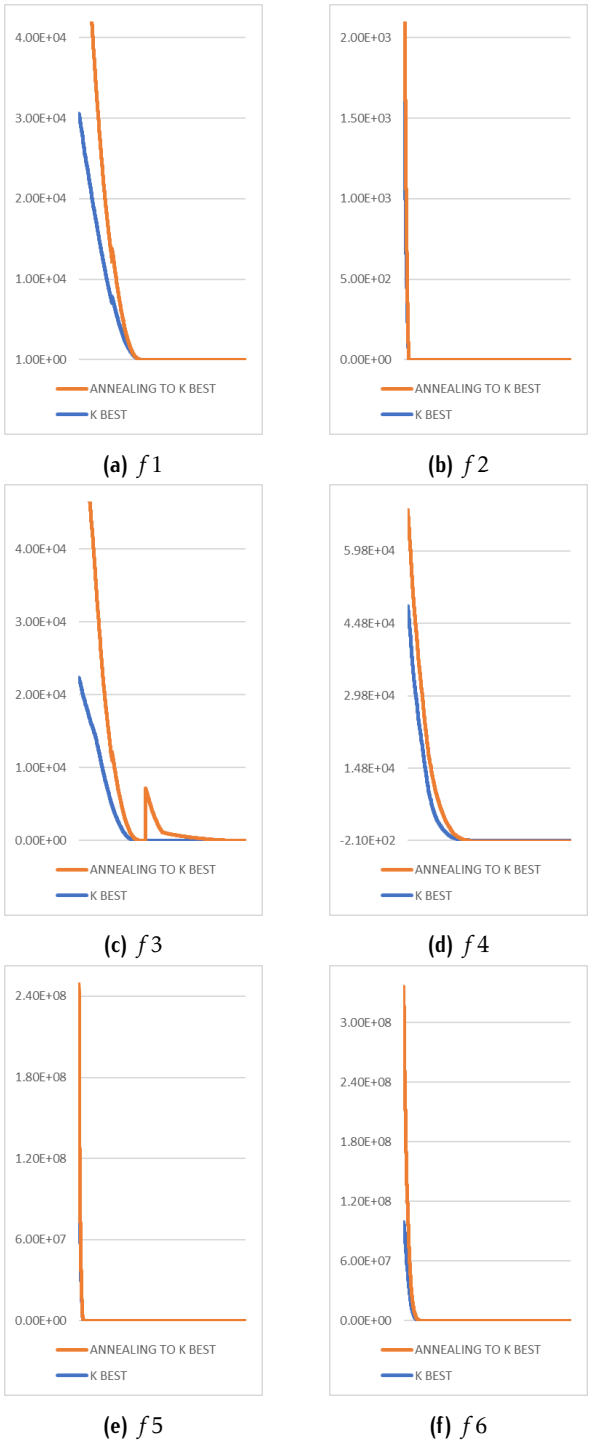


Figure 2. Convergence curves of the benchmark functions

4.5. Conclusions

Our algorithm obtained very significantly better results than the original implementation of the Firefly Algorithm. The tables with the results show that for every function, no matter the crossover technique nor the NT or k_best parameters, our algorithm obtained the best results.

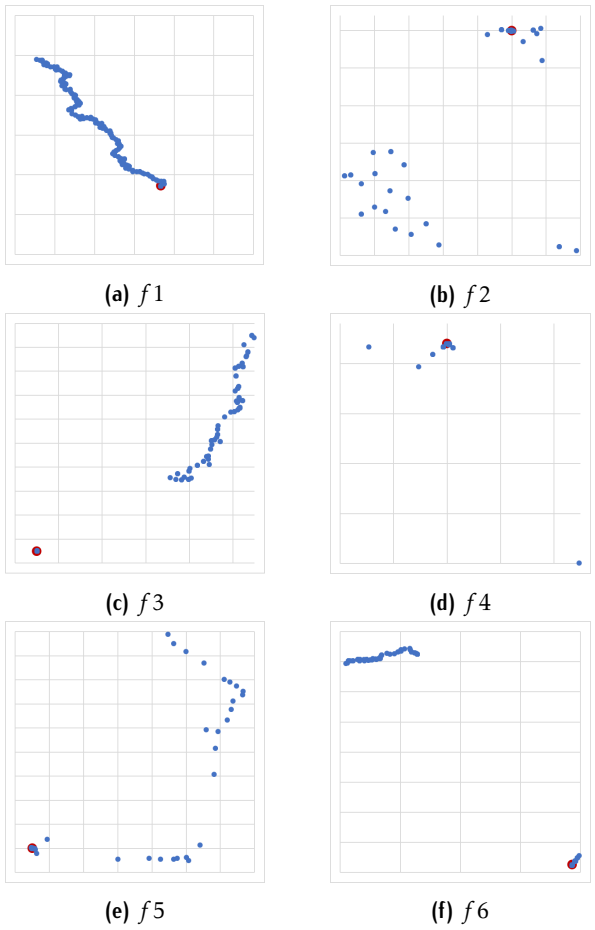


Figure 3. Trajectory of the best firefly with crossover with the best K

As we mentioned in the analysis section, the algorithm seems to converge fast to optimal values, so we seek to perform tests varying the POI parameter to start exploiting the search space in a better moment.

References

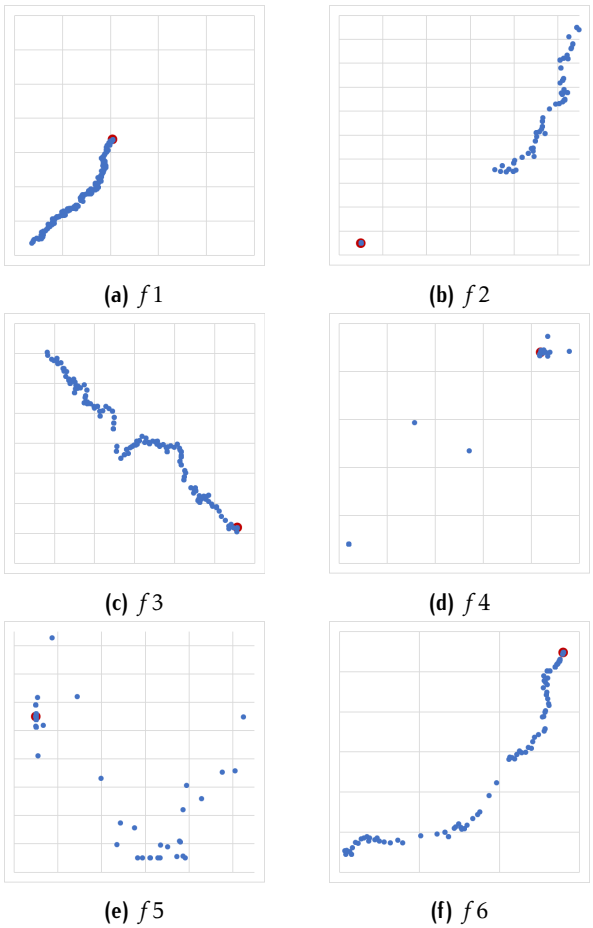
[1] Wang, H., Wu, Z. and Rahnamayan, S. (2011) Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Comput.* **15**: 2127–2140. doi:10.1007/s00500-010-0642-7.

[2] Gigerenzer, G., Hertwig, R. and Pachur, T. (2011) *Heuristics: The Foundations of Adaptive Behavior*. doi:10.1093/acprof:oso/9780199744282.001.0001.

[3] Yang, X.S. (2010) *Nature-inspired metaheuristic algorithms* (Luniver press).

[4] Ayawli, B., Chellali, R., Appiah, A. and Kyeremeh, F. (2018) An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning. *Journal of Advanced Transportation* **2018**: 1–27. doi:10.1155/2018/8269698.

[5] Fister jr, I., Yang, X.S., Fister, I., Brest, J. and Fister, D. (2013) A brief review of nature-inspired algorithms



(a) $f1$ (b) $f2$ (c) $f3$ (d) $f4$ (e) $f5$ (f) $f6$

Figure 4. Trajectory of the best firefly with crossover with annealing with the best K

for optimization. *Elektrotehniski Vestnik/Electrotechnical Review* **80**.

[6] Yan, G.W. and Hao, Z.J. (2013) A novel optimization algorithm based on atmosphere clouds model. *International Journal of Computational Intelligence and Applications* **12**(01): 1350002.

[7] Storn, R. and Price, K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **11**: 341–359. doi:10.1023/A:1008202821328.

[8] Kaveh, A. and Farhoudi, N. (2013) A new optimization method: Dolphin echolocation. *Advances in Engineering Software* **59**: 53 – 70. doi:https://doi.org/10.1016/j.advengsoft.2013.03.004, URL http://www.sciencedirect.com/science/article/pii/S0965997813000318.

[9] Yang, X.S. (2012) Flower pollination algorithm for global optimization. In Durand-Lose, J. and Jonoska, N. [eds.] *Unconventional Computation and Natural Computation* (Berlin, Heidelberg: Springer Berlin Heidelberg): 240–249.

[10] Yang, X.S., Karamanoglu, M. and Xingshi, H. (2013) Multi-objective flower algorithm for optimization. *Procedia Computer Science* **18**: 861–868. doi:10.1016/j.procs.2013.05.251.

[11] Reeves, C.R. (1995) A genetic algorithm for flowshop sequencing. *Computers and Operations Research* **22**(1): 5–13.

[12] Eberhart, R. and Kennedy, J. (1995) A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*: 39–43.

[13] Dorigo, M. (1992) *Optimization, Learning and Natural Algorithms (in Italian)*. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

[14] Yang, X.S. and Deb, S. (2010) Cuckoo search via levy flights .

[15] Yang, X.S. (2009) Firefly algorithms for multimodal optimization. In Watanabe, O. and Zeugmann, T. [eds.] *Stochastic Algorithms: Foundations and Applications* (Berlin, Heidelberg: Springer Berlin Heidelberg): 169–178.

[16] Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) Grey wolf optimizer. *Advances in Engineering Software* **69**: 46 – 61. doi:https://doi.org/10.1016/j.advengsoft.2013.12.007, URL http://www.sciencedirect.com/science/article/pii/S0965997813001853.

[17] Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H. and Mirjalili, S.M. (2017) Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software* **114**: 163 – 191. doi:https://doi.org/10.1016/j.advengsoft.2017.07.002, URL http://www.sciencedirect.com/science/article/pii/S0965997816307736.

[18] Hatamlou, A. (2013) Black hole: A new heuristic optimization approach for data clustering. *Information Sciences* **222**: 175 – 184.

[19] Martinez, F., Marmolejo, J. and Murillo-Suarez, A. (2018) A new heuristic algorithm to solve circle packing problem inspired by nanoscale electromagnetic fields and gravitational effects: 1–6. doi:10.1109/NANOFIM.2018.8688621.

[20] Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. (2009) Gsa: A gravitational search algorithm. *Information Sciences* **179**(13): 2232 – 2248. Special Section on High Order Fuzzy Sets.

[21] Geem, Z.W., Kim, J. and Loganathan, G. (2001) A new heuristic optimization algorithm: Harmony search. *Simulation* **76**: 60–68. doi:10.1177/003754970107600201.

[22] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by simulated annealing. *Science* **(4598)**(220): 671–680.

[23] Shah-Hosseini, H. (2009) The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation* **1**(1-2): 71–79. Cited By 177.

[24] Manfrin, M., Birattari, M. and Stützle, T. (2006) Parallel ant colony optimization for the traveling salesman problem. *LNCS 4150: Proc of the 5th Int Workshop on Ant Colony Optimization and Swarm Intelligence* : 221–234.

[25] JáJá, J. (1992) An introduction to parallel algorithms.

[26] March, J.G. (1991) Exploration and exploitation in organizational learning. *Organization science* **2**(1): 71–87.

[27] Martinez-Rios, F. and Murillo-Suarez, A. (2018) A new swarm algorithm for global optimization of multimodal functions over multi-threading architecture hybridized with simulating annealing. *Procedia Computer Science* **135**: 449 – 456. doi:https://doi.org/10.1016/j.procs.2018.08.196, URL http://www.sciencedirect.com/science/article/pii/S1877050918314868. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

[28] Hashmi Adil Hashmi, A. (2013) Firefly algorithm for unconstrained optimization. *IOSR Journal of Computer Engineering* **11**: 75–78. doi:10.9790/0661-1117578.

[29] Rey, D. and Neuhäuser, M. (2011) *Wilcoxon-Signed-Rank Test* (Berlin, Heidelberg: Springer Berlin Heidelberg), 1658–1659. doi:10.1007/978-3-642-04898-2_616, URL https://doi.org/10.1007/978-3-642-04898-2_616.