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.

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

Copyright © 2020 Alfonso Murillo-Suarez et al., licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/), which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.

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

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
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. 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 another, the less attractive it is).

2.2. Mathematical formulation of the Firefly Algorithm

The attractiveness 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.

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 \( \alpha \), the attractiveness \( \beta_0 \) when the distance \( r \) from the source is 0, the environment light absorption coefficient \( \gamma \), 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 \( \beta \) 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 $I(r) = I_0/r^2$, where $I_0$ 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 $\gamma$. The effect of both factors can be expressed with the following Gaussian form:

$$I(r) = I_0e^{-\gamma r^2} \tag{1}$$

Remembering that the attractiveness $\beta$ 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:

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

where $\beta_0$ 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:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{D} (x_{ik} - x_{jk})^2}, \tag{3}$$

where $x_{ik}^k$ 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:

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

where the constant $\beta_0$ 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:

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

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, β₀, γ, Max_Iterations, POI
Result: Best solution

1. Generate initial population of fireflies \( x_i \) \( (i = 1, 2, \ldots, N) \)
2. Evaluate the objective function for all N fireflies by \( f(x_i) \)
3. Formulate light intensity as \( I_i = -f(x_i) \)
4. While \( t < \text{Max}_\text{Iterations} \) do
   5. Copy the parameters of all fireflies \( x_i \) to a temporal swarm \( \text{temp}_x \)
   6. Copy the intensity of all fireflies \( I_i \) to a temporal array \( \text{temp}_I \)
   7. For \( i = 1 \) to \( N \) do
      8. For \( j = 1 \) to \( N \) do
         9. If \( \text{temp}_I_j > \gamma \text{temp}_I_i \) then
            10. If \( t < \text{POI} \% \) of \( \text{Max}_\text{Iterations} \) then
                11. Move temporal firefly \( i \) towards
                    12. Temporal firefly \( j \) according to
                        13. Equation 4
            14. Else
                15. Move temporal firefly \( i \) towards
                    16. Temporal firefly \( j \) according to
                        17. Equation 5
       18. End
    19. Evaluate temporal solutions and update
        20. Temporal light intensity
    21. Foreach Firefly \( i \) in \( x_i \) do
        22. If \( \text{temp}_x_i \) is better than \( x_i \) then
            23. \( x_i \leftarrow \text{temp}_x_i \)
            24. \( I_i \leftarrow \text{temp}_I_i \)
        25. End
    26. Rank the fireflies and update the best solution
        27. Found so far
    28. \( t \leftarrow t + 1 \)
29. Return the best solution

- 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 \( k_{\text{best}} \) number of best threads. For each of the remaining threads, a random \( k_{\text{best}} \) 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 \( k_{\text{best}} \) thread assigned.

- Crossover with Simulated Annealing with the K best threads. This technique also sorts the threads and selects the \( k_{\text{best}} \) 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 \( k_{\text{best}} \) ones or not. In order to determine this, the following expression is calculated as presented in [27]:

\[
g(t) = e^{\frac{-t}{\text{Max}_\text{Iterations}}}, \quad (6)
\]

where \( t \) is the current iterations and \( \text{Max}_\text{Iterations} \) 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 \( k_{\text{best}} \) threads. The thread is going to receive the parameters of the assigned \( k_{\text{best}} \) thread only if \( g(t) < \text{random}[0, 1] \).

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.
- \( k_{\text{best}} \): 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.
- \( \alpha, \beta_0, \gamma \): the randomization factor, the attractiveness at the source, and the light absorption coefficient used for the Firefly Algorithm (Algorithm 2).

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 the 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.

A. Murillo-Suarez, F. Martinez-Rios

EAI Endorsed Transactions on
Energy Web
Online First
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
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 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 parameters, and varying the number of threads N as in [28]. The remaining parameters were set as:

- **CNI** = 2000
- **α** = 0.2
- **β** = 1.0
- **γ** = 0.2
- **Max_Iterations** = 10000
- **POI** = 50%
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.

### 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 $CN1$ of 100 and the remaining parameters stayed the same. The red dot shows the last value obtained (the best value).

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$e$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
<th>$7T$</th>
<th>$8T$</th>
<th>$9T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
<td>1.00E+00</td>
</tr>
</tbody>
</table>
A new heuristic with a multi-threaded implementation of a modified Firefly Algorithm

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.

References


A. Murillo-Suarez, F. Martinez-Rios

Figure 4. Trajectory of the best firefly with crossover with annealing with the best K for optimization. Elektrotehniški Vestnik/Electrotechnical Review 80.


doi:10.1016/j.procs.2013.05.251.


