Gradus Vol 8, No 1 (2021) 173-183
ISSN 2064-8014

SURVEY ON FIVE NATURE-INSPIRED OPTIMIZATION ALGORITHMS

Ahmad Reda [1] and Zsolt Csaba Johanyák [2]*

[1] Department of Automation and Info-communication, GEIK, Faculty of Mechanical Engineering and Informatics, University of Miskolc, Hungary
[2] Department of Information Technology, John von Neumann University, Kecskemét Hungary
https://doi.org/10.47833/2021.1.CSC.001

Keywords:
Particle Swarm Optimization
Firework Algorithm
Clonal Selection
Firefly Algorithm
Cuckoo Search

Article history:

Received	10 January 2021
Revised	28 April 2021
Accepted	1 June 2021

Abstract
This paper presents a literature review about Particle Swarm Optimization (PSO), Firework, Firefly, Clonal Selection, and Cuckoo Search algorithms, which are among the most common natural-inspired optimization algorithms. These algorithms were tried on different benchmark functions. The obtained results were analyzed, and the performance was compared. The results showed that PSO and Firefly Search algorithms provided the best performance in the studied cases.

1 Introduction

In the last decades, the complexity of the real-world problems has significantly increased and the use of the traditional algorithms became ineffective. In this context, the nature-inspired computational methodologies have attracted the attention and interest of the researchers in different areas due to their simplicity, leverage and efficiency. Many algorithms have been developed and presented such as Particle Swarm Optimization, Firework, Firefly algorithms, etc. This type of algorithm has been used to address a variety of optimization problems where the traditional algorithms don't provide satisfactory performance.

This paper was organized as follows: after a brief introduction in the first section, the Particle Swarm Optimization algorithm was presented in the second section. Firework, Clonal, Firefly and Cuckoo Search were presented in sections three, four, five and six respectively. In section seven, the experiments and the obtained results are discussed. The conclusions of the paper are presented in section eight.

2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization is a metaheuristic global optimization algorithm proposed by Kennedy and Eberhart in 1995 [12]. It is robust stochastic optimization technique based on the movements of and intelligence of swarms. PSO was inspired by social behavior of the organism's groups where the information can be share between the members. Swarms of n particles (individual agents) communicate with one another using search directions (gradients) and each particle is a candidate solution of the optimization problem. PSO performs searching in the space of the fitness function or objective function $f(x_i)$ using a swarm of particles that updates from iteration to iteration. The goal is to find the global best among all the current best solution. In other words, the goal is to find the best solution among the possible solutions in the search space [1] [15] [21].

Each particle can be characterized by its position vector $X_i^t = (x_i(1), x_i(2), .., x_i(n))^T$ and its velocity vector $V_i^t = (v_i(1), v_i(2), .., v_i(n))^T$, here i denotes the number of the particle, while n is the number of the dimensions of the search space. The velocity vector contains the gradient (direction)

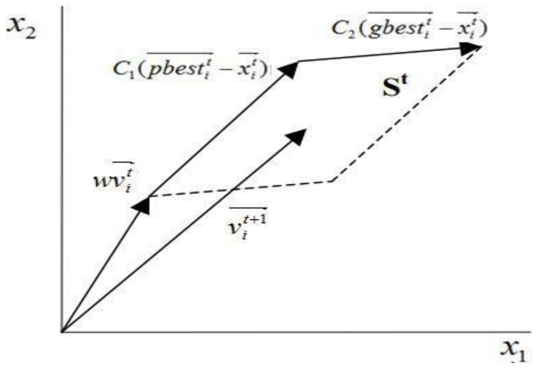
* Corresponding author.
E-mail address: autareda@uni-miskolc.hu

of which particles travel, so it describes the movements of the particle in the sense of direction. These vectors are updated in each iteration based on equations (1) and (2).

$$\mathrm{V_i^{t+1}} = \mathrm{wV_i^t} + \mathrm{c_1 r_1^t}\left(\mathrm{pbest_i} - \mathrm{X_i^t}\right) + \mathrm{c_2 r_2^t}\left(\mathrm{gbest_i} - \mathrm{X_i^t}\right) \tag{1}$$

$$\mathrm{X_i^{t+1}} = \mathrm{X_i^t} + \mathrm{V_i^{t+1}} \tag{2}$$

where: c_1, c_2 are self-confidence coefficient and social coefficient, respectively; r_1, r_2 are random numbers uniformly distributed in the [0, 1] interval, and w is the inertial weight, $\mathrm{pbest_i}$ denotes the best position of the ith particle so far (personal best), while $\mathrm{gbest_i}$ stands for the overall best position identified so far (global best). Figure 1 and Figure 2 show the update of the particle's velocity and its position at t^{th} iteration [10].



x_2
$C_2(\overline{gbest_i^t - x_i^t})$
$C_1(\overline{pbest_i^t - x_i^t})$
S^t
$\overline{wv_i^t}$
$\overline{v_i^{t+1}}$
x_1

Figure 1. The velocity vector

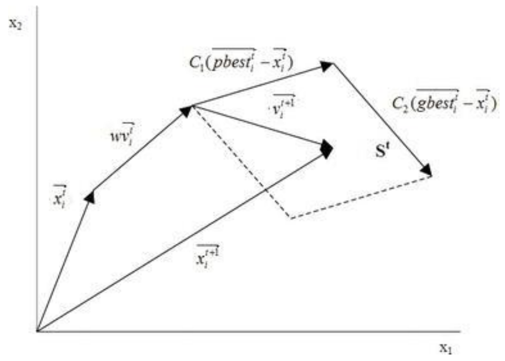


Figure 2. The updated velocity vector at (t) iteration

The PSO algorithm's flowchart can be represented as follow [10]:

```
1. Initialization: For each particle $i$ in a swarm population size P:
   1.1 Initialize $X_i$ randomly.
   1.2 Initialize $V_i$ randomly.
   1.3 Initialize the current $best_i$
   1.4 Initialize the global pest $gbest$.
2. Repeat until stopping criterion is satisfied
   2.1 For particle $i$
       2.1.1 Update $V_i^t$ and $X_i^t$ according to the equations 1 and 2.
       2.1.2 Evaluate the objective function $f\ (X_i)$ at the new position.
       2.1.3 $pbest_i \leftarrow X_i^t$ If $f\ (pbest_i) > f\ (X_i^t)$.
       2.1.4 $gbest \leftarrow X_i^t$ If $f\ (gbest\ ) > f\ (X_i^t)$.
```

3 Firework Optimization Algorithm

In the last years, Swarm Intelligence (SI) Algorithms such as Swarm Optimization (PSO) and Clonal Selection Algorithm have gained a significant attention and become very popular optimization algorithms among the researchers [4]. A new Swarm Intelligence algorithm called Firework Algorithm was developed by Tan and Zhu. It was inspired by the emergent swarm behavior of fireworks and it uses the stochastic search technique. The search technique used in Firework algorithm bases on creating explosion around specific points, and the sparks of the explosion is considered as a local search space around the point. The success of the Firework algorithms mainly based on two factors, the selected locations and the good design of the explosions [8].

Based on the nature of the firework explosions, two main behaviors can be observed, the number and distribution areas of the generated sparks. In a good firework explosion, it can be noticed that the number of the sparks is quite large, and they are located in the explosion center [17].

By comparing the two types of firework explosion, it is clear that the more generated sparks with less amplitude the best firework we get. Meaning that, for Firework algorithm it is better to use more sparks to search in the local areas. The firework algorithms can be described as follows.

1. *Select n locations ($\{x_i \mid i = 1..n\}$).*
2. *Evaluate each location (yi=f(xi)).*
3. *Calculate the number of sparks for each location (s_i).*
4. *Calculate the amplitude of each explosion (A_i).*
5. *Calculate the number of dimensions in which sparks will move away from the original location (z).*
6. *For each location and for each spark calculate the displacement (new position) of the spark.*
7. *Evaluate each spark.*
8. *Keep the best spark location and select n-1 locations from the positions of the sparks. These will serve as the locations for the fireworks in the next cycle.*
9. *Go to step 2.*

By considering the following general optimization problem:

$$\text{Minimize } f(x) \in \mathbb{R}, x_{\min} \leq x \leq x_{\max} \tag{3}$$

where $f(x)$ an objective function is $x = x1, x2, \ldots\ldots, xd$ is a location in the potential space, and ($x_{\min}, x_{\max}$) are the bounds of the space, the number of the generated sparks for each explosion x_i can be determined as follows:

$$s_i = m.\frac{y_{\max} - f(x_i) + \xi}{\sum_{i=1}^{n}(y_{\max} - f(x_i)) + \xi} \tag{4}$$

where m is a parameter used to control the total number of the generated sparks, $y_{\max}$ is the maximum (worst) value of the objective function among the n fireworks, where $y_{\max} = \max(f(x_i)), (i = 1, 2, \ldots., n)$, and (ξ) is a very small constant to avoid zero division error. The number of spark is limited by lower and upper bounds as follows

$$s_i = \max\left(\|a \cdot m\|, \min\left(\|b \cdot m\|, \|s_i\|\right)\right), a < b < 1, \tag{5}$$

where a and b are constant parameters of the method, and denotes a rounding operation. The Amplitude of explosion for each firework can be defined as it is shown in equation 6, where Â is the maximum explosion amplitude, $y_{min} = \min\,(f(x_i)), (i = 1, 2, \ldots., n)$ is the minimum of the objective function among current set of locations.

$$A_i = \hat{A}.\frac{f(x_i) - y_{min} + \xi}{\sum_{i=1}^{n}(f(x_i) - y_{min}) + \xi} \tag{6}$$

The number of affected directions of the explosion is obtained as follows $z = \left\| N \cdot random(0,1) \right\|$, where $random(0,1)$ is uniformly distributed random number over [0,1], N is the dimensionality of the location (x).

The final (after the firework explosion) position (Step 6) of the spark j is determined by calculating a displacement from the location of the firework (xi) by applying the following calculation steps.

a. Randomly select z dimensions

$$D_j = \{d_k \mid 1 \le d_k \le N, k = 1..z\}, \tag{7}$$

b. Calculate the displacement in a dimension

$$h_j = A_i \cdot random(-1,1), \tag{8}$$

c. For each selected dimension move the spark from the original firework location by applying the calculated displacement.

$$x_{d_k}^j = x_{i,d_k} + h_j,\ k = 1..z \tag{9}$$

d. Apply eventual lower and upper bounds for the coordinates.

$$x_{d_k}^j = \max\left(x_{\min,\,d_k},\ \min\left(x_{\max,\,d_k},\ x_{d_k}^j\right)\right), \tag{10}$$

where $x_{\min,\,d_k}$ and $x_{\max,\,d_k}$ are the lower and the upper bounds in the d_k dimension.

In course of the preparation for the next iteration (Step 8) the location with the best object function value is always kept. The remaining n-1 locations for the fireworks of the next iteration are selected from a collection of positions (K) that contains all the firework and spark locations of the current iteration excluding the previously mentioned best location. In course of the selection the objective function values of the candidates are not taken into consideration, only their distances from other locations are used. Here the key idea is to ensure a diversity of the locations. The selection process can be described by the following steps:

a. For all candidates determine their so-called general distance from other locations

$$R(x_i) = \sum_{x_j \in K} d(x_i, x_j), \tag{11}$$

where K is the set of current locations and d(.) is an arbitrary distance measure.

b. Assign a selection probability to all candidate locations

$$P(x_i) = \frac{R(x_i)}{\sum_{J \in \mathrm{k}} R(x_i)} \tag{12}$$

c. Sort the candidate locations in descending order by their selection probability.

d. Select the first n-1 locations from the list.

4 Clonal Selection Based Optimization

In 2002, De Castro and Von Zuben were inspired by the biological and artificial immunology and they proposed clonal selection algorithm. In immunology, each unique molecule (antigen) is detected and recognized by an antibody, where the antibody is a protein that is used by the immune system to detect and neutralize the foreign objects. In this context, the Artificial Immune System algorithms imitate the principles of immune systems. In the recent years, the clonal selection theory has gained great attention from scholars and inspired them to provide algorithms which based on creating different candidate solutions through cloning, selection and mutation procedures. Clonal Selection Based Algorithm is used to solve several types of problems including problems of multimodal optimization based on clonal selection principle [2] [5]. Comparing to the heuristics algorithms such as neural networks and genetic algorithms, the clonal selection was reported to provide a better performance in different applications such as pattern recognition [11] [18]. The Steps of the Clonal Selection Based algorithm by De Castro and Von Zuben is mainly proposed for the pattern recognition applications which are as follows:

1. Initialize all relevant parameters (antibody size, number of iterations..., etc.), chose the antigen randomly and create the set of the candidate antibodies.
2. For each candidate antibody, the affinity between the antigen and the antibody is calculated and the first (n) highest affinity antibodies are selected.
3. The selected (n) antibodies are cloned and the number (n) is related positively with the affinity factor.
4. A treatment of mutation on the antibodies that are generated after cloning is performed, taking into consideration that the probability of mutation is decreased with the increase of the affinity factor.
5. After the mutation, the affinity of the antibody is calculated, and the antibody highest affinity antibody is selected.
6. The selected antibody in the previous step is compared with the one in the current memory set and the highest affinity antibody is selected to be moved into the memory set.
7. The worst (d) remaining antibodies (have the worst affinity to the remaining antigen) are replaced with (d) new randomly selected antibodies
8. Next iteration is performed starting from the step 2, and the algorithm is terminated when the termination condition is achieved (maximum number of iterations).

From the algorithm steps it can be noted that the algorithm chose the first n antibodies that have the highest affinity of the candidate antibodies in step2. The other steps are about cloning and mutation processes of these antibodies [5]. The main problem types that are studied and solved by clone selection algorithm [18] are in general function optimization, Pattern Recognition (PR), scheduling, Industrial Engineering (IE) related problems and others such as classification, machine learning and time series prediction.

5 Firefly Algorithm

Firefly Algorithm (FA) is metaheuristic algorithms and one of the swarm-based algorithms that was developed by Yang [25]. FA bases on the behavior and flashing patterns of the fireflies. Fireflies communicate with each other by means of flashing patterns which are unique in most cases. In this context and in other words, the Firefly Algorithm emulate the way that the fireflies are communicate using their flashing lights. Two main factors have the essential role, the light intensity and attractiveness. For simplicity, the fact of that the attractiveness is determined by the brightness can be stated. Metaheuristic Algorithms based on approximation solutions to deal with the optimization problem which it may be good for some applications [22]. The Algorithms taking in account the following facts:

- All fireflies are attracted to each other (unisex) regardless their sex.
- The brightness determines the attractiveness of a firefly meaning that attractiveness is proportional to the brightness
- The brighter fireflies attract the less bright ones, and it moves randomly in the case of no difference in brightness.
- Brightness is depending objective function.
- Light intensity is affected by distance and light observation coefficient of the medium that it passes through.

Generally speaking, and based on the facts above, the Firefly Algorithm can be descried with the following steps:

Set algorithms parameters (α: step length of the random movement, γ light absorption)
Set Max Generation: maximum number of iterations
Objective function $f(x)$, $x = (x_1, x_2 \dots . x_d)^T$

```
Light intensity $I_i$ at xi is determined by $f(x_i)$
Generate initial populations of $n$ firlyies.
For iteration =1: MaxGeneration
  Calculate the brightness, $I$
  Sort the solutions in such away : $I_i \geq I_{i-1}$
  For $i = 1 : n - 1$
          For $j = i + 1 : n$
                  If $I_j \geq I_{i-}$
                          Move firefly $i$ towords firefly j
                  End if
          End for
  End for
Move firefly randomly.
End for
Report the best solution.
```

As we mentioned earlier, the brightness (light intensity) is directly related to the distance, where the brightness decreases as the distance increases [6]. Light intensity is subject to inverse square law as it is shown equation 13, where$(\mathrm{I_s})$: is the source light intensity and $(\mathrm{I_r})$ is the light intensity at distance (r).

$$\mathrm{I_r} < \frac{\mathrm{I_s}}{\mathrm{r}^2} \tag{13}$$

The light intensity can be described by equation 14 when it passes through medium light absorption coefficient (γ), Where $(\mathrm{I_o})$ is the light intensity at distance$\mathrm{r} = 0$.

$$I = I_o e^{-\gamma \mathrm{r}} \tag{14}$$

The attractiveness β of the firefly can be described based on the light intensity of the neighboring firefly as equation 15 shows, where β_o is the attractiveness at is $\mathrm{r} = 0$.

$$\beta = \beta_o e^{-\gamma \mathrm{r}^2} \tag{15}$$

The generalized function of light intensity for $(\mathrm{m} > 1)$ is described by equation 16. It is worth to point out that any other monotonically decreasing function can be used.

$$\beta_r = \beta_o e^{-\gamma r^m} \tag{16}$$

Cartesian distance is used to determine the distance between two fireflies $\mathrm{x_i}$ and $\mathrm{x_j}$ as equation 17 shows, where $\mathrm{x_{i,k}}$ is the part of the spatial coordinate $\mathrm{x_i}$.

$$\mathrm{r_{ij}} = \|\mathrm{x_i} - \mathrm{x_j}\| = \sqrt[2]{\sum_{\mathrm{k}=1}^{\mathrm{d}} (\mathrm{x_{i,k}} - \mathrm{x_{j,k}})^2} \tag{17}$$

For two fireflies, in the case of firefly at j is brighter than firefly at i, firefly i will be attracted to firefly j and will move in its direction using the formula described in (*18*), where the second term of the equation represents the attraction, and the third term represents the randomization where (α) is an algorithm parameter describes the length of the step for the random movement and $(\in)$ is a random vector from uniform distribution in range [0,1].

$$\mathrm{x_i^{t+1}} = \mathrm{x_i^t} + \beta_o e^{-\gamma \mathrm{r}^2}(\mathrm{x_j^t} - \mathrm{x_i^t}) + \alpha \mathrm{x} \in_\mathrm{i}^\mathrm{t} \tag{18}$$

Generally speaking, the distance (r) is not only about Euclidean distance, it may represent any essential measure based on the optimization problem itself. For example (r) can be defined as time interval. Firefly Algorithm is suitable for parallel computing since different FA algorithms can work almost independently [24]. The attraction mechanism inspired other researchers to develop other algorithms based on this concept, for example Charged System search which uses Coulomb's law and search algorithm which uses Newton's law of gravitation.

6 Cuckoo Search Algorithm

Cuckoo Search (CS) is metaheuristic algorithm which inspired by some cuckoo species laying their eggs in the nest of individuals of other species birds that increase their survival and productivity. It was developed by Xin-She Yang and Suash Deb in 2009 [23]. Consequently, in this algorithm there are two different species of birds which are the Cuckoo birds and the host birds. The Cuckoo eggs can be discovered by the host bird and in this case, there are two possibilities:

1. The host bird will throw the egg away.
2. The host bird will abandon the nest and build a new one.

The probability of discovering the alien eggs by the host bird (P_a) can be determined as$P_a \in (0,1)$, and for the simplicity of the implementations, three main rules are suggested:

1. One egg can be laid by the cuckoo bird at a time and it is placed in a randomly selected nest. Mathematically speaking: each egg represents a solution and it is stored in nest:
   Cuckoo egg = New solution
   Eggs in nest = Set of solutions
2. The nest that will be passed to the next generation is the one with the higher quality of eggs. Mathematically speaking:
   High quality egg = Best solution near optimal value. Meaning that the eggs which are more similar to the eggs of the host bird have more opportunity to survive and to a new mature cuckoo.
3. The number of available host nest is defined and fixed. The probability of discovering the cuckoo eggs by the host bird is $P_a \in (0,1)$ and the discovered egg can be thrown away or the host bird leaves the nest and builds a new nest. Mathematically speaking:
   Number of host nest = Fixed (i.e., Population).
   Host bird discovers cuckoo eggs = Worse solution.

The new solution of the cuckoo search algorithm can be determined based on equation 19 which is stochastic equation for global random walk (based on Lévy flights) [19], [3], Where x_i^t is the current location,$i = (1, 2, 3, ..., n)$, a is step size, λ is levy exponential and $\oplus$ is entry wise multiplication.

$$x_i^{t+1} = x_i^t + a \oplus Levy(\lambda) \tag{19}$$

The Cuckoo Search algorithm is a simulation of Random Search Walk Process. On the other hand, CS algorithm uses a combination of the global and local random walk which can be described based on equation 20, Where x_i^t and x_k^t are represents two solutions that are selected randomly, H is Heaviside function, ε is a random number and s is a step size.

$$x_i^{t+1} = x_i^t + as \oplus H(P_a - \varepsilon) \oplus (x_J^t - x_K^t) \tag{20}$$

The flow chart of the Cuckoo Search algorithm can be described as (*figure 3*) shows [13]:

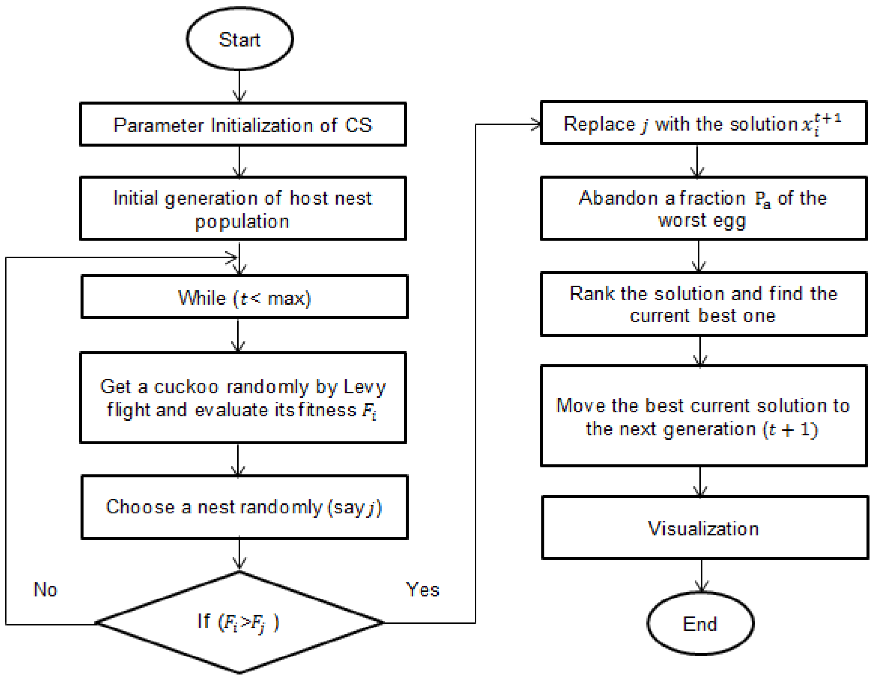

Start
Parameter Initialization of CS
Initial generation of host nest population
While (t< max)
Get a cuckoo randomly by Levy flight and evaluate its fitness F_i
Choose a nest randomly (say j)
If (F_i>F_j)
No
Yes
Replace j with the solution x_i^{t+1}
Abandon a fraction P_a of the worst egg
Rank the solution and find the current best one
Move the best current solution to the next generation ($t+1$)
Visualization
End

Figure 3. Workflow of Cuckoo search algorithm

The parameters of the CS algorithm need to be chosen, where the probability P_a and the population size n are the essential parameters as we can set the Levy exponent λ and the step size (a) as a constants. The parameters can be chosen based on the following recommendations which are sufficient for most of the optimization problems: $\lambda = 1.5, a = 0.01, n = 15$ to 40 and $P_a = 0.25$. The implementations of the Cuckoo Search algorithm in a variety of optimization problems have shown a promising performance [7], [26]. In addition to the simplicity of implementations of the Cuckoo Search algorithm, it has several advantages such as the ability to deal with multi-criteria optimizations problems and it can be still hybridized with other swarm-based algorithms.

7 Experiments and Results

The performance of the investigated PSO, Firework, Firefly, Clonal Selection, and Cuckoo Search optimization algorithms were compared by conducting experiments on five benchmark functions. The used benchmark functions, their global minimum values, the corresponding x vectors as well as the lower and upper boundaries of the search space are presented in Table 1. Here d denotes the number of dimensions of the search space.

Table1. Benchmark Functions

Function Name	*Formula*	$x_{i\,min}$	$x_{i\,max}$	x^*	$f(x^*)$
Griewank Function	$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	-100	100	(0,0,…,0)	0
Rastrigin Function	$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$	-5.12	5.12	(0,0,…,0)	0
Schwefel Function	$f(x) = \left(\sum_{i=1}^{d} x_i^2\right)^{10}$	-10	10	(0,0,…,0)	0

Function Name	*Formula*	$x_{i\,min}$	$x_{i\,max}$	x^*	$f(x^*)$
Rosenbrock Function	$f(x) = \sum_{i=1}^{d-1}[(1-x_i)^2 + 100(x_{i+1} - x^2)^2\,]$	-5	5	(1,1,…,1)	0
Sphere Function	$f(x) = \sum_{i=1}^{D} x_i^2$	-5	5	(0,0,…,0)	0

The obtained results are shown in Table 2. The initial parameters of the optimization algorithms were set as follows. The starting point was determined randomly and the number of dimensions was 10. The random number generator was initialized using the same seed, which ensured the repeatability of the experiments.

In case of Particle Swarm Optimization, the population size = 50, self-confidence coefficient=0.2, social coefficient =0.2, inertial weight coefficient = 0.6, maximum number of allowed iterations=100.

In case of Firework Optimization Algorithm, the number of locations = 50, the constant influencing the number of sparks=40, constant for the maximum value of the explosion amplitude=10, constant for the lower bound of number of sparks = 0.5, constant for the upper bound of sparks = 0.9, and constant for the minimum value of the explosion amplitude = 1.

In case of Firefly optimization algorithm, the population size (number of fireflies) = 50, randomness strength coefficient = 1, attractiveness constant = 1, absorption coefficient = 0.01, randomness reduction factor = 0.97.

In case of Clonal Selection based optimization the number of antibodies in the repertoire = 50, number of antibodies selected for hypermutation in the first generation = 13, the initial hypermutation rate = 0.5, the number of antibodies selected for cloning = 14, the coefficient that determines the number of clones that are created for each selected antibody = 0.4, hypermutation coefficient = 0.8, number of random antibodies created at the end of each iteration cycle = 15.

In case of Cuckoo Search Algorithm population size = 50, discovery rate of alien eggs = 0.25.

Table 2. Minimum/average cost function values obtained over runs

Evaluation Function	*Best Cost/Average Cost*				
	PSO	*Firework*	*Firefly*	*Clonal Selection*	*Cuckoo Search*
Griewank	0.1588/0.4123	1.3655/2.9063	2.2482/3.4903	1.3784/1.5336	0.6261/07781
Rastrigin	6.6862/17.6243	30.6396/59.6454	11.2885/19.5941	44.1714/62.7618	23.8862/32.2878
Schwefel	0.00003/33.1325	0.0001/0.0110	1e-6/1.7982e-5	694.5651/565940	0.2398/13.6949
Rosenbrock	16.8345/85.6232	52.9212/182.5048	9.7122/14.8007	839.9228/1761.3	39.2572/76.9222
Sphere	0.0661/0.5381	0.3149/0.8866	0.0427/0.0821	6.1113/9.2779	0.1763/0.3336

The goal of the search algorithms was to find the minima of the benchmark functions. Therefore, the performance of the algorithms was measured by the actual lowest function value they found after the allowed number of generations/iterations. In case of each algorithm the search process was carried out 20 times and two performance indicators were taken into considerations, i.e. the result of the best run (the lowest function value found) and the average of the lowest function values found at the end of the individual runs. The results are presented in Table 2. By taking into consideration the fact that some algorithms provide better performance based on the problem itself, the obtained results show that in case of the Griewank and the Rastrigin functions PSO provided the best results regarding both performance indicators, while in case of the Schwefel, Rosenbrock and Sphere functions the Firefly algorithm ensured the best results.

8 Conclusions

In this paper, an overview of five Nature-Inspired Optimization Algorithms (PSO, Firework, Firefly, Clonal Selection, and Cuckoo Search) was presented. The performance of these algorithms was compared based on conducting several experiments on five benchmark functions. The experiments showed that PSO and Firefly algorithms provided better performance compared to the other algorithms, meaning that both are considered as a promising algorithm to be used with a variety of real-life applications. On the other hand, it is worth to point out that the results and the accuracy of the algorithms are directly affected by the chosen values for their parameters. Thus, this study provides a good reference in order to choose the suitable parameters, but generally speaking, choosing the values of the parameters is still considered as a challenge which will be taken in consideration with more details in the future works.

Further research planes include the application and comparison of the different optimization algorithms in case of fuzzy models (e.g. [9] [14] [20]).

Acknowledgment

This research is supported by EFOP-3.6.1-16-2016-00006 "The development and enhancement of the research potential at John von Neumann University" project. The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

References

[1] Brownlee, J. : Clever Algorithms: Nature-Inspired Programming Recipes, 2011, 1st edn, pp. 29–86, ISBN 978-1-4467-8506-5
[2] Brownlee. J. : Clonal Selection Algorithms, CIS Technical Report 070209A, Complex Intelligent Systems Laboratory, Swinburne University of Technology, Melbourne, Australia, 2007.
[3] Chechkin. A., Metzler. R., Klafter. J., Conchar. V. : Introduction to the Theory of Lévy Flights, Anomalous Transport: Foundations and Applications, Wiley-VCH, 2008.
[4] Das, S., Abraham. A., Konar. A. : Swarm intelligence algorithms in bioinformatics. Studies in Computational Intelligence Vol. 94, Springer (2008), pp. 113-147
[5] De Castro. L. N., Zuben. F. Von. : Artificial Immune Systems - Part I: Basic Theory and Applications, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil, TR DCA 01/99, 1999.
[6] Fister. I., Yang. X. S., Fister. D., Fister. J. I. : Firefly algorithm: a brief review of the expanding literature, in Cuckoo Search and Firefly Algorithm, 2014, Springer, New York, NY, USA , pp. 347–360.
[7] Gandomi. A. H., Yang. X. S., Alavi. A. H. : Cuckoo search algorithm: a meteheuristic approach to solve structural optimization problems" , Engineering with Computers, 2013, Vol. 29, No. 1, pp. 17-35.
[8] Gao, H., Diao. M., : Cultural firework algorithm and its application for digital filters design,International Journal of Modelling, Identification and Control, 2011, Vol. 14, No. 4, pp. 324-331.
[9] Guechi, E.H. , Lauber, J. , Dambrine, M., Klančar, G. and Blažič, S. (2010): PDC control design for non-holonomic wheeled mobile robots with delayed outputs, Journal of Intelligent and Robotic Systems, vol. 60, no. 3-4, pp. 395-414, Dec. 2010, DOI: 10.1007/s10846-010-9420-0
[10] Jain, N. K., Nangia. U., Jain. J. A. : Review of Particle Swarm Optimization. J. Inst. Eng. India Ser. B 99, 2018, pp. 407–411, DOI: 10.1007/s40031-018-0323-y
[11] Johanyák. Z.C. : Clonal Selection Based Parameter Optimization for Sparse Fuzzy Systems, in Proceedings of IEEE 16th International Conference on Intelligent Engineering Systems, Lisbon, Portugal, 2012, pp. 369-373.
[12] Kennedy, J.; Eberhart, R. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks IV. / Perth, 1995, pp. 1942–1948.
[13] Mareli. M., Twala. B. : An adaptive cuckoo search algorithm for optimization, Appl. Comput. Informat,2018, Vol. 14, No. 2, pp. 107-115, DOI: 10.1016/j.aci.2017.09.001.
[14] K. Mls, R. Cimler, J. Vaščák, and M. Puheim, Interactive evolutionary optimization of fuzzy cognitive maps, Neurocomputing, vol. 232, pp. 58-68, Apr. 2017.
[15] Precup, R.E. and David R.C.: Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems. Oxford, UK: Butterworth-Heinemann, Elsevier, 2019, ISBN-13: 978-0128163580
[16] Ruba, A., Tüű-Szabó, B., Kóczy, T.L., Földesi, P.: Optimization of the Time-Dependent Traveling Salesman Problem Using Interval-Valued Intuitionistic Fuzzy Sets, AXIOMS 9 : 2 p. 53 (2020), DOI: 10.3390/axioms9020053
[17] Tan, Y., Zhu, Y. : Fireworks algorithm for optimization. In International Conference in Swarm Intelligence. Springer Berlin Heidelberg, 2010, pp. 355-364.

[18] Ulutas . L. H., Kulturel-Konak. S. : A review of clonal selection algorithm and its applications, Artificial Intelligence Review, 2011, Vol. 36, pp. 117-138, DOI: 10.1007/s10462-011-9206-1.

[19] Valian. E., Mohanna. S., Tavakoli. S. : Improved cuckoo search algorithm for global optimization, Int. J. Commun. Inf.Tech., 2011, Vol. 1, No. 1, pp. 31-44., DOI: 10.5121/ijaia.2011.2304

[20] Vincze, D. and Kovács, S.: Incremental rule base creation with fuzzy rule interpolation-based Q-learning, Studies in Computational Intelligence 313 pp. 191-203. ,13 p. (2010), DOI: 10.1007/978-3-642-15220-7_16

[21] Wang, D., Tan, D., Liu. L. : Particle swarm optimization algorithm: An overview, Soft Comput., 2017, Vol. 22, No. 2, pp. 387-408, DOI: 10.1007/s00500-016-2474-6.

[22] Wang. H., Wang. W.J., Sun. H., Rahnamayan. S. : Firefly algorithm with random attractionInt, J. Bio Inspired Comput., 2016, Vol. 8, No. 1 , pp. 33-41.

[23] Yang. X.S., Deb. X. : Cuckoo search: recent advances and applications, Neural Comput Appl,2013, Vol. 24, No. 1, pp. 169–174.

[24] Yang .X.S., Firefly algorithm, in: Engineering Optimization, 2010, pp. 221–230.

[25] Yang. X.S.: Nature-inspired metaheuristic algorithms, Luniver Press; UK, 2008

[26] Yildiz. A. Z. : Cuckoo search algorithm for the selection of optimal machine parameters in milling operations, Int J Adv Manuf Technol, 2013, Vol. 64, No. 4, pp. 55-61. 10.1007/s00170-012-4013-7