MTGWA: A Multithreaded Gray Wolf Algorithm with Strategies Based on Simulated Annealing and Genetic Algorithms

Felix Martinez-Rios[1](✉), Alfonso Murillo-Suarez[1], Cesar Raul Garcia-Jacas[2], and Juan Manuel Guerrero-Valadez[1]

[1] Facutad de Ingeniería, Universidad Panamericana, Mexico City, Mexico
{felix.martinez,alfonso.murillosuarez, juanmanuel.guerrerovaladez}@up.edu.mx

[2] Departamento de Ciencias de la Computación, CICESE, Ensenada, Baja California, Mexico
cesarrjacas1985@gmail.com

Abstract. In this paper, we present an improvement of the Gray Wolf algorithm (GWO) based on a multi-threaded implementation of the original algorithm. The paper demonstrates how to combine the solutions obtained in each of the threads to achieve a final solution closer to the absolute minimum or even equal to it. To properly combine the solutions of each of the threads of execution, we use strategies based on simulated annealing and genetic algorithms. Also, we show the results obtained for twenty-nine functions: unimodal, multimodal, fixed dimension and composite functions. Experiments show that our proposed improves the results of the original algorithm.

Keywords: Nature-inspired algorithm · Optimization · Multi-threaded execution · Optimization techniques · Metaheuristics · Gray Wolf algorithm

1 Introduction

Metaheuristic optimization algorithms have developed impressively in the last two decades. Some of these, such as Artificial Bee Colony (ABC) [9,10,31], Ant Colony Optimization (ACO) [4,24,25], Bat-Inspired Algorithm (BAT) [1,19,29], Particle Swarm Optimization (PSO) [5,8,22], have had great repercussions among scientists from different fields. Metaheuristics have become popular methods for solving problems for four main reasons: simplicity, flexibility, derivation-free mechanism and optimal local avoidance [12].

The metaheuristics are quite simple and have been mostly inspired by concepts related to physical phenomena, natural phenomena or animal behaviors in nature. This simplicity allows metaheuristics to be applied to different problems

J. A. Marmolejo-Saucedo et al. (Eds.): COMPSE 2020, LNICST 359, pp. 157–172, 2021.
https://doi.org/10.1007/978-3-030-69839-3_11

without special changes in the structure of the algorithm. Also, most metaheuristics are free of derivation because, unlike gradient-based optimization approaches, metaheuristics optimizes the problems stochastically, the process begins with random solutions which are modified with the metaheuristic in the search space to find the optimum. Finally, metaheuristics have superior capabilities to avoid local optimum. Compared to conventional optimization techniques because due to the stochastic nature of metaheuristics, it allows them to avoid local solutions [16,30].

The metaheuristic algorithms can be divided into two main classes: those based on a single solution (for example, Simulated Annealing [6,11,13]) and those based on populations (for example, Firefly algorithm [3,17,26]). In the single solution based search process, start with a candidate solution. This unique candidate solution then improves throughout the iterations. The population-based metaheuristics use a set of random initial solutions called population, and are improved for the iterations. Population-based metaheuristics have some advantages over single solution metaheuristics. Multiple candidate solutions share information about the search space that provokes in sudden leaps towards more promising values; these multiple candidate solutions mutually cooperate to avoid optimal solutions locally. Finally, population-based metaheuristics generally have a higher and better search space exploration compared to algorithms based on individual solutions [2,7,30].

One of the most exciting branches of population-based metaheuristics is swarm intelligence (SI). SI algorithms are easy to implement. A common feature is the division of the search process into two phases: exploration and exploitation. The exploration phase refers to the research process of the promising areas of the search space. Exploitation refers to the search capacity in the promising regions obtained in the exploration phase. Find a proper balance between these two phases is considered a difficult task due to the stochastic nature of metaheuristics [15,21].

2 Gray Wolf Algorithm

Gray wolves are considered top predators of the food chain. Gray wolves mostly prefer to live in a herd of size with 5 to 12 wolves on average. These herds have a rigorous dominant social hierarchy. The leaders are male and female called alphas and are responsible for making decisions about hunting, resting places, etc. The pack obeys the decisions of the alpha wolf. The second level in the gray wolf hierarchy is beta. The betas are subordinate wolves that help the alpha in decision making or other activities. The beta wolf respects the alpha but gives orders to the lower level. The lesser gray wolf is omega. It may seem that omega is not an important individual in the group, but it has been observed that they maintain the structure of domination and care of the young. If a wolf is not alpha, neither beta nor omega, it is called delta, and they have functions of scouts and sentries.

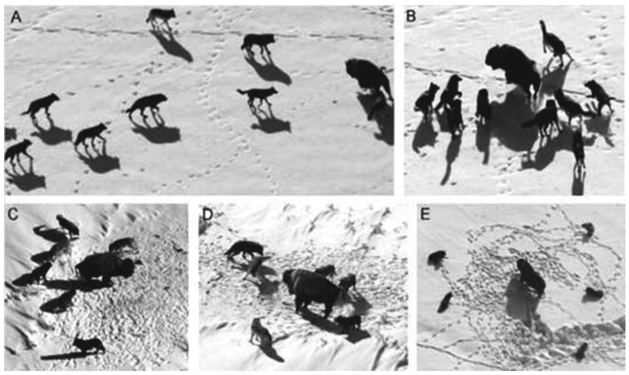

Fig. 1. Typical behaviors of wolf hunting in Yellowstone National Park. (A) Approach, track, and persecution of the prey. (B - D) Persecution, harassment and surrounding maneuver. (E) Wolves at the end of the hunt in the last configuration that approximates a regular polygon [20].

In addition to the social hierarchy of wolves mentioned above, group hunting is another interesting social behavior of gray wolves. According to Muro et al. [20], the main phases of the gray wolf hunt are:

1. Tracking, chasing, and approaching the dam.
2. Pursue, surround, and harass the dam until it stops.
3. The movement to surround it.
4. Attack on the prey.

GWO algorithm [18,27,28] models this technique of hunting and social hierarchy to perform the optimization. In Fig. 1, we can see real examples of these stages.

To model the social hierarchy of wolves, we consider that the alpha wolf (X_α) is the one in the position with the lowest value of the function to optimize in each iteration of the algorithm. Equally, the second and third-best solutions are named beta wolf (X_β) and delta wolf (X_α), respectively. As mentioned earlier, gray wolves surround their prey during hunting and to model this behavior the following equations are proposed:

$$\overrightarrow{D} = \left|\overrightarrow{C} * \overrightarrow{X_p}(iter) - \overrightarrow{X_i}(iter)\right| \tag{1}$$

$$\overrightarrow{X}(iter+1) = \overrightarrow{X_p}(iter) - \overrightarrow{A} * \overrightarrow{D} \tag{2}$$

$$\begin{aligned}\overrightarrow{D_\alpha} &= \left|\overrightarrow{C_1} * \overrightarrow{X_\alpha} - \overrightarrow{X_i}\right| \\ \overrightarrow{D_\beta} &= \left|\overrightarrow{C_1} * \overrightarrow{X_\beta} - \overrightarrow{X_i}\right| \\ \overrightarrow{D_\delta} &= \left|\overrightarrow{C_1} * \overrightarrow{X_\delta} - \overrightarrow{X_i}\right|\end{aligned} \tag{3}$$

$$\overrightarrow{A} = 2\overrightarrow{a}\overrightarrow{r_1} - \overrightarrow{a} \tag{4}$$

$$\overrightarrow{a} = 2 - 2\frac{iteration}{MaxIterations} \tag{5}$$

$$\overrightarrow{C} = 2\overrightarrow{r_2} \tag{6}$$

where *iter* indicates the current iteration, $\overrightarrow{X_p}$ is the position vector of the prey (position of the alpha wolf), and $\overrightarrow{X_i}$ indicates the position of each gray wolf in the group. The vectors $\overrightarrow{A}$ and $\overrightarrow{D}$ are calculated with the following equations:

In Eq. 5 it can be seen how the components of $\overrightarrow{a}$ are linearly reduced from 2 to 0 in the course of the iterations. On the other hand $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors in the interval $[0, 1)$.

Gray wolves can recognize the location of the prey and surround it, the alpha wolf guides this process and the beta wolf and delta have a better idea of the position of the prey than the rest of the group. To simulate the above, we keep the three best positions (alpha, beta and, delta wolves respectively) regarding the value of the function we are optimizing. The remaining wolves update their positions using the following equations:

$$\begin{aligned}\overrightarrow{X_1} &= \overrightarrow{X_\alpha} - \overrightarrow{A_1} * \overrightarrow{D_\alpha}\\ \overrightarrow{X_2} &= \overrightarrow{X_\beta} - \overrightarrow{A_2} * \overrightarrow{D_\beta}\\ \overrightarrow{X_3} &= \overrightarrow{X_\delta} - \overrightarrow{A_3} * \overrightarrow{D_\delta}\end{aligned} \tag{7}$$

$$\overrightarrow{X}(iter + 1) = \frac{\overrightarrow{X_1}(iter) + \overrightarrow{X_2}(iter) + \overrightarrow{X_3}(iter)}{3} \tag{8}$$

The pseudocode of the GWO algorithm is presented in Algorithm 1. Random parameters $\overrightarrow{A}$ and $\overrightarrow{C}$ help the candidate solutions to have hyper-spheres with different random radii for the search space exploration process. The balance between exploration and exploitation is guaranteed by the values of $\overrightarrow{a}$ and $\overrightarrow{A}$, which allow the algorithm a smooth transition between smooth exploration exploitation.

The multi-threaded implementation of the Gray Wolf Algorithm shown in Algorithm 1 starts initializing six parameters:

1. **N_GW** number of Gray Wolf by each thread τ in P.
2. **P** value that indicates how many iterations of each thread τ will execute before the implementation of the selected crossover technique.
3. **Maxlter** indicates the total number of iterations of the Gray Wolf Algorithm that each thread is going to execute.
4. $\mathbf{S^{best}}$ is the best solution obtained.

Algorithm 1: Pseudocode of the Gray Wolf Algorithm (GWO).

```
   Data: n,MaxIterations
   Result: Function fitness and position
1  begin
2      Initialize the Gray Wolf population $W = w_i(i = 1, 2, ..., n)$
3      Calculate $\overrightarrow{A}$, $\overrightarrow{a}$, and $\overrightarrow{C}$ using Eqs. 4, 5, and 6
4      Calculate the fitness for each wolf $w_i$
5      $X_\alpha$ = the best wolf agent
6      $X_\beta$ = the second best wolf agent
7      $X_\delta$ = the third best wolf agent
8      $iter = 1$
9      while $iter \leq MaxIterations$ do
10         foreach wolf $w_i$ from $W$ population do
11             Update $X_i$ position of $w_i$ using Eq. 8
12         end
13         Update $\overrightarrow{A}$, $\overrightarrow{a}$, and $\overrightarrow{C}$ using Eqs. 4, 5, and 6
14         Calculate the fitness for each wolf $w_i$
15         Update $X_\alpha$ = the best wolf agent
16         Update $X_\beta$ = the second best wolf agent
17         Update $X_\delta$ = the third best wolf agent
18     end
19     Return the best solution $X_\alpha$ and function fitness
20 end
```

The multithreaded implementation of GWA executes independent versions of Algorithm 1 on each thread, as shown in Algorithm 2 on line 7, until the number of iterations in each thread reaches the cut-off value P. Once the cutoff value is reached, all threads return the solution reached S_τ^{best}, and the best of all solutions S^{best} is obtained (lines 9 to 11 Algorithm 2), once this is done, the crossover techniques are executed. In the experiments for this work, four crossover techniques were implemented.

- **To the best(TB):** This cross method takes the thread with the best solution S^{best} and copies the values obtained for all other threads.
- **Annealing to the best (ATB):** Similar to the simulated Annealing algorithm [6], in this ease, the best solution obtained S^{best} is copied to the rest of the threads when the following condition is met: [9]:

$$rand\,[0,1) \leq exp\left(-\frac{1}{\log\left(\frac{MaxIter}{t}\right)}\right) \tag{9}$$

$$\begin{aligned} &foreach\ \tau\ in\ P \\ &\quad if\ \tau\ \neq\ \tau_{best} \\ &\quad\quad for\ i\ =\ 1\ to\ pos \\ &\quad\quad\quad S_\tau(i) = S^{best}(i) \end{aligned} \tag{10}$$

– **Genetic with the best (GTB):** This method is inspired by the procedures of the genetic algorithm [13]. A random number $pos = rand(1, dimension)$ is selected and, the first pos values of the best solution are combined with the results of each of the remaining threads as:
– **Annealing genetic with the best (AGTB):** This method combines the previous two (Annealing to best and Genetic whit the best) if Eq. 9 is true, we execute the combination of the best solution $S^{best}(i)$ with the remaining threads using the Eq. 10.

Algorithm 2: MTGWA: Multi-threaded implementation of Gray Wolf Algorithm.

```
Data: $N\_GW$, $P$, $MaxIter$, $S^{best}$
Result: $S^{best}$
1  begin
2      foreach $\tau$ in $\Gamma$ do
3          Initialize $\tau$ with $n = N\_GW$, $MaxIterations = P$
4      end
5      $t \leftarrow 0$
6      while $t \leq MaxIter$ do
7          foreach $\tau$ in $\Gamma$ do
8              Execute GWA (Algorith 1) in $\tau$ for $P$ iterations
9              $S^{best}_{\tau} \leftarrow$ best solution obtained in $\tau$
10             if $S^{best}_{\tau}$ is better than $S^{best}$ then
11                 $S^{best} \leftarrow S^{best}_{\tau}$
12             end
13         end
14         foreach $\tau$ in $\Gamma$ do
15             Execute thread-crossover technique
16         end
17         $t \leftarrow t + P$
18     end
19     Return $S^{best}$
20 end
```

In Fig. 2 we see how Algorithm 2 works. K threads of the GWO algorithm are executed; each one improves its solution S_i. When the threads reach iteration P, and each one returns its best solution. For example, in iteration P, suppose that the best solution was obtained by thread 3 (box red), then in generation $P+1$ all the wires continue to run based on the initial solution obtained by thread 3 and it is constructed using one of the four crossover methods explained above.

As seen in Fig. 2, the execution of the threads continues until the generations reach the next $2P$ value cut, and for example in this case, the best result is reached by thread 2. Then the process of the solution combination is repeated and the algorithm continues until the last generation M.

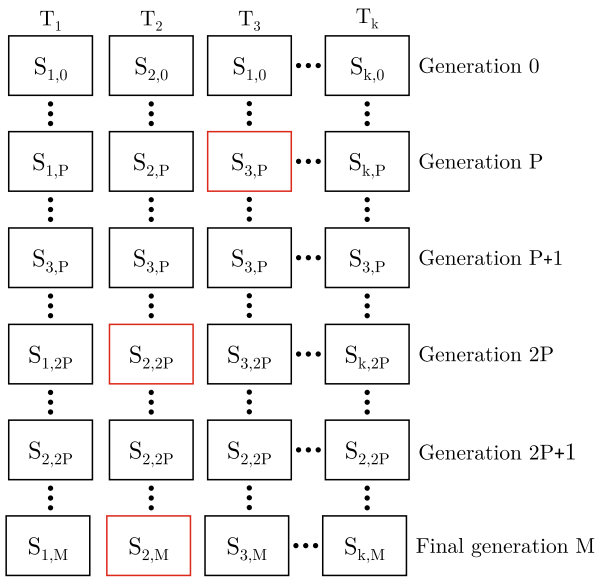

Fig. 2. Operation of the multithreaded implementation of the Gray Wolves algorithm. k is the number og threads, M is the maximum number of iterations and P is the cutoff value. (Color figure online)

3 Experiments

In this section, different experiments are carried out to tune the parameters used in Algorithm 2. The parameters we need to tune are the value of the cut generation P and the number of threads that will be used. We will use the same functions as Mirjalili et al. to compare our results with those previously obtained [18]. The benchmark function used for the experiments is shown in Tables 3, 4, 5, and 6 obtained from [14]. For all the experiments, we used a swarm with 30 gray wolves. All experiments were executed in a Dell Inspiron 7472 laptop, with Intel Core i7-8550U, with Microsoft's Windows 10 Home OS, and 16 GB of RAM. The algorithm was developed in C#.

To tune the algorithm parameters, we will execute it 30 times with each of the 13 selected multimodal and unimodal functions. In Table 1, we can see the results of the algorithm for different cut-off values P. With these results and taking into account the cut-off value with which our algorithm obtained the lowest value of the objective function, we can select $P = 100$ (making an average weighted of the P values for which the lowest optimization value was obtained).

Now with the value of $P = 100$, we will execute the algorithm modifying the number of threads that are used. Table 2 shows the results of the algorithm executions with 2 threads up to 10 threads.

With the results obtained and shown in Table 2, we performed a check with the Wilcoxon test [23] between the results obtained with the original algorithm and each of the results obtained with our MTGWA algorithm. This analysis showed that from 6 threads, our algorithm achieved better results than the original. We also observed that the average execution time was 5 s for each of the functions, but from 9 threads the execution time increased by 20%. With the previous results, we will execute the four crossover variants for 6, 7, and 8 threads.

Table 1. Results of experiments to tune the cut-off value P. The best value obtained for each function appears in bold.

Id	GWO	50	100	150	200	250	300	350	400	450
f1	6.59E-28	**9.93E-29**	1.06E-28	1.62E-28	1.86E-28	3.93E-28	5.04E-28	2.08E-28	6.28E-28	6.04E-28
f2	7.18E-17	**2.40E-17**	3.19E-17	5.06E-17	5.73E-17	3.87E-17	4.13E-17	5.01E-17	5.73E-17	3.96E-17
f3	3.29E-06	**1.10E-26**	5.15E-26	3.83E-26	3.96E-26	2.75E-26	5.37E-26	1.90E-26	7.67E-26	6.74E-26
f4	5.61E-07	**1.16E-08**	1.52E-08	1.90E-08	2.80E-08	2.35E-08	3.14E-08	2.58E-08	2.95E-08	2.51E-08
f5	26.81	26.48	26.61	26.26	26.77175	**26.33**	26.67	26.76	26.60	26.47
f6	0.816	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f7	0.00221	0.00103	0.00102	**0.00145**	0.00148	0.00158	0.00113	0.00127	0.00140	0.00179
f8	**−6123**	−3341	−3955	−4339	−4542	−5028	−5066	−5196	−5281	−5201
f9	0.310	0.480	0.342	1.33	0.432	**0.225**	0.794	0.704	0.552	0.799
f10	1.06E-13	**1.07E-14**	7.39E-14	7.96E-14	7.36E-14	7.39E-14	8.39E-14	8.25E-14	7.93E-14	2.28E-14
f11	0.00448	0.00115	0.00270	0.00122	0.00121	**0**	**0**	**0**	**0**	**0**
f12	0.0534	0.0418	0.0402	**0.0244**	0.0280	0.0334	0.0356	0.0415	0.0286	0.0266
f13	0.654464	0.430	0.431	0.480	**0.429**	0.421	0.557	0.440	0.480	0.606

Table 2. Results of executions with $P = 100$ and the number of threads between 2 and 10.

Id	GWO	2	3	4	5	6	7	8	9	10
f1	6.59E-28	1.08E-28	5.67E-29	2.56E-29	5.46E-29	2.62E-29	1.97E-29	1.76E-29	1.08E-29	1.02E-29
f2	7.18E-17	3.21E-17	1.68E-17	1.86E-17	1.67E-17	8.87E-18	1.08E-17	1.41E-17	1.25E-17	9.77E-18
f3	3.29E-06	1.53E-26	1.48E-26	9.59E-27	5.28E-27	4.68E-27	2.94E-27	1.37E-27	2.13E-27	1.68E-27
f4	5.61E-07	2.34E-08	1.68E-08	1.38E-08	6.76E-09	5.66E-09	4.63E-09	4.31E-09	4.34E-09	3.38E-09
f5	26.81	26.87	26.57	26.40	26.62	26.13	26.34	26.08	26.09	26.51
f6	0.81658	0	0	0	0	0	0	0	0	0
f7	0.00221	0.00154	0.00064	0	0	0	0	0	0	0
f8	−6123	−3837	−4376	−4293	−4381	−4423	−4394	−4543	−4521	−4444
f9	3.11E-01	5.06E-01	6.04E-15	4.26E-15	3.55E-16	0	0	0	0	3.55E-16
f10	1.06E-13	5.48E-14	6.26E-14	6.05E-14	4.31E-14	4.49E-14	2.50E-14	4.81E-14	4.13E-14	2.82E-14
f11	0.00449	0.00088	0	0	0	0	0	0	0	0
f12	0.05344	0.03485	0.01971	0.02850	0.02111	0.01842	0.02185	0.01538	0.01905	0.01377
f13	0.65446	0.45689	0.47354	0.46292	0.37474	0.32217	0.31358	0.34517	0.21400	0.30800

Table 3. Unimodal benchmark functions.

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f1$	Sphere	$f(x)=\sum_{i=1}^{d} x_i^2$	30	$x_i \epsilon\left[-100,100\right]$	$f(x)=0$
$f2$	Schwefel 2.22	$f(x)=\sum_{i=1}^{d} \lvert x_i\rvert+\prod_{i=1}^{d} \lvert x_i\rvert$	30	$x_i \epsilon\left[-10,10\right]$	$f(x)=0$
$f3$	Rotated Hyperellipsoid	$f(x)=\sum_{i=1}^{d}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$x_i \epsilon\left[-100,100\right]$	$f(x)=0$
$f4$	Schwefel 2.21	$f(x)=\underset{i=1,\ldots,d}{max} \lvert x_i\rvert$	30	$x_i \epsilon\left[-10,10\right]$	$f(x)=0$
$f5$	Rosenbrock	$f(x)=\sum_{i=1}^{d}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	$x_i \epsilon\left[-30,30\right]$	$f(x)=0$
$f6$	Step 2	$f(x)=\sum_{i=1}^{d}\lfloor x_i+0.5\rfloor^2$	30	$x_i \epsilon\left[-100,100\right]$	∞ and $f(x)=0$
$f7$	Quartic with noise	$f(x)=\sum_{i=1}^{d} i x_i^2+random\left[0,1\right]$	30	$x_i \epsilon\left[-1.28,1.28\right]$	$f(x)=0+random\left[0,1\right]$

Table 4. Multimodal benchmark functions.

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f8$	Schwefel	$f(x)=418.9829d-\sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$x_i \epsilon\left[-500,500\right]$	$f(x)=0$
$f9$	Rastrigin	$f(x)=\sum_{i=1}^{d}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	30	$x_i \epsilon\left[-5.12,5.12\right]$	$f(x)=0$
$f10$	Ackley 1	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right)+20+\exp\left(1\right)$	30	$x_i \epsilon\left[-32,32\right]$	$f(x)=0$
$f11$	Griewank	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$x_i \epsilon\left[-600,600\right]$	$f(x)=0$
$f12$	Generalized Penalized 1	$F(x)=\frac{\pi}{d}\left\{10\sin^2\left(\pi y_1\right)+\sum_{i=1}^{d-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}$ $+\sum_{i=1}^{d} u\left(x_i,a,k,m\right), where$ $y_i=1+\frac{1}{4}\left(x_i+1\right), u\left(x_i,a,k,m\right)=\begin{cases}K\left(x_i-a\right)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$ $a=10, k=100, m=4$	30	$x_i \epsilon\left[-50,50\right]$	$f(x)=0$
$f13$	Generalized Penalized 2	$f(x)=0.1\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{d-1}\left(x_i-1\right)^2\left[1+\sin^2\left(3\pi x_{i+1}\right)\right]+\left(x_d-1\right)^2\left[1+\sin^2\left(2\pi x_d\right)\right]\right\}$ $+\sum_{i=1}^{d} u\left(x_i,a,k,m\right), where$ $u\left(x_i,a,k,m\right)=\begin{cases}K\left(x_i-a\right)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$ $a=5, k=100, m=4$	30	$x_i \epsilon\left[-50,50\right]$	$f(x)=0$

Tables 4 shows the results of the final experiments of our algorithm. For unimodal functions, we see that better results were obtained in all functions, even in functions $f6$ and $f7$, the exact minimum 0 was obtained. Experiments with multimodal functions also showed better results for our algorithm, although the expected minimum value was not reached in function $f8$. For multimodal functions of fixed dimension, the results reached the minimum expected values except in three functions $f20$, $f21$ and $f23$. The executions with the composite functions showed that our algorithm was better in 4 functions of the 6 tested (Tables 7, 8 and 9).

Table 5. Multimodal fixed-dimension benchmark functions.

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f14$	Shekel's Foxhole	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}\left(x_i - a_{i,j}\right)^6}\right]^{-1}, where$ $a_{i,j} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{bmatrix}$	2	$x_i \epsilon \left[-65.536, 65.536\right]$	$f(x) = 0.9980038377$
$f15$	Kowalik	$f(x) = \sum_{i=1}^{10} \left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2, where$ $a = [0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246]$ $b = [4, 2, 1, 0.5, 0.25, 1/6, 0.125, 0.1, 1/12, 1/14, 0.0625]$	4	$x_i \epsilon \left[-5, 5\right]$	$f(x) = 0.00030748610$
$f16$	Six Hump Camel	$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2$	2	$x_i \epsilon \left[-5, 5\right]$	$f(x) = -1.0316285$
$f17$	Branin No. 1	$f(x) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos\left(x_1\right) + 10$	2	$x_i \epsilon \left[-5, 5\right]$	$f(x) = 0.397887$
$f18$	Goldstein Price	$f(x) = \left[1 + \left(x_1 + x_2 + 1\right)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]$ $\left[30 + \left(2x_1 - 3x_2\right)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	$x_i \epsilon \left[-2, 2\right]$	$f(x) = 3$
$f19$	Hartmann 3D	$f(x) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right), where$ $\alpha = \left(1.0, 1.2, 3.0, 3.2\right)^T$ $A = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix} P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$	3	$x_i \epsilon \left[0, 1\right]$	$f(x) = -3.86$
$f20$	Hartmann 6D	$f(x) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right), where$ $\alpha = \left(1.0, 1.2, 3.0, 3.2\right)^T$ $A = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix} P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$	6	$x_i \epsilon \left[0, 1\right]$	$f(x) = -3.32$
$f21$	Shekel 4.5	$f(x) = -\sum_{i=1}^{5} \left(\sum_{j=1}^{4} \left(x_j - C_{ji}\right)^2 + \beta_i\right)^{-1}, where$ $\beta = \frac{1}{10}\left(1, 2, 2, 4, 4, 6, 3, 7, 5, 5\right)^T$ $C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$	4	$x_i \epsilon \left[0, 10\right]$	$f(x) = -10.1532$
$f22$	Shekel 4.7	$f(x) = -\sum_{i=1}^{7} \left(\sum_{j=1}^{4} \left(x_j - C_{ji}\right)^2 + \beta_i\right)^{-1}, where$ $\beta = \frac{1}{10}\left(1, 2, 2, 4, 4, 6, 3, 7, 5, 5\right)^T$ $C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$	4	$x_i \epsilon \left[0, 10\right]$	$f(x) = -10.403$
$f23$	Shekel 4.10	$f(x) = -\sum_{i=1}^{10} \left(\sum_{j=1}^{4} \left(x_j - C_{ji}\right)^2 + \beta_i\right)^{-1}, where$ $\beta = \frac{1}{10}\left(1, 2, 2, 4, 4, 6, 3, 7, 5, 5\right)^T$ $C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$	4	$x_i \epsilon \left[0, 10\right]$	$f(x) = -10.5364$

Table 6. Composite benchmark functions.

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f24$	CF1	$f_1, f_2, ..., f_{10} = Sphere\, Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/100, 5/100, ..., 5/100]$	30	$x_i \epsilon \left[-5, 5\right]$	0
$f25$	CF2	$f_1, f_2, ..., f_{10} = Griewank\, Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/100, 5/100, ..., 5/100]$	30	$x_i \epsilon \left[-5, 5\right]$	0

(*continued*)

Table 6. (*continued*)

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f26$	CF3	$f_1, f_2, ..., f_{10} = Griewank\,Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [1, 1, 1, ..., 1]$	30	$x_i \epsilon [-5, 5]$	0
$f27$	CF4	$f_1, f_2 = Ackley\,Function$ $f_3, f_4 = Rastrigin\,Function$ $f_5, f_6 = Weierstrass\,Function$ $f_7, f_8 = Griewank\,Function$ $f_9, f_{10} = Sphere\,Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	30	$x_i \epsilon [-5, 5]$	0
$f28$	CF5	$f_1, f_2 = Rastrigin\,Function$ $f_3, f_4 = Weierstrass\,Function$ $f_5, f_6 = Griewank\,Function$ $f_7, f_8 = Ackley\,Function$ $f_9, f_{10} = Sphere\,Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	30	$x_i \epsilon [-5, 5]$	0
$f29$	CF6	$f_1, f_2 = Rastrigin\,Function$ $f_3, f_4 = Weierstrass\,Function$ $f_5, f_6 = Griewank\,Function$ $f_7, f_8 = Ackley\,Function$ $f_9, f_{10} = Sphere\,Function$ $[\delta_1, \delta_2, ..., \delta_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5,$ $0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	30	$x_i \epsilon [-5, 5]$	0

4 Conclusion

In this paper, we show a multi-threaded implementation of the gray wolf algorithm. Several threads of gray wolf algorithms are executed in this new algorithm and the results are combined with four different strategies every 100 generations. The experiments with twenty-nine benchmark functions showed that our proposal was better in 17 functions and in 6 of the functions the same results were obtained as the original algorithm that is equal to the minimum reported.

Table 7. MTGWA with six threads final experiments.

Id	GWO	TB	ATB	GTB	AGTB
f1	6.590E-28	2.991E-29	6.827E-29	6.151E-29	**1.041E-28**
f2	7.180E-17	1.240E-17	2.326E-17	2.170E-17	2.665E-17
f3	3.290E-06	4.847E-27	1.033E-26	9.350E-27	1.287E-26
f4	5.610E-07	4.866E-09	9.915E-09	7.925E-09	1.176E-08
f5	26.8126	26.5574	26.4353	**26.0196**	26.2580
f6	8.1658E-01	**0**	**0**	**0**	**0**
f7	2.2130E-03	1.9221E-05	**0**	**0**	**0**

(*continued*)

Table 7. (*continued*)

Id	GWO	TB	ATB	GTB	AGTB
f8	−6123	−8199	−7217	−7955	**−8201**
f9	3.105E-01	8.882E-16	5.921E-17	2.961E-16	1.066E-15
f10	1.060E-13	4.296E-14	5.504E-14	3.147E-14	5.930E-14
f11	4.4850E-03	**0**	**0**	**0**	**0**
f12	5.3438E-02	2.5081E-02	2.3753E-02	1.8494E-02	2.3009E-02
f13	6.5446E-01	3.3783E-01	3.1999E-01	**2.2355E-01**	2.7501E-01
f14	4.0425	1.3948	1.2297	1.0973	1.3952
f15	3.3700E-05	3.0750E-04	3.0764E-04	3.0783E-04	3.0890E-04
f16	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
f17	0.3979	0.3979	0.3979	0.3979	0.3979
f18	3	3	3	3	3
f19	−3.8626	−3.8628	−3.8628	−3.8628	−3.8628
f20	−3.2865	−3.3224	−3.3224	−3.3224	−3.3224
f21	−10.1514	−10.1531	−10.1531	−10.1531	−10.1531
f22	−10.4015	−10.4028	−10.4028	−10.4028	−10.4028
f23	−10.5343	−10.5363	−10.5363	−10.5363	−10.5363
f24	4.384E+01	1.759E-31	4.902E-31	4.652E-31	1.277E-30
f25	9.180E+01	1.528E-15	6.200E-16	4.248E-16	4.847E-16
f26	6.144E+01	6.300E-15	2.866E-15	2.558E-15	1.598E-15
f27	123.124	432.319	335.355	483.908	559.494
f28	1.0214E+02	9.092E-17	6.666E-17	3.052E-17	5.828E-17
f29	43.143	821.745	860.151	811.769	837.791

Table 8. MTGWA with seven threads final experiments.

Id	GWO	TB	ATB	GTB	AGTB
f1	6.590E-28	1.557E-29	4.644E-29	5.045E-29	7.605E-29
f2	7.180E-17	1.135E-17	2.131E-17	1.892E-17	2.837E-17
f3	3.290E-06	3.952E-27	8.333E-27	5.157E-27	1.658E-26
f4	5.610E-07	5.252E-09	9.2246E-09	8.558E-09	1.183E-08
f5	26.8126	26.5158	26.2442	26.5281	26.1631
f6	8.1658E-01	**0**	**0**	**0**	**0**
f7	2.2130E-03	4.8910E-06	**0**	**0**	**0**
f8	−6123	−8150	−7125	−7851	−8190
f9	3.105E-01	1.184E-16	7.224E-15	**0**	2.901E-15

(*continued*)

Table 8. (*continued*)

Id	GWO	TB	ATB	GTB	AGTB
f10	1.060E-13	4.343E-14	4.923E-14	2.839E-14	5.267E-14
f11	4.4850E-03	**0**	**0**	**0**	**0**
f12	5.3438E-02	1.8182E-02	2.4452E-02	1.7096E-02	2.4125E-02
f13	6.5446E-01	2.8916E-01	2.9516E-01	2.3428E-01	3.6272E-01
f14	4.0425	1.1624	1.0641	1.0641	1.4285
f15	3.3700E-05	3.0749E-04	**3.0769E-04**	3.0751E-04	3.0808E-04
f16	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
f17	0.3979	0.3979	0.3979	0.3979	0.3979
f18	3	3	3	3	3
f19	−3.8626	−3.8628	−3.8628	−3.8628	−3.8628
f20	−3.2865	−3.3224	−3.3224	−3.3224	−3.3224
f21	−10.1514	−0.9832	−10.1531	−10.1531	−10.1531
f22	−10.4015	−10.4028	−10.4028	−10.4028	−10.4028
f23	−10.5343	−10.5363	−10.5363	−10.5363	−10.5363
f24	4.384E+01	**1.182E-31**	3.268E-31	4.351E-31	1.053E-30
f25	9.180E+01	1.783E-15	5.091E-16	2.509E-16	**8.106E-17**
f26	6.144E+01	6.196E-15	1.296E-15	1.550E-15	6.232E-16
f27	123.124	527.919	348.933	439.759	385.945
f28	1.0214E+02	9.29E-17	8.356E-17	2.134E-17	7.274E-17
f29	43.143	909.596	770.303	821.229	859.656

Table 9. MTGWA with eight threads final experiments.

Id	GWO	TB	ATB	GTB	AGTB
f1	6.590E-28	1.849E-29	4.920E-29	3.111E-29	7.364E-29
f2	7.180E-17	**1.085E-17**	1.833E-17	1.770E-17	2.670E-17
f3	3.290E-06	**2.215E-27**	5.244E-27	8.742E-27	1.367E-26
f4	5.610E-07	**4.316E-09**	7.917E-09	7.891E-09	9.706E-09
f5	26.8126	26.4452	26.3718	26.2147	26.1552
f6	8.1658E-01	**0**	**0**	**0**	**0**
f7	2.2130E-03	1.8666E-05	**0**	**0**	**0**
f8	−6123	−7918	−6878	−7810	−8079
f9	3.105E-01	5.921E-17	5.921E-17	3.553E-16	4.145E-16
f10	1.060E-13	4.047E-14	3.869E-14	**2.792E-14**	5.835E-14
f11	4.4850E-03	**0**	**0**	**0**	**0**

(*continued*)

Table 9. (*continued*)

Id	GWO	TB	ATB	GTB	AGTB
f12	5.3438E-02	1.8677E-02	1.9437E-02	**1.2636E-02**	1.8983E-02
f13	6.5446E-01	2.6864E-01	2.8205E-01	2.3913E-01	3.1093E-01
f14	4.0425	1.0643	1.0973	**0.9980**	1.2297
f15	3.3700E-05	3.1171E-04	3.0753E-04	3.0750E-04	3.0764E-04
f16	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
f17	0.3979	0.3979	0.3979	0.3979	0.3979
f18	3	3	3	3	3
f19	−3.8626	−3.8628	−3.8628	−3.8628	−3.8628
f20	−3.2865	−3.3224	−3.3224	−3.3224	−3.3224
f21	−10.1514	−10.1531	−10.1531	−10.1531	−10.1531
f22	−10.4015	−10.4028	−10.4028	−10.4028	−10.4028
f23	−10.5343	−10.5363	−10.5363	−10.5363	−10.5363
f24	4.384E+01	1.256E-31	3.870E-31	3.026E-31	1.013E-30
f25	9.180E+01	7.095E-16	5.730E-16	1.674E-16	8.484E-17
f26	6.144E+01	6.320E-15	9.230E-16	1.317E-15	**6.220E-16**
f27	123.124	445.452	297.489	542.278	476.529
f28	1.0214E+02	1.323E-16	6.588E-17	**1.589E-17**	5.394E-17
f29	43.143	859.377	878.409	796.441	815.896

References

1. Akhtar, S., Ahmad, A.R., Abdel-Rahman, E.M.: A metaheuristic bat-inspired algorithm for full body human pose estimation. In: 2012 9th Conference on Computer and Robot Vision, pp. 369–375 (2012)
2. Bertsimas, D., Tsitsiklis, J.: Simulated annealing. Stat. Sci. **8**, 10–15 (1993)
3. Patle, B.K., Parhi, D.R., Jagadeesh, A., Kashyap, S.K.: On firefly algorithm: optimization and application in mobile robot navigation. World J. Eng. **14**(1), 65–76 (2017). https://doi.org/10.1108/WJE-11-2016-0133
4. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**(2), 243–278 (2005)
5. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, 1995. MHS 1995, pp. 39–43, October 1995
6. Henderson, D., Jacobson, S., Johnson, A.: The Theory and Practice of Simulated Annealing, pp. 287–319. Springer, Boston, April 2006. https://doi.org/10.1007/0-306-48056-5_10
7. Ingber, L.: Simulated annealing: practice versus theory. Math. Comput. Model. **18**(11), 29–57 (1993)

8. Jiang, Y., Hu, T., Huang, C., Wu, X.: An improved particle swarm optimization algorithm. Appl. Math. Comput. **193**(1), 231–239 (2007). https://doi.org/10.1016/j.amc.2007.03.047. http://www.sciencedirect.com/science/article/pii/S009630030-700392X
9. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
10. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**(1), 21–57 (2014)
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671. https://science.sciencemag.org/content/220/4598/671
12. Kouba, N.E.L.Y., Boudour, M.: A brief review and comparative study of nature-inspired optimization algorithms applied to power system control. In: Li, X., Wong, K.-C. (eds.) Natural Computing for Unsupervised Learning. USL, pp. 35–49. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-98566-4_2
13. Koulamas, C., Antony, S., Jaen, R.: A survey of simulated annealing applications to operations research problems. Omega **22**(1), 41–56 (1994). https://doi.org/10.1016/0305-0483(94)90006-X. http://www.sciencedirect.com/science/article/pii/030504839490006X
14. Liang, J.J., Suganthan, P.N., Deb, K.: Novel composition test functions for numerical global optimization. In: Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005, vol. 2005, pp. 68–75, June 2005
15. Martinez-Rios, F., Murillo-Suarez, A.: A new swarm algorithm for global optimization of multimodal functions over multi-threading architecture hybridized with simulating annealing. Procedia Comput. Sci. **135**, 449–456 (2018). https://doi.org/10.1016/j.procs.2018.08.196, 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
16. Martinez-Rios, F., Murillo-Suarez, A.: Packing algorithm inspired by gravitational and electromagnetic effects. Wireless Netw. **26**(8), 5631–5644 (2019). https://doi.org/10.1007/s11276-019-02011-9
17. Memari, A., Ahmad, R., Akbari Jokar, M.R., Abdul Rahim, A.R.: A new modified firefly algorithm for optimizing a supply chain network problem. Appl. Sci. 9(1), p. 7 (2019). https://doi.org/10.3390/app9010007
18. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007. http://www.sciencedirect.com/science/article/pii/S0965997813001853
19. Mishra, S., Shaw, K., Mishra, D.: A new meta-heuristic bat inspired classification approach for microarray data. Procedia Technology **4**, 802–806 (2012). https://doi.org/10.1016/j.protcy.2012.05.131, http://www.sciencedirect.com/science/article/pii/S2212017312004100, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012
20. Muro, C., Escobedo, R., Spector, L., Coppinger, R.: Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations. Behav. Process. **88**(3), 192–197 (2011). https://doi.org/10.1016/j.beproc.2011.09.006. http://www.sciencedirect.com/science/article/pii/S0376635711001884

21. Olorunda, O., Engelbrecht, A.P.: Measuring exploration/exploitation in particle swarms using swarm diversity. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 1128–1134, June 2008. https://doi.org/10.1109/CEC.2008.4630938
22. Ouyang, Z., Liu, Y., Ruan, S.J., Jiang, T.: An improved particle swarm optimization algorithm for reliability-redundancy allocation problem with mixed redundancy strategy and heterogeneous components. Reliab. Eng. Syst. Saf. **181**, 62–74 (2019). https://doi.org/10.1016/j.ress.2018.09.005. http://www.sciencedirect.com/science/article/pii/S0951832018304125
23. Rey, D., Neuhäuser, M.: Wilcoxon-signed-rank test. In: International Encyclopedia of Statistical Science, pp. 1658–1659, January 2011. https://doi.org/10.1007/978-3-642-04898-2_616
24. Socha, K., Blum, C.: An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput. Appl. **16**(3), 235–247 (2007). https://doi.org/10.1007/s00521-007-0084-z
25. Tirkolaee, E.B., Alinaghian, M., Hosseinabadi, A.A.R., Sasi, M.B., Sangaiah, A.K.: An improved ant colony optimization for the multi-trip capacitated arc routing problem. Comput. Electr. Eng. **77**, 457–470 (2019). https://doi.org/10.1016/j.compeleceng.2018.01.040. http://www.sciencedirect.com/science/article/pii/S0045790617330501
26. Wang, G.G., Guo, L., Duan, H., Wang, H.: A new improved firefly algorithm for global numerical optimization. J. Comput. Theor. Nanosci. **11**, 477–485 (2014). https://doi.org/10.1166/jctn.2014.3383
27. Wang, J.S., Li, S.X.: An improved grey wolf optimizer based on differential evolution and elimination mechanism. Sci. Rep. **9**(1), 1–21 (2019). https://doi.org/10.1038/s41598-019-43546-3
28. Long, W., Xu, S.: A novel grey wolf optimizer for global optimization problems. In: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 1266–1270 (2016)
29. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Gonzalez, J.R., Pelta, D.A., Cruz, C., Terrazas G., Krasnogor N. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65–74. Springer, Berlin Heidelberg (2010). https://doi.org/10.1007/978-3-642-12538-6_6
30. Yang, X.-S., He, X.: Nature-inspired optimization algorithms in engineering: overview and applications. In: Yang, X.-S. (ed.) Nature-Inspired Computation in Engineering. SCI, vol. 637, pp. 1–20. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30235-5_1
31. Zhou, J., et al.: An individual dependent multi-colony artificial bee colony algorithm. Inf. Sci. **485**, 114–140 (2019). https://doi.org/10.1016/j.ins.2019.02.014. http://www.sciencedirect.com/science/article/pii/S0020025519301239 http://www.sciencedirect.com/science/article/pii/S0020025519301239