Journal of Information Systems and Telecommunication
Vol.10, No.1, January-March 2022, 1-10
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Statistical Analysis and Comparison of the Performance of Meta-Heuristic Methods Based on their Powerfulness and Effectiveness

Mehrdad Rohani[1*], Hassan Farsi[1], Seyed Hamid Zahiri[1]

[1].Department of Electronic and Computer Engineering, University of Birjand, Birjand, Iran

Received: 04 May 2021 / Revised: 24 Jul 2021/ Accepted: 24 Aug 2021
DOI: https://doi.org/10.52547/jist.16067.10.37.49

Abstract

In this paper, the performance of meta-heuristic algorithms is compared using statistical analysis based on new criteria (powerfulness and effectiveness). Due to the large number of meta-heuristic methods reported so far, choosing one of them by researchers has always been challenging. In fact, the user does not know which of these methods are able to solve his complex problem. In this paper, in order to compare the performance of several methods from different categories of meta-heuristic methods new criteria are proposed. In fact, by using these criteria, the user is able to choose an effective method for his problem.

For this reason, statistical analysis is conducted on each of these methods to clarify the application of each of these methods for the users. Also, powerfulness and effectiveness criteria are defined to compare the performance of the meta-heuristic methods to introduce suitable substrate and suitable quantitative parameters for this purpose. The results of these criteria clearly show the ability of each method for different applications and problems.

Keywords: Effectiveness; Meta-heuristic Algorithms; Optimization; Powerfulness; Statistical Analysis.

1- Introduction

Optimization is conducting a process to find the best acceptable answer by considering the limits and requirements of the issue. To solve an optimization issue, there might be different answers for the desired optimal parameter, in which function, namely the goal function, is defined to compare these answers and choosing an optimal answer. An optimization method must be able to extract the optimal answer for this function. The advance of the computer in the last five decades leads to the improvement of the optimization methods. Each of these methods has a different ability to solve an optimization problem. However, due to existing the great number of optimization methods, the most important question that arises is which method is suitable, and provides the best performance for solving the problem. These methods can be classified into three broad categories: Enumeration methods, calculates-based methods, and random methods, which are explained in the following.

Enumeration methods: In each iteration, only one point t belonging to the answer space is examined. This category of methods is simpler than the others in terms of implementation but needs considerable calculations. In these methods, no mechanism exists to decrease the scope of the search space, and the scope of the search space is very large. For instance, dynamic programming is an example of these methods that acts completely unintelligent [1, 2].

Calculates-based methods: In these methods, the set of necessary and sufficient conditions are used that apply to the answer to the problem. These methods usually use the gradients of the goal to search. It is possible that sometimes, due to discontinuity of the goal function. Its gradients cannot be calculable. Therefore, these methods also face challenges [3].

Random methods: One of the uncommon methods to find the optimal answer for a problem is to consider all of the possible answers. In this case, the goal function is calculated for all of the possible answers and at the end, the best answer is selected. In this case, the complete count leads to the exact answer to the problem. Using this method in practical problems, is not possible due to the vast range of possible answers for the problem. Given this issue, the effort is always to present methods that have the ability to decrease the search space. To solve this problem, random search methods such as heuristic methods and meta-heuristic methods are presented. This category of

✉ **Hassan Farsi**
Hfarsi@Birjand.ac.ir

methods is able to present a proper answer and close to the optimal answer in a limited time and unlike enumeration and calculates based methods. Random search methods are mainly based on enumeration methods except that they use additional information to guide the search. These methods are completely general in terms of the application area. They are able to solve very complex problems.

The main problem of the heuristic methods is that they get caught in local optimal points and early converge to these points. Meta-heuristic methods are presented to solve this problem [4]. In fact, meta-heuristic methods belong to the category of methods that have a solution to exit local optimal points. These methods are able to be used in a broad range of problems. Optimization methods are used in various fields [5, 6]. The user chooses one of the methods based on the application.

In this paper, we try to examine a set of methods and present statistical analysis. For this reason we determine the stability of each method and their real-time performance. Also, two new criteria powerfulness and effectiveness have been introduced and the performance of each methods has been examined by these two new criteria. In the next section, meta-heuristic methods from different categories are introduced. In the third section, benchmarks as well as conventional evaluation criteria (Best fitness, average run time and standard deviation) are introduced. Also, new criteria and statistical analysis have been introduced in this section. In the fourth section, the tests and results are reviewed.

2- Meta-Heuristic Methods

During the last three decades, the introduction of new meta-heuristic methods and also their application in different devices has been considerably increasing. In 1983, Kirkpatrick proposed the simulated annealing method [7]. In 1992, Koza introduced the genetic programming method [8]. After that Walker et al introduced the first algorithm based on bee colonies in 1993 [9]. In 1994, the term of meta-heuristic was used by Golver when introducing the tabu search method [10]. In 1995, Kennedy and Eberhat introduced particle swarm optimization [11]. In 2002, Passino introduced the bacterial foraging-based method [12]. In 2008, a bio-geography-based optimization algorithm was introduced by Simon [13]. Many methods in this area have been presented in the last decade. In 2014, the grey wolf algorithm was introduced by Mirjalili et al. [14]. In 2015, the Ant Lion Optimizer (ALO) was presented by Mirjalili [15]. In the same year, the moth-flame optimization algorithm was presented by Mirjalili [14]. In 2016, Mirjalili introduced dragonfly (DA), multi-verse (MVO), sine cosine (SCA), and whale optimization (WOA) algorithms for optimization [17-20]. In 2017, the salp swarm algorithm (SSA) was presented by Mirjalili [21]. Development of these algorithms has occurred at a high speed in recent years.

These methods are able to exit the local optimal answers and move to the global optimal answer in a short period of time. The important factor in these methods is the dynamic balance between the exploration and exploitation strategies. Exploration is able to properly search the answer space. Exploitation strategy performs the search operation in spaces with more possibility and prevents the loss of time in search space in which the possibility of the optimal answer is low. Meta-heuristic methods are divided by categories of methods including methods based on single-point and based on population, inspired by nature and without inspiration by nature, with memory and memoryless, and probabilistic-definitive. Some of the meta-heuristic methods are memoryless. They do not use the information obtained during the search. Some of them take advantage of the obtained information. Single point-based methods change an answer during the search process, while population-based methods consider a set of answers during the search [22]. Generally, this type of method is slower comparing to the single-point methods but they are able to produce more desirable answers. However, due to the advance in computer calculation power, the population-based methods hold more importance. Among these methods, WOA, GWO, DA and SSA can be pointed out. All of these methods belong to the category of nature-inspired methods which have memory and they are also based on population. Another category of these methods is presented in figure 1 [23].

2-1- Whale Optimization Algorithm (WOA)

This method is inspired by social behavior, the mechanism of which is based on the social movement of humpback whales and how they hunt. Humpback whales are able to identify the location of the prey [20]. The primary location of the search agents in modeling the algorithm is considered to be close to the desired situation, and after determining the best search agent, other agents update their location according to that. This behavior is modeled through the Eq. (1) and Eq. (2). All the equations in this subsection are adopted from [20].

$$\vec{D} = \left|\vec{C}\vec{X}(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D} \quad (2)$$

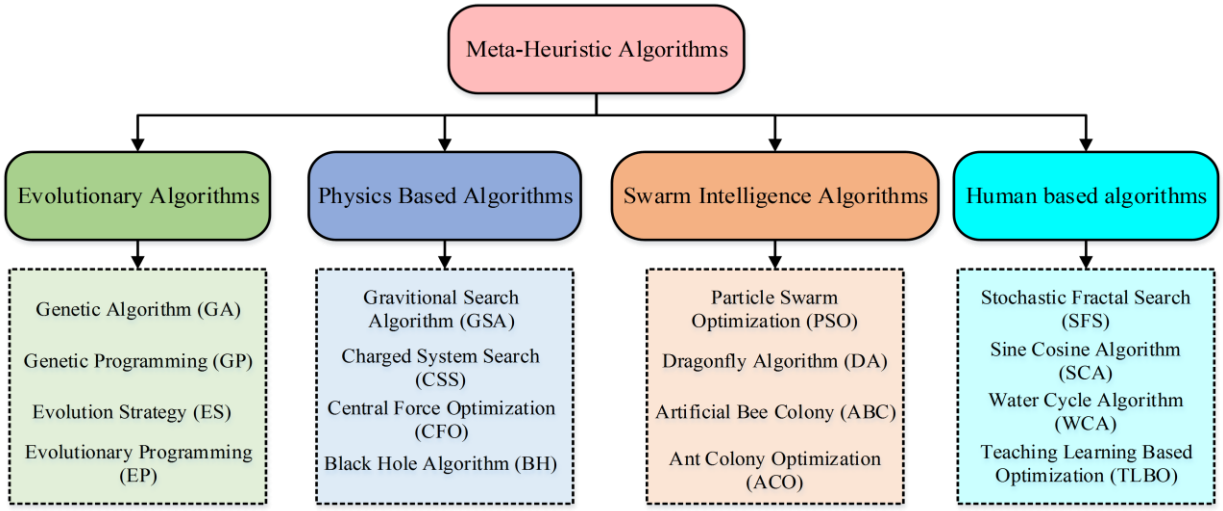

Fig. 1. Classification of meta-heuristic algorithms [23]

Where t is the current iteration, $\vec{A}$ and $\vec{C}$ are the coefficient vectors, $\overrightarrow{X^*}$ is the location vectors for the best response obtained in the t iteration and $\vec{X}$ is the location vector. Also, $\overrightarrow{X^*}$ must be updated in each iteration in case of the existence of a better answer. Coefficient vectors $\vec{A}$ and $\vec{C}$ are calculated with Eq. (3) and Eq. (4).

$$\vec{A} = 2\vec{a}.\vec{r}.\vec{a} \quad (3)$$

$$\vec{C} = 2\vec{r} \quad (4)$$

Where a is decreased linearly from 2 to zero with the increase in iterations. Also, the vector r is a random number in the range between 0 and 1.

Bubble-net attacking method (exploitation phase): In this attack, two methods of contraction blocking mechanism according to Eq. (1) and spiral updating location according to Eq. (2) are used. There is a 50% probability that the whale will choose one of these two mechanisms, and there is a 50% possibility that the whale chooses one of these two mechanisms. Each of these behaviors is mathematically modeled by Eq. (5) and Eq. (6).

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D} \quad if\ p \geq 0.5 \quad (5)$$

$$\vec{X}(t+1) = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X^*}(t) \quad if\ p < 0.5 \quad (6)$$

Where p is a random number between zero and one, b is a constant to define a logarithmic spiral shape, and it is a random number between 1 and -1, $\overrightarrow{D'}$ is the distance of the lth whale to the prey, which is represented as Eq. (7).

$$\overrightarrow{D'} = \overrightarrow{X^*}(t) - \vec{X}(t) \quad (7)$$

Search for prey (exploitation phase): In this method, random selection is used to update the search agents to let the algorithm perform a global search in the search space, which is modeled by Eq. (8) and Eq. (9).

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \quad (8)$$

$$\vec{D} = \left|\vec{C}.\overrightarrow{X_{rand}} - \vec{X}\right| \quad (9)$$

Where X_{rand} is the selected random location between the current population and $|A| \geq 1$ is chosen to make the search agents perform a global search.

2-2- Grey Wolf Optimization Algorithm (GWO)

Footnotes should be typed in singled-line spacing at the bottom of the page and column where it is cited. Footnotes should be rare. The grey wolf method is one of the methods inspired by social behavior. Grey wolf is considered as the highest-ranking hunter since there is no natural hunter for these animals. These wolves live in a group of 5 or 15, and their leader is recognized as alpha wolf (α). Alpha wolves are not the strongest members of the group but they are the best at managing it. The second category in terms of the hierarchy is the beta (β), which helps α in decision makings. If it is necessary, the β wolves who replace the alpha. The third category is the delta wolves (δ), and after that, is the omega (ω) wolves. To model the social behavior of the wolves, a random population is generated. The grey wolf algorithm takes advantage of three answers, namely alpha, beta, and delta, and omega answers follow these three answers [14]. To model the three phases, first, it is necessary for the points around the prey to be predetermined and then the wolves will start moving toward it and at the end, the attack will start. Location vectors of the wolves are modeled using the Eq. (10) and Eq. (11). All the equations in this subsection are adopted from [14].

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \quad (10)$$

$$\vec{D} = \left|\vec{C}.\overrightarrow{X_{rand}} - \vec{X}\right| \quad (11)$$

Where t is the running iteration, $\vec{A}$ and $\vec{C}$ are the coefficient vectors, $\overrightarrow{X_P}$ is the prey vector, and $\vec{X}$ indicates the location vector of a grey wolf. Vectors A and C are calculated using the Eq. (12) and Eq. (13).

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1}.\vec{a} \quad (12)$$

$$\vec{C} = 2\overrightarrow{r_2} \quad (13)$$

Where a reduces linearly from 2 to 0 with the increase of the iterations. Also, r_1 and r_2 vectors are random numbers between 0 and 1. To mathematically model the behavior of the grey wolves, it is assumed that the alpha (the best candidate), beta, and delta have the best information about the prey. Therefore, the three obtained answers are saved and the location of the other wolves is updated based on these answers. The Eq. (14) and Eq. (15) indicate the modeling of the wolves' behavior.

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}\overrightarrow{X_\alpha}(t) - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}\overrightarrow{X_\beta}(t) - \vec{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}\overrightarrow{X_\delta}(t) - \vec{X}\right| \quad (14)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}, \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}, \overrightarrow{X_3} = \overrightarrow{X_\beta} - \overrightarrow{A_3} \quad (15)$$

After calculating theX_i, point X will be updated as Eq. (16) in the next level.

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (16)$$

Finally, after finishing the iterations, the location of the alpha wolf will be selected as the optimum point.

2-3- Dragonfly Optimization Algorithm (DA)

The dragonfly algorithm is based on mass intelligence and inspired by nature. The main idea is based on the behavior of dragonflies while hunting for food and prey [17]. The mass behavior and mass formation of the dragonflies are performed for two purposes: prey which is called the static mass or nutrition, and the immigrant or dynamic mass. These behaviors are modeled by the Eq. (17). All the equations in this subsection are adopted from [17].

$$S_i = \sum_{j=1}^{N} X - X_j$$

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}$$

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \quad (17)$$

Where S_i is separation, A_i is alignment, and C_i is cohesion. X is current location, N is the number of neighbors, X_j is the jth neighbor, and V_j is the speed of the jth neighbor. The location of the food (goal) and the enemy (search agent) is modeled by Eq. (18) and Eq. (19).

$$F_i = X^+ - X$$

$$E_i = X^- + X \quad (18)$$

Where X is the location of the search agent. X^+ is the location of the food source, and X^- is the location of the enemy. The behavior of the dragonflies is modeled by Eq. (19).

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \quad (19)$$

s, a, c, f and e are the weight values for adjusting the exploration and exploitation processes. The parameter W is the weight of inertia and t is the iteration counter. The position of each search agent is expressed according to the Eq. (20).

$$X_{t+1} = X_t + \Delta X_{t+1} \quad (20)$$

To improve the exploration in the search space and modeling the static behavior of the dragonflies, when there is no neighbor the random walking process is added to this algorithm to update the location of the dragonflies according to the Eq. (21).

$$X_{t+1} = X_t + Lavy(d) \times X_t \;\; ; \;\; Lavy(d) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \quad (21)$$

Where d is the dimension of the location vector and $Lavy(d)$ forms the random walking process, where r_1 and r_2 are random numbers between 0 and 1, and β is a constant number that is assumed to be 1.5. σ is calculated according to the Eq. (22) where $\Gamma(x) = (x-1)!$.

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}}\right)^{\frac{1}{\beta}} \quad (22)$$

Finally, the location of the dragonflies is updated based on the two static and dynamic behaviors and the best answer is selected as the optimal answer to the problem [17].

2-4- Salp Swarm Optimization Algorithm (SSA)

This method is inspired by the social behavior of the salps. These creatures move in the deep waters in groups and under the name of salp chain. This behavior, as some researchers believe, is for better movement and fast access to food [21]. To model the behavior of the salps, they are divided into two groups of leader and followers. The leader is the first member of the salps chain and the others are called the followers. The food source for the salps is known as the *F* matrix, and the location of the salps is modeled by the Eq. (23). All the equations in this subsection are adopted from [21].

$$X_j^1 = \begin{cases} F_j + c_1\left((ub_j - lb_j)c_2 + lb_j\right) & c_3 \geq 0 \\ F_j - c_1\left((ub_j - lb_j)c_2 + lb_j\right) & c_3 < 0 \end{cases} \quad (23)$$

Where X_j^1 is the location of the first salp (leader) in the jth dimension, F_j is the location of the food in the j th dimension, ub_j is the upper bound of the jth dimension, and lb_j is the lower bound of the jth dimension. The parameters c_2 and c_3 are random numbers between 0 and 1, but the parameter c_1 has an important role, namely exploration, in the search space, and is modeled by Eq. (24).

$$c_1 = 2e^{-(\frac{4l}{L})^2} \quad (24)$$

Where l is the running iteration and L is the maximum iteration. The location of the follower salps is expressed by Eq. (25).

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \quad (25)$$

Where $i \geq 2$ and x_j^i is the location of the ith follower salp in the jth dimension. In this way, the SSA algorithm updates the location of the leader and its followers in each iteration and introduces the best location as the best answer in the last iteration [21].

3- Stability Analysis of Meta-Heuristic Methods Based on Statistical Methods

So far, different methods have been reported to evaluate the stability of meta-heuristic methods based on statistical analysis. In this section, we introduce the proposed method while referring to the methods that have been expressed so far in research.

3-1- Conventional Stability Analysis Methods

Stability analysis of the meta-heuristic methods is performed using two methods of mathematical analysis and statistical analysis. Dorigo in the optimization method of ants ACO proved by limiting the pheromone that the ant method converges to the optimal answer, and the method will be stable [24]. Clerc performed statistical analysis for the particle swarm method and guaranteed the convergence under some conditions for the available parameters in the problem [25]. In this research, Monte Carlo evaluation is used to analyze the stability of the meta-heuristic methods. This analysis includes the examination of convergence in different optimization problems. Also, the ability of each method in encountering each of these problems is determined with high confidence through many iterations. The time parameter is also calculated for each algorithm so that a user can have the ability to choose a method based on online and offline applications. Also, two new metrics namely powerfulness and effectiveness are introduced to examine the convergence and power of a method in obtaining the global answer.

3-1-1- Benchmark Functions

Researchers face varies problems with different complexities. They use different meta-heuristic algorithms to solve and optimize their problems. We have used 23 benchmark functions with different complexities including unimodal, multimodal, and fixed-dimension. The performance of each algorithm has been tested by solving these 23 problems [26-28]. These functions are presented in table 1-3. Fmin indicates the optimum point in the search space. Table 1 indicates the F1-F7 functions that have one global optimum point, and search the exploitation process in the search space, and test the performance of a method. Benchmark functions of F8-F13 are multimodal functions and they are shown in table 2. This category has several local optimum points, and these local optimum points exponentially increase with respect to the increase in the dimension. These functions are perfectly capable of test the exploration process in the search space. The third category is the fixed-dimension multimodal. These functions also have one global optimum point and several local optimum points that, similar to the second category, analyze the effectiveness and powerfulness of a method. These are shown in table 3. Figure 2 indicates the 3-D representation of the benchmark functions' search space.

Table 1: Description of unimodal benchmark

Function	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$F_4(x) = \boldsymbol{max}_i\{\lvert x_i \rvert,\ 1 \leq i \leq n\}$	[-100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	[-30,30]	0
$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	[-100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	[-1.28,1.28]	0

Although these answers have been reported in the base articles, since meta-heuristic algorithms act randomly, a method has to be repeated a considerable number of times and an average of the result has to be presented. As such, the average of the best answer has been calculated for each algorithm with 1000 iterations for each run and 100 runs for each algorithm. These results are reported in table 4, table 5, table 6 and table 7. Also the convergence curves are reported in red color for each method on benchmark functions in figure 3.

3-1-2- Best Fitness Metric

The first assessment metric of every method is the best answer of the algorithm to each benchmark function.

3-1-3- Average Run Time Metric

The second assessment metric is the run time of each algorithm. In this metric, the average of the run time of each algorithm for convergence to 5% range of the best answer obtained has been calculated and reported in the tables. This metric can affect the selection of a user when using an optimization method, and, depending on the application, it can be a tradeoff between the best answer and the run time of the algorithm.

3-1-4- Standard Deviation Metric

The next analysis is the standard deviation parameter. This parameter indicates the deviation rate of the calculated answers and the reliability of a method.

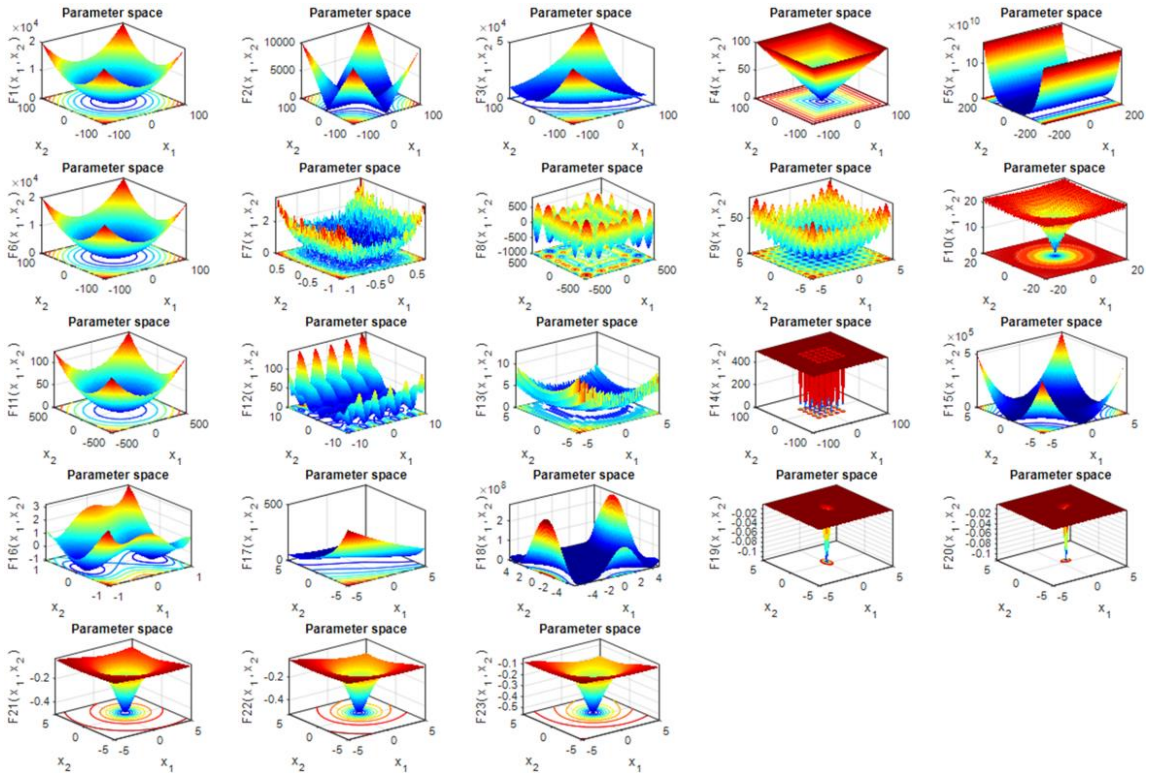

Fig. 2. Search space of composite benchmark functions.

Table 2: Description of multimodal benchmark

Function	Range	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500,500]	-418.98×5
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12,5.12]	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600]	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i,10,100,4)$, $y_i = 1+\frac{x_i+1}{4}$, $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$	[-50,50]	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	[-50,50]	0

Table 3: Description of fixed-dimension multimodal benchmark

Function	Range	f_{min}
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	[-65, 65]	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	[-5,5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	[-5,5]	-1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos x_1 + 10$	[-5,5]	0.398
$F_{18}(x) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)] \times [30+(2x_1-3x_2)^2 \times (18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	[-2,2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	[0,1]	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	[0,1]	-3.32
$F_{21}(x) = -\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	[0,10]	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	[0,10]	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	[0,10]	-10.5363

3-2- The Proposed Stability Analysis Methods

3-2-1- Powerfulness (PF)

A new metric namely the powerfulness is introduced to investigate the convergence of each method. The number of search agents has been considered to be 30 for all methods. Each search agent has fitness in each iteration of the experiment. With increasing the number of iterations and progress of the algorithm, the fitness values are updated in each iteration. In this metric, fitness is investigated after 1000 iterations. If the amount of algorithm fitness is convergence to the global answer with a 0.001 precision, then that algorithm has obtained the desired answer. In Eq. (26), N is equal to 100. This means that the algorithm is repeated 1000 times in each run and this operation will be averaged 100 times. The sign 1(.) will be equal to 1 if the condition inside the parentheses is satisfied. In Eq. (27), f_{global} indicates the global optimum answer and the $f_{calculated}$ shows the calculated optimum answer.

$$PF = Powerfullness = \frac{\sum_{i=1}^{N} 1(\Delta f_i \leq 0.001)}{N} \quad (26)$$

$$\Delta f_i = |f_{global} - f_{calculated}| \quad (27)$$

3-2-2- Effectiveness (EF)

Effectiveness has been introduced to investigate the powerfulness of an algorithm in obtaining the global answer. In this metric, the fitness of all of the search agents is examined after 1000 iterations. If the fitness rate of all of the search agents is exactly equal to the global answer, then that algorithm has obtained the desired answer. In Eq. (28), N is equal to 100. This means that the algorithm is repeated 1000 times in each run and this operation will be averaged 100 times. The sign 1(.) will be equal to 1 if the condition inside the parentheses is satisfied. In Eq. (29) f_{global} indicates the global optimum answer and the $f_{calculated}$ shows the calculated optimum answer.

$$EN = Effectiveness = \frac{\sum_{i=1}^{N} 1(\Delta f_i = 0)}{N} \quad (28)$$

$$\Delta f_i = |f_{global} - f_{calculated}| \quad (29)$$

4- Experimental Results

In this section, the results of the introduced different analysis are presented. The obtained results are presented in tables and graphs. The tables 4-7 also include two sections. The first section of the tables includes the metrics of the best answer, standard deviation, and average time for 23 benchmark functions in different columns. The analysis of each method is reported in a separate table. The column "Reported" is the obtained results of the main reference of each method that has reported the average values of the best answer (Avg Best) and the deviation rate (Avg Std). The column "Calculated" is the results of the performed runs in this research that include the average of the best answer (Avg Best), standard deviation (Avg Std), and average run time (Avg Time). In the second part of the tables including two columns indicates the new metric namely the effectiveness and powerfulness on 23 functions for each algorithm in the "EN" and "PF" columns.

As an instance, three functions F4, F9 and F15 were selected from 23 benchmark functions. In figure 3, convergence curves are shown for 1000 iterations and 100 runs on each algorithm.

4-1- Discussion on Results

In this section, some tables are provided to examine the obtained results and to summarize the large amount of the obtained data. In these tables, we have tried to evaluate the results in a way so that the user can easily select the method in accordance with their needs and application. In this section, the averaging operation is performed on the F1-F7 benchmark functions called unimodal, F8-F13 functions called the Multimodal, and F14-F23 functions named the Fixed-dimension multimodal. The highest value is specified as Bold and the lowest value is specified as Underline in each column.

Average PF criterion

The PF criterion is investigated in Table 8. In this table, the DA method has been able to provide the best result for the unimodal functions with 100 % value. The GWO, and SSA methods are next in the ranking with 85.71%. Also, the WOA method with 59.85% has the lowest amount in the PF criterion. The SSA method in multi-modal functions could have the best performance in this type of function with 100%. In these functions, the WAO, GWO and DA methods have the equal ranking with 83.33%. The best value for the fixed-dimension multimodal functions is 100% for the SSA and DA methods while the WOA method has the lowest amount with 80.70%. As observed from table 8, the WOA method has the weakest performance in the PF metric.

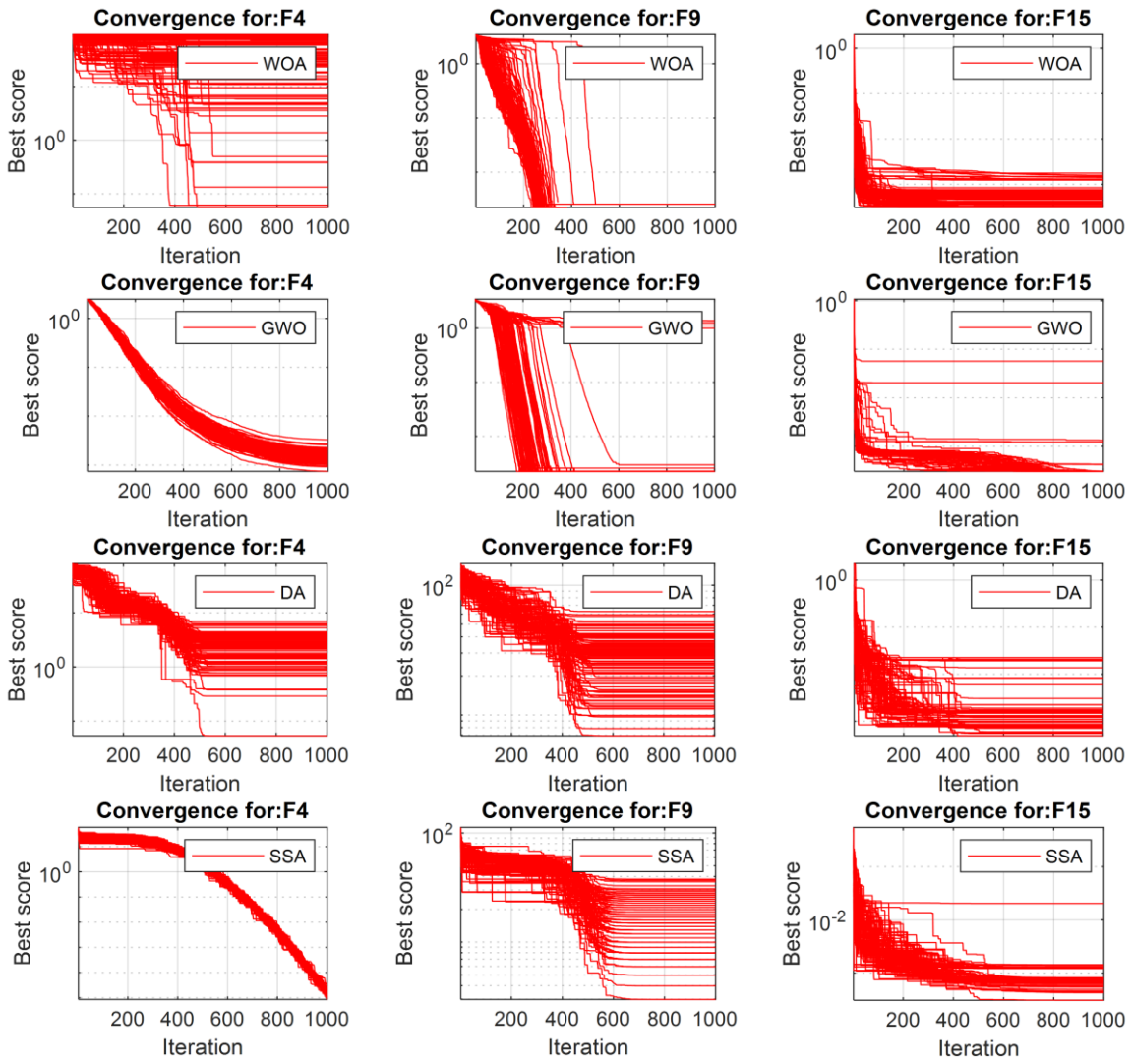

Fig.3. Convergence curves of WOA, GWO, DA and SSA algorithms that obtained from F4, F9 and F15

Table 4: WOA algorithm

WOA	Reported		Calculated				
Func Test	**Avg Best**	**Avg Std**	**Avg Best**	**Avg Std**	**Avg Time**	**EN**	**PF**
F1	1 .41E −30	4 .91E −30	6.59E-149	4.59E-148	7.70E-02	100	100
F2	1 .06E −21	2 .39E −21	1.17E-102	1.09E-101	8.33E-02	100	100
F3	5 .39E −07	2 .93E −06	20783.72516	10157.99167	0.296841885	0	0
F4	0 .072581	0 .39747	36.54531187	31.21106177	0.048915693	0	19
F5	27 .86558	0 .763626	27.13631191	0.510781163	0.06949779	0	100
F6	3 .116266	0 .532429	0.086487861	0.123716391	0.043686051	0	100
F7	0 .001425	0 .001149	0.001603156	0.001861557	1.42E-01	0	0
F8	−5080.76	695.7968	-10989.67576	1612.087325	0.055816373	0	0
F9	0	0	0	0	0.030713043	100	100
F10	7 .4043	9 .897572	4.09E-15	2.23E-15	8.11E-02	100	100
F11	0 .0 0 0289	0 .001586	0.006625063	0.026231674	0.024578934	94	100
F12	0 .339676	0 .214864	0.00603913	0.010095281	0.297053286	5	100
F13	1 .889015	0 .266088	0.222387626	0.178598496	0.271053991	0	100
F14	2 .111973	2 .498594	2.265421628	2.647242836	0.232368918	0	100
F15	0 .0 0 0572	0 .0 0 0324	0.000663949	0.000350605	2.88E-02	87	100
F16	−1 .03163	4 .2E −07	-1.031628453	5.10E-11	5.08E-02	99	100
F17	0 .397914	2 .7E −05	0.397887754	9.41E-07	3.07E-02	18	100
F18	3	4 .22E −15	3.00004213	0.000223778	1.14E-02	3	100
F19	−3 .85616	0 .002706	-3.859688471	0.003343901	2.80E-02	12	100
F20	−2 .98105	0 .376653	-3.194394261	0.255546232	0.032039408	7	100
F21	−7 .04918	3 .629551	-8.835498564	2.475814202	0.02778591	0	19
F22	−8 .18178	3 .829202	-8.194635405	2.801778303	0.052576967	0	43
F23	−9 .34238	2 .414737	-8.759948348	2.843792657	0.061611228	0	45

Table 5: GWO algorithm

GWO	Reported		Calculated				
Func Test	**Avg Best**	**Avg Std**	**Avg Best**	**Avg Std**	**Avg Time**	**EN**	**PF**
F1	6.59E-28	6.34E-05	4.28E-59	1.08E-58	2.35E-01	100	100
F2	0.029014	0.042144	1.24E-34	1.86E-34	2.31E-01	100	100
F3	3.29E-06	79.14958	1.19E-14	5.91E-14	3.80E-01	100	100
F4	5.61E-07	1.315088	1.51E-14	2.84E-14	2.13E-01	100	100
F5	26.81258	69.90499	26.69458625	0.742391628	0.216549825	0	100
F6	0.816579	0.000126	0.621146399	0.371011473	0.204170201	6	100
F7	0.002213	0.100286	0.000871956	0.000492435	2.58E-01	0	0
F8	-6123.1	-4087.44	-5928.936636	879.2135455	0.262058383	0	0
F9	0.310521	47.35612	0.405729452	1.451143899	0.04926481	93	100
F10	1.06E-13	0.077835	1.64E-14	2.88E-15	1.21E-01	100	100
F11	0.004485	0.006659	0.003455183	0.008551483	5.97E-02	85	100
F12	0.053438	0.020734	0.038594141	0.02273283	0.520416024	0	100
F13	0.654464	0.004474	0.546402772	0.199821466	0.513642556	2	100
F14	4.042493	4.252799	3.855341713	4.114698941	0.191481026	0	100
F15	0.000337	0.000625	0.004964973	0.008460522	2.60E-02	77	100
F16	-1.03163	-1.03163	-1.031628448	5.88E-09	7.65E-02	100	100
F17	0.397889	0.397887	0.397898355	7.68E-05	5.67E-02	86	99
F18	3.000028	3	3.000009343	1.19E-05	3.46E-02	22	99
F19	-3.86263	-3.86278	-3.861542428	0.002668951	4.39E-02	1	100
F20	-3.28654	-3.25056	-3.26296912	0.093013085	0.05581884	0	100
F21	-10.1514	-9.14015	-9.450634359	1.847937725	0.095443428	0	100
F22	-10.4015	-8.58441	-10.19071187	1.042754866	0.128936701	0	100
F23	-10.5343	-8.55899	-10.15788663	1.53690521	0.138736578	0	100

Table 6: WOA algorithm

DA	Reported		Calculated				
Func Test	Avg Best	Avg Std	Avg Best	Avg Std	Avg Time	**EN**	**PF**
F1	2.85E-18	7.16E-18	8.709019492	8.470920307	8.907355455	0	100
F2	1.49E-05	3.76E-05	1.541030894	1.180641229	10.18661535	0	100
F3	1.29E-06	2.10E-06	174.129871	375.1554082	7.222378913	1	100
F4	0.000988	0.002776	2.329263546	1.416677941	11.25669589	0	100
F5	7.600558	6.786473	891.914574	1802.132876	9.897716262	1	100
F6	4.17E-16	1.32E-15	8.566349466	13.79450972	11.06746413	0	100
F7	0.010293	0.004691	0.019568582	0.012024944	8.895041751	0	100
F8	-2857.58	383.6466	-2818.040967	355.3376662	15.73808298	0	0
F9	16.01883	9.479113	27.50861574	11.72206506	13.87184983	0	100
F10	0.23103	0.487053	2.6695387	1.019690292	12.87132673	0	100
F11	0.193354	0.073495	0.777684867	0.300217353	11.87462016	3	100
F12	0.031101	0.098349	1.682915888	1.241529733	11.20859426	0	100
F13	0.002197	0.004633	0.618542299	0.541132351	10.60521735	0	100
F14	103.742	91.24364	1.047468983	0.406581789	1.589713017	0	100
F15	193.0171	80.6332	0.003213229	0.005656206	6.15745856	0	100
F16	458.2962	165.3724	-1.031626118	7.82E-06	4.344534415	37	100
F17	596.6629	171.0631	0.397888712	5.26E-06	4.605963481	0	100
F18	229.9515	184.6095	3.000012725	3.43E-05	4.557858122	0	100
F19	679.588	199.4014	-3.862609267	0.000666809	5.160709293	0	100
F20	2.85E-18	7.16E-18	-3.254038178	0.095517834	12.11661363	0	100
F21	1.49E-05	3.76E-05	-7.423171	2.765308816	8.4690914	12	100
F22	1.29E-06	2.10E-06	-8.53677162	2.682979658	7.306447437	15	100
F23	0.000988	0.002776	-8.216681515	2.963678316	8.738156894	24	100

Table 7: GWO algorithm

SSA	Reported		Calculated				
Func Test	Avg Best	Avg Std	Avg Best	Avg Std	Avg Time	**EN**	**PF**
F1	0	0	1.27E-08	3.54E-09	0.181400388	100	100
F2	0.2272	1	0.009109382	0.066197136	0.102852086	97	100
F3	0	0	2.41E-09	1.54E-09	0.209767524	100	100
F4	0	0.6556	1.55E-05	4.02E-06	0.131577107	100	100
F5	0	0	97.88033843	301.1603942	0.104756503	0	100
F6	0	0	6.69E-10	2.52E-10	0.139161432	100	100
F7	0.0028	0.007	0.005825155	0.003611733	0.141422613	0	0
F8	1	0.0071	-2843.882009	300.9224241	0.108420644	0	100
F9	0.4254	0.9502	17.92912747	7.936964991	0.104917015	0	100
F10	0.0598	0.5279	0.692505073	0.860483282	0.11545747	58	100
F11	0	0	0.201908936	0.084030016	0.152391405	0	100
F12	0	0	0.39478531	0.806546346	0.228802092	72	100
F13	0	0	0.002297837	0.004722304	0.249327454	82	100
F14	0.0557	0.809	1.017884379	0.13986487	0.186448041	0	100
F15	0	0	0.001850349	0.004276931	0.027172082	73	100
F16	0.1952	0.1527	-1.031628453	8.35E-15	0.099933664	0	100
F17	0	0.0651	0.397887358	1.65E-14	0.090015482	0	100
F18	0.1417	0.5571	3	9.31E-14	0.108162701	0	100
F19	0.0832	0.7059	-3.862782148	5.77E-14	0.139162503	0	100
F20	_	_	-3.22247955	0.046978076	0.121644601	0	100
F21	_	_	-8.340784046	2.85966839	0.10572867	70	100
F22	_	_	-8.715288573	2.853571407	0.104817694	80	100
F23	_	_	-9.445250741	2.555722593	0.129785398	83	100

Table 8: Average PF criterion

Algorithm	Function type		
	Unimodal	Multi modal	Fixed-dimension multimodal
WOA	59.85	83.33	80.70
GWO	85.71	83.33	99.80
DA	**100**	83.33	**100**
SSA	85.71	**100**	**100**

Besides the powerfulness metric, the convergence rate of each method on each benchmark function can be examined by convergence graphs depicted using red color in figure 3. This test is performed visually. By observing any red convergence graph it can be realized that how cohesive an algorithm behaves on a benchmark function. If the graphs are drawn coherently and close to each other, the convergence rate is high for example GWO and SSA methods on F4 function. On the contrary, if the drawn lines are at a distance from each other, it shows the different behavior of each algorithm on a benchmark function. This means that an algorithm results in different answers in each run and the reliability on that benchmark function is decreased by that algorithm for example WOA algorithm on F4 functions.

Average EF criterion

In this section, average values EF are shown in table 9. The first column indicates the average values for the unimodal function, in which the SSA method has the highest value with 71%. This means that the SSA method has provided the highest value in obtaining the global optimum answer. In this column, the DA method has the weakest performance to obtain the global answer with only 0.28%. The second column indicates the performance of the methods in multi-modal functions. In this column, the WOA method has the best performance with 49.83%. Also, the DA method is known as the weakest method with 0.50%. The third column is calculated for fixed-dimension multimodal functions. In this column, the SSA method results in the best performance with 30.6%. Also, the DA method provides the lowest value with 8.80%.

Table 9: Average EN criterion

Algorithm	Function type		
	Unimodal	Multi modal	Fixed-dimension multimodal
WOA	28.57	**49.83**	22.60
GWO	58.00	46.66	28.60
DA	0.28	0.50	8.80
SSA	**71**	35.33	**30.6**

Average run time

One of the important matters in engineering discussions is to obtain the desired answer in the shortest time. Thus, the average run time of each algorithm is investigated in Table 10. The average time is calculated on one type of benchmark function to test the powerfulness of each method in run time. In the first column, the average run time on the unimodal functions is calculated that includes F1-F7 functions. The fastest algorithm is the WOA algorithm with 0.1091 seconds while the DA method is the slowest method with 9.6333 seconds. In the multi-modal functions that include F8-F13 functions, the WOA algorithm is the best algorithm with 0.1267 seconds, and the DA algorithm has the weakest performance with 12.6949 seconds. In the third column, the fixed-dimension multimodal functions that include F14-F23 functions are present. In this column, the WOA algorithm is the fastest algorithm with 0.0556 seconds while the DA method is the slowest method in terms of runtime with only 6.3046 seconds. The obtained results show that the DA method is the slowest algorithm in comparison with the other methods.

Table 10: Average run time (sec)

Algorithm	Function type		
	Unimodal	Multi modal	Fixed-dimension multimodal
WOA	**0.1091**	**0.1267**	**0.0556**
GWO	0.2483	0.2543	0.0848
DA	9.6333	12.6949	6.3046
SSA	0.1444	0.1598	0.1112

5- Conclusion

In the research performed in 2020, more than 300 meta-heuristic methods have been reported since the appearance of the genetic algorithm [29]. In the past, the number of meta-heuristic methods was limited. For example, users chose between genetic algorithms and particle swarm optimization. Nowadays users being confused about choosing the type of the method. Undoubtedly, due to the great numbers of meta-heuristic methods, it is time to write or even invent new metrics and standards to classify and rank these methods. Otherwise, we are facing with a large number of meta-heuristic algorithms that are constantly being added to their collection. This causes that the user is unable to choose an appropriate method for his problem.

In this paper, we have tried to investigate the statistical performance of WOA, GWO, DA and SSA algorithms besides defining powerfulness and effectiveness metrics. These metrics help users to compare the performance of different meta-heuristic methods. In this regard, a suitable platform and solution are provided to choose between them by the users.

In future work, we will gradually introduce new evaluation criteria in order to accurately use meta-heuristic algorithms.

References

[1] R. Bellman, "Dynamic Programming", Science, Vol. 153, No. 3731, 1966, pp. 34-37.

[2] W. Kuo, V. R. Prasad, F. A. Tillman, and C.-L. Hwang, Optimal Reliability Design: Fundamentals and Applications, Cambridge University Press, 2001.

[3] J. A. Snyman, Practical Mathematical Optimization. Springer, 2005.

[4] I. BoussaïD, J. Lepagnot, and P. Siarry, "A Survey on Optimization Metaheuristics", Information Sciences, Vol. 237, 2013, pp. 82-117.

[5] A. Sezavar, H. Farsi, and S. Mohamadzadeh, "A Modified Grasshopper Optimization Algorithm Combined with CNN for Content Based Image Retrieval", International Journal of Engineering, Vol. 32, No. 7, 2019, pp. 924-930.

[6] A. H. Hosseinian and V. Baradaran, "A Multi-Objective Multi-Agent Optimization Algorithm for the Community Detection Problem", J. Inform. Syst. Telecommun, Vol. 6, No. 1,2019, pp. 169-179.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization By Simulated Annealing", Science, Vol. 220, No. 4598, 1983, pp. 671-680.

[8] J. R. Koza and J. R. Koza, Genetic Programming, On the Programming of Computers :Natural Selection MIT press, 1992.

[9] A. Walker, J. Hallam, and D. Willshaw, "Bee-Havior in a Mobile Robot: The Construction of a Self-Organized Cognitive Map and Its Use in Robot Navigation within a Complex, Natural Environment", IEEE International Conference on Neural Networks, 1993, pp. 1451-1456.

[10] F. Glover, "Tabu Search for Nonlinear and Parametric Optimization (With Links to Genetic Algorithms)", Discrete Applied Mathematics, Vol. 49, No. 1-3, 1994, pp. 231-255.

[11] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in Proceedings of ICNN'95-International Conference on Neural Networks, 1995, Vol. 4, pp. 1942-1948.

[12] K. M. Passino, "Biomimicry of Bacterial Foraging for Distributed Optimization and Control", IEEE control systems magazine, Vol. 22, No. 3, 2002, pp. 52-67.

[13] D. Simon, "Biogeography-Based Optimization", IEEE Transactions on Evolutionary Computation, Vol. 12, No. 6, 2008, pp. 702-713.

[14] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer", Advances in Engineering Software, Vol. 69, 2014, pp. 46-61.

[15] S. Mirjalili, "The Ant Lion Optimizer", Advances in Engineering Software, Vol. 83, 2015, pp. 80-98.

[16] S. Mirjalili, "Moth-flame Optimization Algorithm: A Novel Nature-Inspired Heuristic Paradigm", Knowledge-Based Systems, Vol. 89, 2015, pp. 228-249.

[17] S. Mirjalili "Dragonfly Algorithm: a New Meta-Heuristic Optimization Technique for Solving Single-Objective, Discrete, and Multi-Objective Problems", Neural Computing and Applications, Vol. 27, No. 4, 2016, pp. 1053-1073.

[18] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-Verse Optimizer: a Nature-Inspired Algorithm for Global Optimization", Neural Computing and Applications, Vol. 27, No. 2, pp. 495-513, 2016.

[19] S. Mirjalili, "SCA: a Sine Cosine Algorithm for Solving Optimization Problems", Knowledge-Based Systems, Vol. 96, 2016, pp. 120-133.

[20] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm", Advances in Engineering Software, Vol. 95, 2016, pp. 51-67.

[21] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A Bio-Unspired Optimizer for Engineering Design Problems", Advances in Engineering Software, Vol. 114, 2017, pp. 163-191.

[22] S. M. Almufti, "Historical Survey on Metaheuristics Algorithms", International Journal of Scientific World, Vol. 7, No. 1, 2019, pp. 1.

[23] S. Shirke and R. Udayakumar, "Evaluation of Crow Search Algorithm (CSA) for Optimization in Discrete Applications", International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 584-589.

[24] M. Dorigo and G. Di Caro, "Ant Colony Optimization: a New Meta-Heuristic", in Proceedings of the Congress on Evolutionary Computation-CEC99, Vol. 2, 1999, pp. 1470-1477.

[25] M. Clerc, Particle Swarm Optimization. John Wiley & Sons, 2010.

[26] J. G. Digalakis and K. G. Margaritis, "On Benchmarking Functions for Genetic Algorithms", International Journal of Computer Mathematics, Vol. 77, No. 4, 2001, pp. 481-506.

[27] M. Molga and C. Smutnicki, "Test Functions for Optimization Needs", Test Functions for Optimization Needs, Vol. 101, 2005, pp. 48.

[28] X.-S. Yang, "Firefly Algorithm, Stochastic Test Functions and Design Optimisation," International Journal of Bio-Inspired Computation, Vol. 2, No. 2, 2010, pp. 78-84.

[29] D. Molina, J. Poyatos, J. Del Ser, S. García, A. Hussain, and F. Herrera, "Comprehensive Taxonomies of Nature-and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations", Cognitive Computation, Vol. 12, No. 5, 2020, pp. 897-9339.