An Adaptive Average Grasshopper Optimization Algorithm for Solving Numerical Optimization Problems

NAJWAN OSMAN-ALI, JUNITA MOHAMAD-SALEH
School of Electrical and Electronic Engineering,
Universiti Sains Malaysia,
Nibong Tebal 14300, Penang,
MALAYSIA

Abstract: - The grasshopper optimization algorithm (GOA), inspired by the behavior of grasshopper swarms, has proven efficient in solving globally constrained optimization problems. However, the original GOA exhibits some shortcomings in that its original linear convergence parameter causes the exploration and exploitation processes to be unbalanced, leading to a slow convergence speed and a tendency to fall into a local optimum trap. This study proposes an adaptive average GOA (AAGOA) with a nonlinear convergence parameter that can improve optimization performance by overcoming the shortcomings of the original GOA. To evaluate the optimization capability of the proposed AAGOA, the algorithm was tested on the CEC2021 benchmark set, and its performance was compared to that of the original GOA. According to the analysis of the results, AAGOA is ranked first in the Friedman ranking test and can produce better optimization results compared to its counterparts.

Key-Words: - Grasshopper optimization algorithm, GOA, meta-heuristics, optimization, swarm intelligence.

Received: October 29, 2022. Revised: March 16, 2023. Accepted: April 11, 2023. Published: May 10, 2023.

1 Introduction

Optimization is important for producing fast and accurate solutions to various problems. Most optimization problems are challenging to solve because of their nonlinearity, multimodal objective landscape, and nonlinear constraints, [1]. Optimization techniques for solving such problems can be divided into two main categories, [2], traditional and nature-inspired. Traditional algorithms for solving optimization problems include gradient-based, interior-point, and trust-region methods. Even with modern computers, these algorithms can be computationally intensive when computing derivatives, particularly for problems with discontinuities in their objective functions, and may not have derivatives in certain regions. Such limitations of traditional methods have diverted optimization research towards solutions based on nature-inspired methods. Nature-inspired algorithms work as global optimizers based on interacting agents to generate search moves within the search space.

In recent years, numerous nature-inspired algorithms have been developed that can be categorized as single- or population-based. A single-agent algorithm generated a single solution for each run. Examples of single-based algorithms are Guided Local Search (GLS), [3], Variable Neighborhood Search (VNS), [4], and Iterated Local Search (ILS) [5]. Under population-based agents, all algorithms emulate the behavior of nature, such as swarming, physics, evolutionary, and human behavior, where they generate a set of multiple solutions in each run, [6]. Under the swarming category, the algorithm's source of information is collective behavior in nature, such as bee movement when collecting honey or deciding to move to a new nest, or the movement of ants foraging for food. Popular algorithms in this category include Particle Swarm Optimization (PSO), [7], Artificial Bee Colony (ABC), [8], [9], Bat algorithm (BA), [10], and Ant Colony Optimization (ACO), [11]. It is also possible to create an algorithm based on physics phenomena such as the Gravitational Search Algorithm (GSA), [12], Water Evaporation Optimization (WEO), [13], and Thermal Exchange Optimization (TEO), [14]. Another category involves algorithms inspired by evolutionary phenomena such as selection, recombination, and mutation. Popular algorithms in this category include the Genetic Algorithm (GA), [15], [16], Evolution Strategy (ES), [17], and Differential Evolution (DE), [18], [19]. The last category comprises algorithms that emulate human behavior. Examples of algorithms in this category are the Teaching Learning-Based Algorithm (TLBA), [20], Imperialist Competitive Algorithm (ICA), [21], and Harmony Search (HS), [22].

Although nature-inspired algorithms have different approaches and methodologies for solving optimization problems, they all have one common set of phases during the search process: exploration and exploitation. In the exploration phase, the algorithm explores the search space as w idely as possible for solutions, whereas during exploitation, the algorithm focuses its searches on areas surrounding promising regions for potential optimal solutions. A key to nature-inspired algorithms is to achieve a balance between exploration and exploitation.

With numerous optimization algorithms, there is no conclusion regarding which algorithm performs best for all optimization problems, [23]. This problem is highlighted in the "No Free Lunch (NFL) theorem", [24], which states that no single state-of-the-art optimization algorithm can be expected to perform better than any other algorithm on a ll classes of optimization problems. Therefore, using this inference, building a nature-inspired algorithm should be based on the application on w hich the algorithm is to be used, instead of building an all-around working algorithm.

Among the various nature-inspired algorithms, the Grasshopper Optimization Algorithm (GOA) has been successfully applied to various applications, such as optimizing parameters on support vector machines, [25], solving optimization problems in an automatic voltage regular system, [26], and obtaining the values of seven unknown parameters of a proton exchange membrane fuel-cell stack, [27]. Although the GOA can produce well-optimized solutions, it has several shortcomings, [28]. One of these is its linear convergence parameter, which causes the exploration and exploitation phases to become unbalanced, leading to a slow convergence speed and a tendency to fall into one of the local optima traps. Various studies have been conducted to overcome these shortcomings, and enhanced GOA can be categorized into variant and hybrid versions. Under the variant category, the original GOA was improved by integrating the Levy Flight, [29], employing chaotic maps to balance exploration and exploitation, [30], and using a natural selection strategy and dynamic feedback mechanism, [31]. The modified algorithms are categorized as GOA variants. In the hybrid category, the original GOA algorithm or its variants are combined with other nature-inspired algorithms, such as the Genetic Algorithm (GA), [32], ABC, [33], and Grey Wolf Optimizer (GWO), [34]. The hybrid strategy usually provides a better ability for the algorithm to move out of a local optimum trap or solve any movement issues, but usually at the cost of additional complexity and a longer computational time.

This work aims to overcome the disadvantages of the GOA through two improvements, producing a GOA variant referred to as AAGOA. The first improvement to the AAGOA employs a modified parameter convergence value that balances the exploitation and exploration of the GOA, and the second improvement is the implementation of an adaptive average for enhanced fitness of grasshopper agents. The proposed AAGOA was tested using the CEC2021 real-parameter optimization benchmark problems, [35], and compared with the original GOA.

The remainder of this paper is organized as follows. Section 2 introduces the concept of the original GOA. The proposed AAGOA is explained in detail in Section 3. This is followed by Section 4, which presents and compiles the experimental results based on the CEC2021 benchmark or test functions to assess AAGOA performance. Finally, Section 5 c oncludes the study based on the simulation results.

2 Grasshopper Optimization Algorithm (GOA)

Grasshoppers behave differently, depending on their environment. In a sw arm, one grasshopper tends to first move independently and then try its best to evade the other grasshoppers. Only when a grasshopper is triggered by other grasshoppers, such as a touch on its leg will it become aggressive and begin swarming. A swarm of grasshoppers moves to find a source of food from one place to another, [36]. This swarm intelligence behavior was used as the basis for the computational GOA development.

In [37], the authors used the behavior of a swarm of grasshoppers to develop a search strategy in the search space for optimization problems. A key aspect that enables the GOA to converge to a solution is the interaction between the grasshoppers and agents. In a sw arm, the GOA considers three primitive corrective zones of behavior among grasshopper agents: the attraction, comfort, and repulsion zones. Fig. 1 illustrates how the GOA implements interactions between grasshoppers. Each grasshopper had a comfort zone represented by a sphere, as shown in Fig. 1. Any grasshopper within the radius of the comfort zone will have a neutral force, that is, its attraction force will be the same as the repulsion force. Grasshopper A was used as a reference. Grasshopper B is within the comfort zone; hence, it is not attracted to or repelled

by Grasshopper A owing to a neutral force. Grasshoppers that move closer to grasshopper A within the comfort zone, such as grasshopper C, repel themselves away from grasshopper A. Grasshoppers that are outside the comfort zone, such as grasshopper D, tend to be attracted to move towards grasshopper A. All the explained movements continue to be executed by a swarm of grasshopper populations until they converge on a solution.

A mathematical formulation can be used to represent the natural behavior of the grasshopper's movement in a swarm. GOA's main equation can be expressed based on the position of the grasshopper given by [37],

$$X_i = c\left(\sum_{j=1, j\neq i}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j^d - x_i^d}{d_{ij}}\right) + T \quad (1)$$

where X_i refers to the position of the i-th grasshopper in all dimensions, and D is the dimensionality of the search space, where d=1,2,⋯, D. x_i^d and x_j^d are the current positions of the i-th and j-th grasshoppers in the d-th dimension, respectively. Where T denotes the best solution currently available. Equation (1) involves two terms: The first term involves a summation term in the bracket multiplied by c, and the second term is T. The first term considers the position of the other grasshoppers and implements their interaction in the natural environment within an area specified by ub_d and lb_d, representing the upper and lower bounds in the search space in the d-th dimension, respectively. D_{ij} is the distance between the two grasshoppers the i-th and j-th and is calculated using $\left|x_j^d - x_i^d\right|$. Parameter s in the first term represents the strength of the social forces between two grasshoppers, given by [37]:

$$s(r) = fe^{-r/l} - e^{-r} \quad (2)$$

Where f and l are the attraction intensity and attraction length scale, respectively, and their best values in the original GOA are f=0.5, l=1.5, and r is the normalized value of $\left|x_j^d - x_i^d\right|$ between 1 and 4, [37]. The c parameter in (1) is a monotonically decreasing coefficient that reduces linearly with every iteration and plays a vital role in GOA exploration and exploitation, as it controls the shrinking and expansion of grasshoppers' comfort, repulsive, and attraction zones. In the original GOA, a balance between exploration and exploitation was implemented by linearly decreasing the value of parameter c using the following equation, [37]:

$$c = c_{\max} - iter\frac{c_{\max} - c_{\min}}{L} \quad (3)$$

where $c_{\max}$ is the maximum value, $c_{\min}$ is the minimum value, *iter* is the current iteration, and L is the maximum number of iterations. The second term in (1) simulates grasshoppers' tendency to move toward the food source. Iterating (1) for all grasshoppers yields a converged solution.

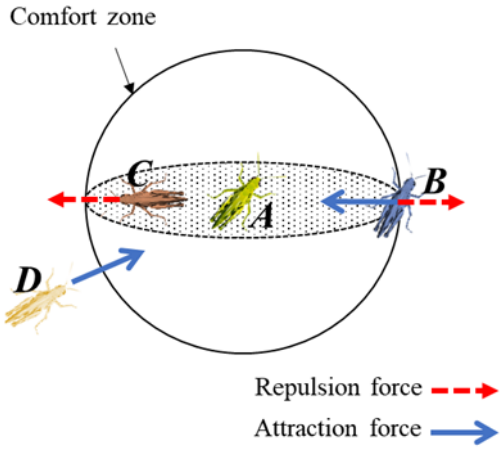

Fig. 1: Primitive corrective patterns among individuals in a swarm of grasshoppers

3 Adaptive Average Grasshopper Optimization Algorithm (AAGOA)

First, AAGOA implements a nonlinear convergence parameter to balance the exploration and exploitation phases. Second, the best solution found thus far in (1) was replaced with the adaptive average value calculated from the current and previous best solutions. Both modifications were implemented concurrently using the modified version of equation (1).

3.1 Modified Parameter Convergence Value

Parameter c in the original GOA starts from a value close to unity at the beginning of the iteration and decreases linearly to a s mall constant value to support the exploitation phase. However, using a linear decrement of the c value would cause the algorithm to converge to a solution too quickly and may miss possible optimal solutions in the search space during the exploration phase.

To overcome this limitation, the version of the GOA variant proposed in this study, AAGOA, adopts a n onlinear decreasing value called c_m. Following the studies of [38], [39], this study adopted a nonlinear approach to the c_m parameter value. The value of the c_m parameter starts at unity,

similar to that of c in the original GOA. However, instead of using a single equation to calculate the c_m parameter value with increasing iterations, the calculation of the c_m parameter depends on the following three conditions.

$$c_m = \begin{cases} \alpha, & 0 < iter \le 0.6 \cdot L \\ 0.1, & 0.6 \cdot L < iter < 0.9 \cdot L \\ 0.05, & 0.9 \cdot L < iter < L \end{cases} \quad (4)$$

where

$$\alpha = \begin{cases} \exp\left(-0.5\left(\dfrac{iter}{L/\sigma}\right)^2\right), & \text{if } \alpha \ge 0.1 \\ 0.1, & \text{otherwise} \end{cases} \quad (5)$$

Equation (1) is updated using c_m, and is represented as follows:

$$X_i = c_m\left(\sum_{j=1, j\neq i}^{N} c_m \frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j^d - x_i^d}{d_{ij}}\right) + T \quad (6)$$

In the proposed modification, the nonlinear c_m changes in three stages. In the first stage, which occurred for the first 60% of the maximum iterations, the c_m value of every iteration was changed based on (4) and (5). In these equations, σ controls the rate of the decrease in c_m. A smaller σ value will decrease the decrement rate, causing the c_m value to remain high during the earlier executions of the algorithm. Conversely, a large value of σ causes a steeper change in c_m. The selection of the σ value depends on the nature of the optimization problem, but a typical value between 3 and 5 produces acceptable results in most optimization problems, [38]. Based on trial-and-error simulation runs, this study used σ = 4 to produce optimum results. During the first stage, the value of c_m is set to 0.1 if the calculation produces a c_m value less than or equal to 0.1. The second stage occurs between 60% and 90% of the maximum number of iterations. During this phase, c_m is set to a constant value of 0.1. The third and last stages occurred at more than 90% of the maximum iteration, where the value of c_m was set to a constant of 0.05.

Fig. 2 compares the changes in the c value obtained using the original GOA and the proposed nonlinear c_m method applied to AAGOA. Based on this graph, the value of c_m in the proposed method supports more exploration during the first 50 iterations. Subsequently, as it approaches 250 iterations, the c_m values converge rapidly to the target position. As it reaches a steady position, the constant value in the second stage allows the algorithm to search globally within the entire space. During the last stage, the search space becomes narrower because of the smaller c_m value. This provides an opportunity for the algorithm to search intensely around the local optimal solution position. This signifies a more focused exploitation stage. Therefore, by using nonlinear c_m equations, the algorithm should be able to improve its exploration and exploitation. These changes can also accelerate the convergence of the algorithm.

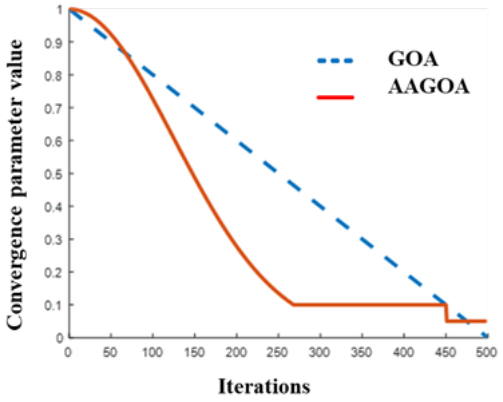

Fig. 2: Variation in convergence parameter values for the GOA and AAGOA

3.2 Adaptive Average for Target Fitness

In the original GOA, the i-th grasshopper position is calculated using Equation (1). Currently, the equation works by adding the current best position, T, found throughout the dimensional space. The second proposed improvement uses the predicted position, P, instead of T. Hence, (1) can be modified as follows:

$$X_i = c_m\left(\sum_{j=1, j\neq i}^{N} c_m \frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j^d - x_i^d}{d_{ij}}\right) + P \quad (7)$$

where the predicted position can be calculated using the following formula:

$$P = T + \hat{A} \quad (8)$$

where $\hat{A}$ is the unit difference between the current best position and the mean of the previous k number of T values represented by $\overline{T_w}$ and can be calculated using the following equation:

$$\hat{A} = \frac{T - \overline{T_w}}{\left|T - \overline{T_w}\right|} \quad (9)$$

where

$$\overline{T_w} = \frac{1}{k}\sum_{i=n-k+1}^{n} T_i \quad (10)$$

Variable n in (10) represents the total number of T

currently recorded. Equations (7) to (10) are executed if the total number of recorded best positions is equal to or greater than *k*. Otherwise, the value of the predicted position is set to the current best position.

The purpose of using the predicted position is to improve the exploitation process by considering past search trends and providing a possible target position that is better than the current position. However, considering different types of optimization problems, using only the predicted position in certain types of problems will not produce better results than using the current best position, as in the original GOA. Hence, the best way to obtain benefits from using both the predicted position and the current best position is to interchange both methods using probability. Based on the trial-and-error method, the best results were achieved when the change was implemented using 0.1 probability, i.e., a 10% chance of using *P* instead of T when calculating the grasshopper's next position.

3.3 AAGOA Implementation

The pseudocode for the proposed AAGOA is presented in Algorithm 1. Table 1 lists the parameter settings used to simulate these algorithms. These parameters were based on the original GOA research to test the performance of the algorithm, [37]. Additional parameter values, such as σ and *k*, are specific to the proposed modification of the GOA.

Algorithm 1 Pseudocode for AAGOA

1. Generate the initial population of Grasshoppers $X_i (i = 1,2,\cdots,n)$ randomly with a selected D value.
2. Set parameter setting values
3. Set probability value of adaptive average implementation
4. Evaluate the fitness $f(X_i)$ of each grasshopper X_i.
5. T = best solution
6. **while** $(\boldsymbol{iter < L})$ **do**
7. Update c_m using equations (4) and (5)
8. **for** $\boldsymbol{i = 1}$ to $\boldsymbol{N}$ (all N grasshoppers in the population) **do**
9. Normalize the distance between grasshoppers
10. Calculate the possibility with a 10% chance of executing equation (7)
11. **if** the possibility is fulfilled **then**
12. Update the position of the current search agent using equation (7)
13. **else**
14. Update the position of the current search agent using equation (6)
15. **end if**
16. Bring back the current search agent if it goes outside the boundaries
17. **end for**
18. Update T if there is a better solution
19. Update record of T
20. **if** the number of records T is larger than *k*
21. Calculate the predicted position using equation (8)
22. **else**
23. Predicted position = T
24. **end if**
25. $\boldsymbol{iter = iter + 1}$
26. **end while**
27. Return the best solution of T

Table 1. Parameter settings of selected algorithms

Algorithm	Parameter setting
GOA	$N = 30$, $L = 500$ $c_{max} = 1$, $c_{min} = 0.00004$, $f = 0.5$, $l = 1.5$
AAGOA	$N = 30$, $L = 500$, $c_{max} = 1$, $c_{min} = 0.00004$, $f = 0.5$, $l = 1.5$, $\sigma = 4$, $k = 10$

The procedure of the algorithm starts with the initialization of X_i, and settings of the parameter values for, L, c_{max}, c_{min}, and σ. One run loop iterates the entire procedure for a preset number of maximum iterations while updating the c_m value using equations (4) and (5) and the current best position.

Referring to the pseudocode, the nested repetition loop involves the execution of equations (1) and (2) for each grasshopper in the population. Probability is calculated to determine which equation will be implemented during a particular

iteration. The best solution was updated at the end of each repetition loop. At the beginning of the iteration, the total number of recorded T was less than k. Therefore, the value of P was assigned to the value of T. The process stops after the execution of the 500-th iterations, the maximum preset iteration number for each simulation run. This maximum number of iterations followed most previously published studies. Thirty grasshopper agents were used for each simulation. In addition, 30 independent simulations were run for every parameter setting, and all runs used 20 dimensions, that is D=20.

4 Performance Evaluation

All algorithms in this work were implemented and executed on MATLAB R2021a using a workstation with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz processor and 32GB RAM. The performance of the proposed AAGOA is assessed in this section using four experiments. The first experiment evaluated the GOA and AAGOA based on the average values, standard deviations, and best values using ten standard benchmark functions for five different operators and generated a total of 50 standard functions. The second test strictly tested the convergence performance of the AAGOA and GOA. The third experiment aimed to test the AAGOA using Wilcoxon's nonparametric ranking test and Friedman's ranking test. The fourth test presents the complexity of the AAGOA.

4.1 CEC2021 Benchmark Test Functions

In this study, CEC2021 benchmark functions were employed to test the efficacy of the proposed AAGOA over its original algorithm. Table 2 presents the CEC2021 benchmark function suites. There are ten functions in CEC2021 with four different types: Function F1 represents unimodal functions, functions F2 to F4 represent basic functions, and functions F5 to F7 represent hybrid functions. Finally, functions F8 to F10 represent composition functions. where Nf represents the number of basic functions forming a particular function. The search space is defined by the upper and lower bounds defined as [-100,100] D, where D represents the dimension number that can be used with the function. These benchmark functions of CEC2021 are single-objective bounds constrained by transformations in bias, shift, and rotation.

Table 2. Summary of CEC2021 bound-constrained real-parameter benchmark functions

Types	No	Functions
Unimodal Functions	F1	Shifted and Rotated Bent Cigar Function
Basic Functions	F2	Shifted and Rotated Schwefel's Function
	F3	Shifted and Rotated Lunacek bi-Rastrigin Function
	F4	Expanded Rosenbrock's plus Griewangk's Function
Hybrid Functions	F5	Hybrid Function 1 (Nf = 3)
	F6	Hybrid Function 2 (Nf = 4)
	F7	Hybrid Function 3 (Nf = 5)
Composition Functions	F8	Composition Function 1 (Nf = 3)
	F9	Composition Function 2 (Nf = 4)
	F10	Composition Function 3 (Nf = 5)
Search range: [-100,100] D, D = 10, D = 20		

Because the operators parameterize the benchmark functions, this suggests that the proposed algorithm can be evaluated by testing it with various possible configurations of operators on all benchmark test functions. Different transformations of bias, shift, and rotation can be represented by binary parameters, with 1 indicating activated and 0 indicating deactivation. The investigated transformations were (000), (010), (011), (100), and (110). For each parameter setting and transformation, the results were evaluated based on the mean of the best values from the 30 independent trial runs. The best algorithm is the one that is able to produce the lowest mean results for most of the benchmark functions.

4.2 Experimental Results

The benchmark functions for the unimodal, basic, hybrid, and composition functions use 30 search agents over 500 iterations. The presented results were recorded based on 30 independent trials with random initial conditions to calculate statistical results. These results include the mean fitness (mean), which represents the average performance and reliability of the algorithm; the standard deviation of fitness (std), which represents the stability of the algorithm; and best fitness (best), which represents the best optimization ability of the algorithm.

The CEC2021 benchmark test functions were optimized using the proposed AAGOA and the original GOA. The optimization results for all compared algorithms on all benchmark test functions are presented in Table 3, Table 4, Table 5, Table 6, and Table 7. To present a qualitative

evaluation of the proposed algorithm, the convergence curves of each benchmark function type for the (000) and (111) transformations are presented in Fig. 3, Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9 and Fig. 10.

There is only one extreme point in the unimodal test function F1. This type of function is suitable for benchmarking the exploitation behavior of the algorithm. Based on Table 3 until Table 7, AAGOA outperforms GOA in terms of best, mean, and std values for all different types of transformations, indicating reliable and better performance compared to GOA with better stability. Fig. 3 and Fig. 7 show that AAGOA can converge quickly compared to GOA, especially during the iteration where the convergence parameter c_m is steeper. Based on this result, it can be concluded that AAGOA has good exploitation capability. The main reason for the better results compared to the GOA is the adaptive average for target fitness which provides a better target position.

The basic functions, F2 to F4 in CEC2021, have several local optima that are used to evaluate the exploration ability. The results in Table 3 until Table 7 show that AAGOA works best in 10 out of the 15 benchmark tests. Although the AAGOA does not achieve the best results in the remaining five benchmark functions, most of the results were close to the GOA. Fig. 4 shows that the AAGOA is able to quickly converge towards a better fitness position compared to the GOA. A similar result is shown in Fig. 8; however, in the beginning, the GOA had better results than AAGOA. As c_m becomes steeper and with the use of an adaptive average to predict better target fitness, the AAGOA convergence capability significantly increases. These results indicate that AAGOA has a competitive exploration ability.

Hybrid function, F5 to F7 consists of combinations of basic functions that are unimodal or multimodal. It can be used to evaluate the performance of both the exploitation and exploration of the algorithm. The results in Table 3 until Table 7 show that the AAGOA performs better than the GOA. AAGOA outperformed GOA in terms of the best, mean, and std values for all different types of transformation, indicating reliable and better performance compared to GOA. AAGOA did not achieve good std values in functions F6(010) and F6(110), but the difference compared to the GOA std values was close. The convergence curves in Fig. 5 and Fig. 9 show that the AAGOA converges faster than the GOA. It was concluded that AAGOA has good exploitation and exploration for hybrid functions.

The ability of an algorithm to avoid local optima can be evaluated using the composition functions F8 to F10. They are suitable for benchmarking exploration and exploitation simultaneously for a large number of local optima. The AAGOA results in Table 3 until Table 7 manage to achieve better results than the GOA in five out of 15 functions. Although it was less than half of the functions, the difference between the GOA and AAGOA results was small. Fig. 6 and Fig. 10 indicate that AAGOA still converges better than GOA despite having a higher fitness value in F8(000) compared to GOA in the early iteration, but AAGOA significantly produces better fitness values with increasing iterations. Based on these results, AAGOA has acceptable exploitation and exploration capabilities for composition functions.

Table 3. Optimized results for (000) transformation

Functions	Criteria	AAGOA	GOA
F1	**best**	**1.27E+03**	1.02E+05
	mean	**3.88E+03**	6.19E+05
	std	**2.73E+03**	4.91E+05
	rank	1	2
F2	**best**	**1.45E+03**	1.63E+03
	mean	**2.43E+03**	2.63E+03
	std	**4.35E+02**	5.38E+02
	rank	1	2
F3	**best**	7.26E+01	**4.31E+01**
	mean	1.25E+02	**9.88E+01**
	std	**3.37E+01**	3.60E+01
	rank	2	1
F4	**best**	**2.45E+00**	3.21E+00
	mean	**5.66E+00**	7.43E+00
	std	**2.07E+00**	2.53E+00
	rank	1	2
F5	**best**	**4.40E+03**	3.06E+04
	mean	**5.39E+04**	3.53E+05
	std	**9.31E+04**	2.09E+05
	rank	1	2
F6	**best**	**7.95E+01**	2.09E+02
	mean	**5.15E+02**	6.39E+02
	std	**2.61E+02**	2.78E+02
	rank	1	2
F7	**best**	**1.68E+03**	5.31E+03
	mean	**1.44E+04**	8.77E+04
	std	**2.47E+04**	7.54E+04
	rank	1	2
F8	**best**	**2.69E+02**	3.95E+02
	mean	**1.63E+03**	1.80E+03
	std	**6.91E+02**	9.09E+02
	rank	1	2
F9	**best**	**3.65E+00**	5.83E+00
	mean	3.81E+01	**1.73E+01**
	std	2.92E+01	**1.49E+01**
	rank	2	1
F10	**best**	**5.05E+01**	5.10E+01
	mean	7.71E+01	**7.02E+01**
	std	1.73E+01	**1.46E+01**
	rank	2	1

Table 4. Optimized results for (010) transformation

Functions	Criteria	AAGOA	GOA
F1	**best**	**1.77E+03**	2.88E+04
	mean	**1.21E+04**	5.18E+05
	std	**7.24E+03**	4.30E+05
	rank	1	2
F2	**best**	**1.40E+03**	1.45E+03
	mean	**2.41E+03**	2.49E+03
	std	**4.70E+02**	6.13E+02
	rank	1	2
F3	**best**	**6.03E+01**	7.62E+01
	mean	1.27E+02	**1.14E+02**
	std	3.48E+01	**2.82E+01**
	rank	2	1
F4	**best**	2.85E+00	**2.72E+00**
	mean	**6.79E+00**	7.08E+00
	std	4.17E+00	**2.47E+00**
	rank	1	2
F5	**best**	**3.20E+03**	1.05E+04
	mean	**1.09E+05**	6.44E+05
	std	**1.45E+05**	6.04E+05
	rank	1	2
F6	**best**	**2.09E+02**	3.36E+02
	mean	**6.66E+02**	6.88E+02
	std	2.67E+02	**2.25E+02**
	rank	1	2
F7	**best**	**4.16E+03**	9.40E+03
	mean	**4.77E+04**	2.25E+05
	std	**4.66E+04**	1.93E+05
	rank	1	2
F8	**best**	**1.00E+02**	1.03E+02
	mean	**7.79E+02**	9.30E+02
	std	**1.19E+03**	1.45E+03
	rank	1	2
F9	**best**	4.49E+02	**4.44E+02**
	mean	**5.06E+02**	5.12E+02
	std	**5.71E+01**	5.82E+01
	rank	1	2
F10	**best**	4.86E+02	**4.21E+02**
	mean	5.07E+02	**5.00E+02**
	std	3.14E+01	**2.98E+01**
	rank	2	1

Table 5. Optimized results for (011) transformation

Functions	Criteria	AAGOA	GOA
F1	**best**	**1.40E+03**	1.82E+05
	mean	**5.51E+03**	9.28E+05
	std	**3.99E+03**	8.48E+05
	rank	1	2
F2	**best**	**1.39E+03**	1.79E+03
	mean	**2.46E+03**	2.60E+03
	std	5.49E+02	**4.77E+02**
	rank	1	2
F3	**best**	7.63E+01	**5.92E+01**
	mean	**1.26E+02**	1.27E+02
	std	4.18E+01	**3.51E+01**
	rank	1	2
F4	**best**	3.54E+00	**2.85E+00**
	mean	9.93E+00	**7.32E+00**
	std	6.06E+00	**2.42E+00**
	rank	2	1
F5	**best**	**3.92E+03**	5.51E+04
	mean	**1.41E+05**	4.80E+05
	std	**1.39E+05**	4.37E+05
	rank	1	2
F6	**best**	**1.62E+02**	1.62E+02
	mean	**6.28E+02**	7.54E+02
	std	**2.68E+02**	2.86E+02
	rank	1	2
F7	**best**	**2.70E+03**	1.20E+04
	mean	**5.35E+04**	1.28E+05
	std	**6.60E+04**	1.39E+05
	rank	1	2
F8	**best**	**1.00E+02**	1.04E+02
	mean	1.83E+03	**1.78E+03**
	std	1.58E+03	**1.51E+03**
	rank	2	1
F9	**best**	4.38E+02	**4.27E+02**
	mean	4.97E+02	**4.94E+02**
	std	**4.46E+01**	5.90E+01
	rank	2	1
F10	**best**	4.02E+02	**4.01E+02**
	mean	4.60E+02	**4.47E+02**
	std	**3.09E+01**	3.61E+01
	rank	2	1

Table 6. Optimized results for (110) transformation

Functions	Criteria	AAGOA	GOA
F1	**best**	**1.77E+03**	2.88E+04
	mean	**1.21E+04**	5.18E+05
	std	**7.24E+03**	4.30E+05
	rank	1	2
F2	**best**	**1.40E+03**	1.45E+03
	mean	**2.41E+03**	2.49E+03
	std	**4.70E+02**	6.13E+02
	rank	1	2
F3	**best**	**6.03E+01**	7.62E+01
	mean	1.27E+02	**1.14E+02**
	std	3.48E+01	**2.82E+01**
	rank	2	1
F4	**best**	2.85E+00	**2.72E+00**
	mean	**6.79E+00**	7.08E+00
	std	4.17E+00	**2.47E+00**
	rank	1	2
F5	**best**	**3.20E+03**	1.05E+04
	mean	**1.09E+05**	6.44E+05
	std	**1.45E+05**	6.04E+05
	rank	1	2
F6	**best**	**2.09E+02**	3.36E+02
	mean	**6.66E+02**	6.88E+02
	std	2.67E+02	**2.25E+02**
	rank	1	2
F7	**best**	**4.16E+03**	9.40E+03
	mean	**4.77E+04**	2.25E+05
	std	**4.66E+04**	1.93E+05
	rank	1	2
F8	**best**	**1.00E+02**	1.03E+02
	mean	**7.79E+02**	9.30E+02
	std	**1.19E+03**	1.45E+03
	rank	1	2
F9	**best**	4.49E+02	**4.44E+02**
	mean	**5.06E+02**	5.12E+02
	std	**5.71E+01**	5.82E+01
	rank	1	2
F10	**best**	4.86E+02	**4.21E+02**
	mean	5.07E+02	**5.00E+02**
	std	3.14E+01	**2.98E+01**
	rank	2	1

Table 7. Optimized results for (111) transformation

Functions	Criteria	AAGOA	GOA
F1	**best**	**1.40E+03**	1.82E+05
	mean	**5.51E+03**	9.28E+05
	std	**3.99E+03**	8.48E+05
	rank	1	2
F2	**best**	**1.39E+03**	1.79E+03
	mean	**2.46E+03**	2.60E+03
	std	5.49E+02	**4.77E+02**
	rank	1	2
F3	**best**	7.63E+01	**5.92E+01**
	mean	**1.26E+02**	1.27E+02
	std	4.18E+01	**3.51E+01**
	rank	1	2
F4	**best**	3.54E+00	**2.85E+00**
	mean	9.93E+00	**7.32E+00**
	std	6.06E+00	**2.42E+00**
	rank	2	1
F5	**best**	**3.92E+03**	5.51E+04
	mean	**1.41E+05**	4.80E+05
	std	**1.39E+05**	4.37E+05
	rank	1	2
F6	**best**	**1.62E+02**	1.62E+02
	mean	**6.28E+02**	7.54E+02
	std	**2.68E+02**	2.86E+02
	rank	1	2
F7	**best**	**2.70E+03**	1.20E+04
	mean	**5.35E+04**	1.28E+05
	std	**6.60E+04**	1.39E+05
	rank	1	2
F8	**best**	**1.00E+02**	1.04E+02
	mean	1.83E+03	**1.78E+03**
	std	1.58E+03	**1.51E+03**
	rank	2	1
F9	**best**	4.38E+02	**4.27E+02**
	mean	4.97E+02	**4.94E+02**
	std	**4.46E+01**	5.90E+01
	rank	2	1
F10	**best**	4.02E+02	**4.01E+02**
	mean	4.60E+02	**4.47E+02**
	std	**3.09E+01**	3.61E+01
	rank	2	1

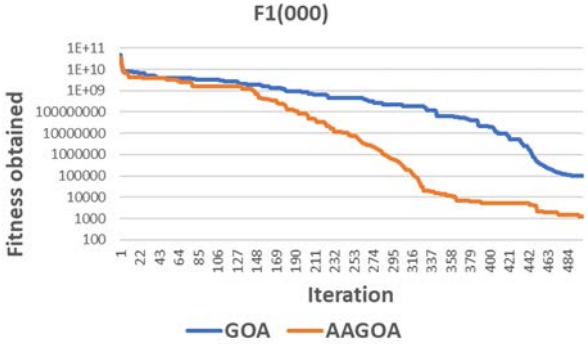

Fig. 3: Convergence curves for unimodal function F1(000)

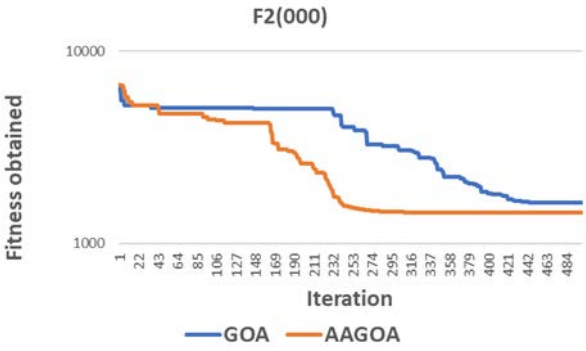

Fig. 4: Convergence curves for basic function F2(000)

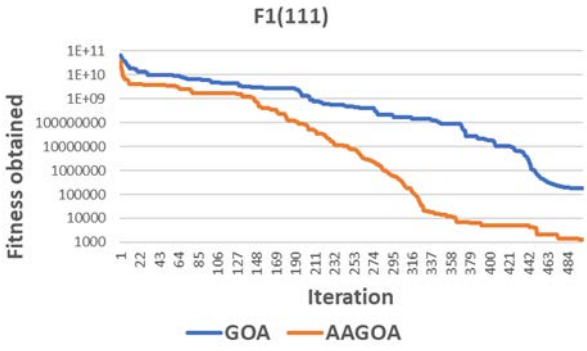

Fig. 6: Convergence curves for unimodal function F1(111)

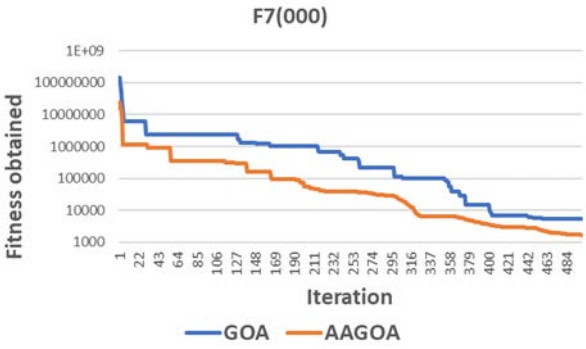

Fig. 5: Convergence curves for hybrid function F7(000)

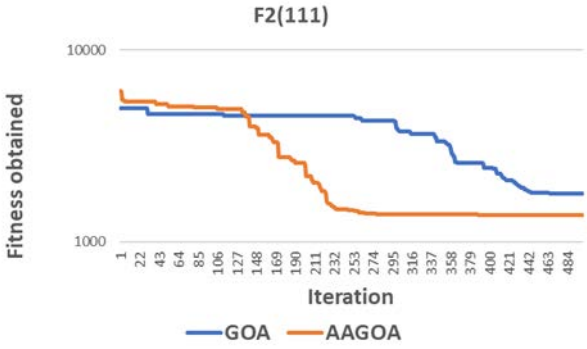

Fig. 7: Convergence curves for basic function F2(111)

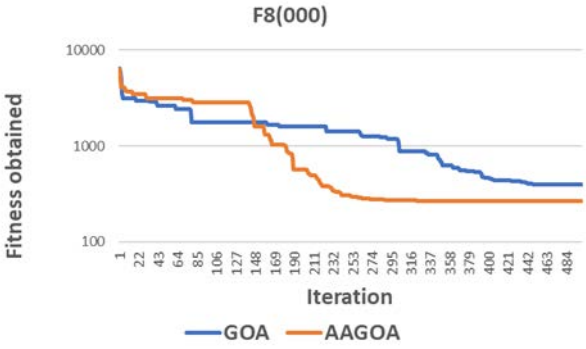

Fig. 6: Convergence curves for composition function F8(000)

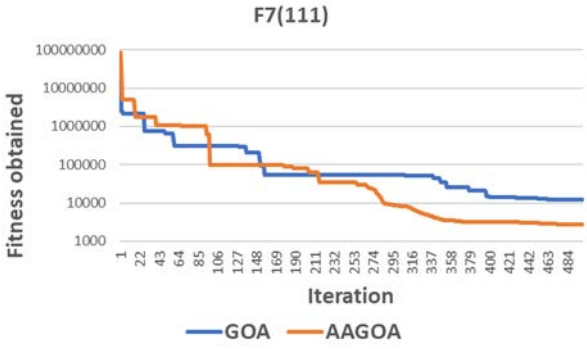

Fig. 8: Convergence curves for hybrid function F7(111)

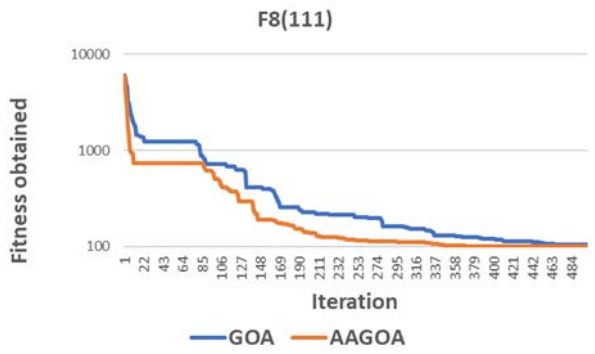

Fig. 10: Convergence curves for composition function F8(111)

In addition to evaluating the statistical performance based on the best, mean, and std results using benchmark functions, non-parametric multiple comparisons were used to further verify the validity of the results using the lowest mean value for all transformations. Two statistical analyses were used: the Friedman ranking test and Wilcoxon signed-rank test. The Friedman ranking test ranks the algorithms from best to worst. The best algorithm received the lowest rank, while the worst algorithm received the highest rank. The Wilcoxon signed-rank test with a 95% confidence interval was used to validate the significance of the improvement provided by the proposed algorithm.

The rank values for each benchmark function obtained using the Friedman ranking test are listed in Table 3 until Table 7. Table 8, Table 9, Table 10, Table 11, and Table 12 display the results for the Wilcoxon signed-rank test, and Table 13 together with Table 14 display a summary result for the Friedman ranking test and Wilcoxon signed-rank test, respectively, based on the various transformations.

From the statistical results in Table 13, it is clear that AAGOA performs best with a Friedman ranking test sum ranking value of 65, compared to GOA with a sum ranking value of 85. AAGOA ranked better for F1 to F7 for most transformations, with lesser results for F8 to F10.

The mean and standard deviations are only used to compare the overall performance of the algorithm. Wilcoxon signed-rank test was used to prove that the results were statistically significant. Based on the Wilcoxon signed-rank test, the AAGOA can improve 19 out of 50 benchmark functions while maintaining 29 similar results to the original GOA. Although AAGOA had two lesser results compared to the original GOA, the improvements were significantly greater. Based on the Wilcoxon signed-rank test, the improvements provided by AAGOA in terms of best, mean, and std with better ranking using the Friedman ranking test were statistically significant.

Table 8. Wilcoxon signed-rank test results for (000) transformation.

Functions	Criteria	AAGOA
F1	*p*-values	1.73E-06
	h	+
F2	*p*-values	1.16E-01
	h	=
F3	*p*-values	1.25E-02
	h	-
F4	*p*-values	2.26E-03
	h	+
F5	*p*-values	2.88E-06
	h	+
F6	*p*-values	6.83E-03
	h	+
F7	*p*-values	2.16E-05
	h	+
F8	*p*-values	2.80E-01
	h	=
F9	*p*-values	1.04E-03
	h	-
F10	*p*-values	9.78E-02
	h	=

Table 9. Wilcoxon signed-rank test results for (010) transformation.

Functions	Criteria	AAGOA
F1	*p*-values	1.73E-06
	h	+
F2	*p*-values	4.65E-01
	h	=
F3	*p*-values	1.11E-01
	h	=
F4	*p*-values	1.85E-01
	h	=
F5	*p*-values	1.36E-05
	h	+
F6	*p*-values	9.92E-01
	h	=
F7	*p*-values	1.73E-06
	h	+
F8	*p*-values	2.05E-04
	h	+
F9	*p*-values	3.82E-01
	h	=
F10	*p*-values	3.29E-01
	h	=

Table 10. Wilcoxon signed-rank test results for (011) transformation.

Functions	Criteria	AAGOA
F1	*p*-values	1.73E-06
	h	+
F2	*p*-values	1.92E-01
	h	=
F3	*p*-values	3.93E-01
	h	=
F4	*p*-values	1.47E-01
	h	=
F5	*p*-values	1.13E-05
	h	+
F6	*p*-values	9.78E-02
	h	=
F7	*p*-values	2.60E-05
	h	+
F8	*p*-values	5.44E-01
	h	=
F9	*p*-values	7.97E-01
	h	=
F10	*p*-values	1.71E-01
	h	=

Table 11. Wilcoxon signed-rank test results for (110) transformation.

Functions	Criteria	AAGOA
F1	*p*-values	1.73E-06
	h	+
F2	*p*-values	4.65E-01
	h	=
F3	*p*-values	1.11E-01
	h	=
F4	*p*-values	1.85E-01
	h	=
F5	*p*-values	1.36E-05
	h	+
F6	*p*-values	9.92E-01
	h	=
F7	*p*-values	1.73E-06
	h	+
F8	*p*-values	2.05E-04
	h	+
F9	*p*-values	3.82E-01
	h	=
F10	*p*-values	3.29E-01
	h	=

Table 12. Wilcoxon signed-rank test results for (111) transformation.

Functions	Criteria	AAGOA
F1	*p*-values	1.73E-06
	h	+
F2	*p*-values	4.65E-01
	h	=
F3	*p*-values	1.11E-01
	h	=
F4	*p*-values	1.85E-01
	h	=
F5	*p*-values	1.36E-05
	h	+
F6	*p*-values	9.92E-01
	h	=
F7	*p*-values	1.73E-06
	h	+
F8	*p*-values	2.05E-04
	h	+
F9	*p*-values	3.82E-01
	h	=
F10	*p*-values	3.29E-01
	h	=

Table 13. Summary of Friedman ranking test

Operators	GOA	AAGOA
000	13	17
010	12	18
011	14	16
110	12	18
111	14	16
Sum	85	65
Rank	2	1

Table 14. Wilcoxon signed-rank test summary.

Operators	+	=	-
000	5	3	2
010	4	6	0
011	3	7	0
110	4	6	0
111	3	7	0
Sum	19	29	2

4.3 Algorithm Complexity

The complexity of the algorithms is measured by the amount of time and space required to solve a problem for a given input size. The complexities of the AAGOA and GOA algorithms were calculated using the method described by CEC2021, [35]. Table 15 shows the computational complexity of both algorithms in 20 dimensions where t_0 is the time computed by running the following codes:

$x = 0.55$
for i = 1:200000

$x = x + x; x = x / 2; x = x * x; x = sqrt(x);$
$x = \log(x); x = \exp(x); x = x/(x+2);$
end

where t_1 is the time required to execute 200,000 evaluations of the benchmark F1 in 20 dimensions. t_2 is the execution time of the algorithm for F1, using the same number of evaluations. $\widehat{t_2}$ is the mean value of t_2 over the five runs.

The procedure was applied to both the GOA and AAGOA. It can be observed in the last row of the table that the computational cost of AAGOA is higher than that of GOA. This is due to the implementation of an adaptive average for the target fitness that adds to the complexity of the AAGOA. Although having a higher computational cost, AAGOA can produce significantly better results than GOA.

Table 15. Computational complexity of AAGOA and GOA for 20 dimensions

Variable	AAGOA	GOA
t_0	8.49E-03	8.49E-03
t_1	2.44E+02	2.44E+02
$\widehat{t_2}$	7.67E+03	3.93E+03
$(\widehat{t_2} - t_1)/t_0$	8.74E+05	4.34E+05

5 Conclusion

This study proposes a modified version of the GOA, referred to as AAGOA. The proposed improvements to the GOA introduced a nonlinear parameter convergence value and an adaptive average for the target fitness. Based on the optimization results using the CEC2021 benchmark functions, the AAGOA produced better results than the GOA on most tested benchmark functions in different transformation combinations (i.e., bias, shift, and rotation). The nonlinear convergence parameter helps the algorithm explore the search space efficiently at the beginning of the iteration and focuses on the local optimum towards convergence. The adaptive average for the target fitness provides the capability to move away from the local optimum entrapment by introducing the predicted target fitness based on the previous target fitness. Future research should focus on reducing the complexity of the AAGOA and implementing the modification performed in AAGOA with other GOA variants to further improve the optimization results using improved benchmark functions.

References:

[1] X. S. Yang, "Nature-Inspired Optimization Algorithms: Challenges and Open Problems", *Journal of Computational Science*, Vol.46, 2020, pp. 101104.

[2] X. S. Yang, M. Karamanoglu, "Nature-Inspired Computation and Swarm Intelligence", *Academic Press*, 2020, pp. 3-18.

[3] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *Eur. J. Oper. Res.*, Vol. 113, No. 2, Mar. 1999, pp. 469–499.

[4] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, Vol. 24, No. 11, Nov. 1997, pp. 1097–1100.

[5] T. Stützle, R. Ruiz, "Iterated Local Search." In: Martí, R., Pardalos, P., Resende, M. (eds) *Handbook of Heuristics*. Springer, 2018.

[6] Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "Grasshopper optimization algorithm: theory, variants, and applications," *IEEE Access*, 9, 2021, pp. 50001–50024.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, 1995, pp. 39–43.

[8] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in P. Melin, O., Castillo, L. T., Aguilar, J. Kacprzyk, and W. Pedrycz, (eds) *Foundations of Fuzzy Logic and Soft Computing (Lecture Notes in Computer Science)*, Vol. 4529, Berlin, Germany: Springer, 2007.

[9] L. Wang, X. Zhang and X. Zhang, "Antenna Array Design by Artificial Bee Colony Algorithm With Similarity Induced Search Method," in *IEEE Transactions on Magnetics*, Vol. 55, No. 6, June 2019, pp. 1-4.

[10] E. S. Ali, "Power System Stabilizers Design Using BAT Algorithm," *WSEAS Transactions on Systems and Control*, Vol. 17, 2022 , pp. 466-476.

[11] A. Colorni, M Dorigo, V. Maniezzo "Distributed optimization by ant colonies" in *Proceedings of the first European conference on artificial life*, 1991. pp. 134–42

[12] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA:Agravitational search algorithm," *Information Sciences*, Vol. 179, No. 13, Jun. 2009, pp. 2232–2248.

[13] A. Kaveh and T. Bakhshpoori, "Water evaporation optimization: A novel physically inspired optimization algorithm," *Computers & Structures*, Vol. 167, Apr. 2016, pp. 69–85.

[14] A. Kaveh and A. Dadras, "A novel meta-heuristic optimization algorithm: Thermal exchange optimization," *Advances in Engineering Software*, Vol. 110, A ug. 2017, pp. 69–84.

[15] J. H. Holland, "Genetic algorithms," *Scientific American*, Vol. 267, No. 1, 1992, pp. 66–73.

[16] Saad Babesse, Fouad Inel, "Optimized Genetic Algorithms Reduced Order Model Based RST Roll Control of Antiroll Bar Dedicated to Semi-active Suspension," *WSEAS Transactions on Systems and Control*, vol. 17, 2022, pp. 504-514.

[17] H. G. Beyer and H. P. Schwefel, "Evolution strategies–A comprehensive introduction," *Natural Computing*, Vol. 1, No. 1, 2002, pp. 3–52.

[18] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, 1997, pp. 341–359.

[19] Z. Zhang, Y. Cai and D. Zhang, "Solving Ordinary Differential Equations With Adaptive Differential Evolution," in *IEEE Access*, Vol. 8, 2020, pp. 128908-128922.

[20] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, Vol. 43, No. 3, Mar. 2011, pp. 303–315.

[21] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proc. IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 4661–4667.

[22] Z.W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, Vol. 76, No. 2, Feb. 2001, pp. 60–68.

[23] N. Lynn, and P. N. Suganthan, "Ensemble particle swarm optimizer," *Applied Soft Computing*, Vol. 55, 2017, pp.533–548.

[24] D. H. Wolpert, W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, Vol. 1, No. 1, 1997, pp. 67–82.

[25] I. Aljarah, A. M. Al-Zoubi, H. Faris, M. A. Hassonah, S. Mirjalili, and H. Saadeh, "Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm," *Cognitive Computation*, Vol. 10, No. 3, 2018, pp. 478–485.

[26] H. Baran and S. Ekinci, "Grasshopper optimization algorithm for automatic voltage regulator system," in *Proceedings of the 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, IEEE, Istanbul, Turkey, May 2018, pp. 152–156

[27] A. A. El-Fergany, "Electrical characterization of proton exchange membrane fuel cells stack using grasshopper optimizer." *IET Renewable Power Generation*, Vol. 12 No.1, 2018, pp.9–17.

[28] H. Feng, H. Ni, R. Zhao, and X. Zhu, "An enhanced grasshopper optimization algorithm to the bin packing problem," *Journal of Control Science and Engineering*, 2020, pp. 1–19.

[29] Z. Elmi, and, M.Ö. Efe, "Multi-objective grasshopper optimization algorithm for robot path planning in static environments," in *2018 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, Feb. 2018, pp. 244–249.

[30] S. Arora, and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Computing and Applications*, Vol. 31, No. 8, 2019, pp.4385–4405.

[31] J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, Z. Su, and Y. Yu, "Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm," *Aerospace Science and Technology*, Vol. 70, 2017, pp.497–510.

[32] M. M. Annie Alphonsa and N. Mohana Sundaram, "A reformed grasshopper optimization with genetic principle for securing medical data," *Journal of Information Security and Applications*, Vol. 47, Aug. 2019, pp. 410–420.

[33] B. P. Dahiya, S. Rani, and P. Singh, "Lifetime improvement in wireless sensor networks using hybrid grasshopper meta-heuristic," *Proceedings of ICRIC 2019: Recent Innovations in Computing, Springer International Publishing*, 2020, pp. 305–320.

[34] T. C. Teng, M. C. Chiang, and C. S. Yang, "A hybrid algorithm based on GWO and GOA for cycle traffic light timing optimization," in *Proc. IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy, Oct. 2019, pp. 774–779.

[35] A. Mohamed, A. Hadi, A. Mohamed, P. Agrawal, A. Kumar, P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization," *Tech. Rep.*, November 2020, [online] Available: https://github.com/P-N-Suganthan/2021-SO-BCO

[36] A.V. Latchininsky, "Locusts and remote sensing: a review," *Journal of Applied Remote Sensing*, Vol. 7, N o. 1, 2013, pp. 075099-075099.

[37] S. Saremi, S. Mirjalili, A. Lewis, "Grasshopper optimization algorithm: theory and application," *Advances in engineering software*, Vol. 105, 2017, pp. 30–47.

[38] P. Mishra, V. Goyal, and A. Shukla, "An improved grasshopper optimization algorithm for solving numerical optimization problems," in *Advances in Intelligent Computing and Communication. Lecture Notes in Networks and Systems*, Vol. 109, S pringer, Singapore, 2020, pp. 179–188.

[39] R. Zhao, H. Ni, H. Feng, and X. Zhu, "A dynamic weight grasshopper optimization algorithm with random jumping," in *Advances in Computer Communication and Computational Sciences: Proceedings of IC4S 2018*, Vol. 924, Springer, Singapore, 2019, pp. 401–413.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

-Najwan Osman-Ali carried out the investigation, formal analysis, methodology, software, and writing-original draft.

-Junita Mohamad-Saleh is responsible for supervision, project administration, writing, review, and editing.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.