Turkish Journal of Engineering
https://dergipark.org.tr/en/pub/tuje
e-ISSN 2587-1366

A novel strategy to avoid local optimum: Army-inspired genetic algorithm (AIGA)

Müslüm Kilinc[1], Emrah Atılgan[*2], Cengiz Atiş [1]

[1] Erciyes University, Department of Civil Engineering, Türkiye, kilinc@erciyes.edu.tr; cdatis@erciyes.edu.tr
[2] Eskişehir Osmangazi University, Department of Computer Engineering, Türkiye, emrah.atilgan@ogu.edu.tr

Cite this study: Kılınç, M., Atılgan, E., & Atış, C. (2024). A novel strategy to avoid local optimum: Army-inspired genetic algorithm (AIGA). Turkish Journal of Engineering, 8 (3), 436-446

https://doi.org/10.31127/tuje.1412271

Keywords
Stochastic optimization
Evolutionary algorithm
Genetic algorithm
Army-inspired strategy
Local optima

Research Article

Received: 31.12.2023
Revised: 27.02.2024
Accepted: 29.02.2024
Published: 05.07.2024

Abstract

Objective functions of which an analytical solution is very difficult or time-consuming are solved using stochastic optimization algorithms. Those optimization algorithms compute an approximate solution for objective functions. For a specific search space, the objective function might have one or more local optima along with the global optimum. When a comparison is made among the algorithms, one optimization algorithm could be more effective than others in finding a solution for certain objective functions. The most important factors affecting the success of optimization algorithms are the greatness of search space and the complexity of the objective function. Reaching the global optimum in huge search spaces is very difficult. In complex objective functions that have many local optima or where the differences between global optimum and local optima are very small, the probability of trapping into the local optimum is high. Existing optimization algorithms could be improved using the search space scanned more successfully to give a better performance. To achieve this aim, we present a novel algorithm, called Army-Inspired Genetic Algorithm (AIGA), which is inspired from military movement. The presented algorithm, apart from other optimization algorithms, searches global optima effectively by dividing the entire search area into territories instead of searching in one piece. Thus, the probability of getting trapped in a local optimum reduces and the probability of finding the global optimum increases. The presented algorithm was tested on well-known benchmark problems. The results shows that AIGA is more efficient algorithm in finding the global optimum than traditional algorithms.

1. Introduction

It is known that there are many functions used in different fields of science. Sometimes it is hard (or impossible) to obtain the exact solution of a function's optimum. Thus, optimization methods have been used for approximation of optimal solutions. Many optimization methods in the fields like engineering, computer science, business, bioinformatics, and statistics, etc., have been applied to obtain the optimal solution [1]. When it is hard to achieve the exact solution and computationally expensive, optimization methods speed up the process. Particularly, while the optimal solution is under investigation in a large search space, optimization algorithms systematically search the best solution. Based on the optimization problem, the main aim is to minimize or maximize the objective function.

Stochastic optimization techniques are the member of optimization methods which use random variables. The process is started with random initial variables and randomly searches the fitness landscape for the possible optimal solution. Over the years, many different algorithms and methods have been developed by researchers for optimization [2-6]. In the following, a brief literature review was given on the heuristic algorithms that used in this study for comparison. Grey Wolf Optimization (GWO) [7] is a metaheuristic inspired by the social structure and hunting patterns of grey wolves. GWO is particularly effective in solving optimization problems with a continuous search space. The algorithm emulates the organizational structure of a wolf pack, where the most dominant members are symbolized by alpha, beta, and delta wolves. By iteratively updating the positions of these wolves, GWO seeks to converge towards the optimal solution. Particle Swarm Optimization (PSO) [8] is a swarm intelligence technique inspired by the collective behavior of birds and fish. In PSO, a group of particles explores a search space, modifying their positions based on both individual and collective experiences. The velocity of each particle is

adapted according to its own best-known position and the overall best-known position of the entire swarm. This enables effective global exploration and exploitation during the optimization process. Firefly Algorithm (FA) [9] draws inspiration from the natural flashing patterns exhibited by fireflies. Fireflies are attracted to each other, and this attraction is used as the basis for optimization. The algorithm utilizes the brightness of fireflies to represent the objective function values. Firefly movements are then governed by attractive and repulsive forces, guiding the search towards optimal solutions in the optimization landscape. Bat Algorithm (BA) [10] draws inspiration from the echolocation behavior of bats. In this method, simulated bats navigate through the exploration space by modifying both their frequencies and loudness. The echolocation pulses determine the proximity of bats to potential solutions. Additionally, randomness is introduced to simulate the foraging behavior of bats, improving the algorithm's capacity to efficiently navigate and take advantage of the search space. Crow Search Algorithm (CSA) [11] is based on the social foraging behavior of crows. The algorithm introduces multiple flocks of crows, each representing a candidate solution. Crows share information about their positions, allowing the algorithm to balance exploration and exploitation. The algorithm incorporates a global exploration strategy to discover diverse solutions and a local exploitation mechanism for refining the search around promising regions. Cuckoo Search (CS) [12] draws inspiration from the reproductive habits of cuckoo birds, which involve depositing their eggs in the nests of different bird species, and the host birds may reject eggs that deviate from their expected characteristics. This rejection behavior is mimicked in the algorithm to generate new solutions. Cuckoo Search aims to balance exploration and exploitation by incorporating random walks and Levy flights [13] to navigate the search space efficiently. Flower Pollination Algorithm (FPA) [14] draws inspiration from the pollination process in plants, where flowers share genetic information to achieve successful reproduction. In the algorithm, flowers represent candidate solutions, and the pollination process corresponds to the exchange of information between flowers. This approach improves the algorithm's capacity to navigate the search space and move towards achieving optimal solutions.

One of the most powerful optimization algorithms is Genetic Algorithms (GAs) inspired by natural selection. GA works according to the Darwin principle: individuals best adapted to development will survive. This improvement is achieved through generations with genetic operators called crossover and mutation. Individuals newly generated using these operators are sorted based on their fitness values, the good ones are kept in the next generation, and those who cannot adapt are removed from the population. Genetic algorithms are among the most preferred search algorithms and optimization algorithms. One of the main disadvantages of GAs is getting trapped into local optima. Although much work has been done to overcome this problem [15-18], it has not yet been fully solved.

The main issue of such stochastic algorithms is largeness search space boundaries. The narrower search space results with more approximate solutions in less computational time. In such algorithms, the solution can be searched in narrower search spaces with the clustering method, however, in this case the computational time might be longer than as expected due to decrease of the population size for each clustered space. As an advantage, the clustering method prevents being trapped with a local solution. If there is no global optimum in a cluster, this will waste time and effort. Moreover, assigning individuals in the population in a non-result cluster will delay reaching the global optimum. The proposed study utilizing modified clustering method based on army movement strategy prevents idle operation in search subspaces and avoids being trapped in a local solution. Army-Inspired Genetic Algorithm (AIGA) was adapted to the traditional genetic algorithm. The efficiency of AIGA has been illustrated by solving the benchmark optimization problems given in the literature [19-20].

2. Method

The main idea of the army-inspired method is inspired from capturing a specific area with a limited number of soldiers. During a war, the army cannot attack the enemy at a single point if enemy's main base is unknown. Priority target is to locate and conquer the headquarters of the enemy. This can be achieved by scanning the whole enemy area. Therefore, the commander should split the whole region into small areas and send a certain number of pioneer soldiers equally to gather information about the quantity and location of the enemy soldiers and bases. Based on feedback information from pioneer soldiers, more soldiers are to be dispatched to the possible main base, and less soldiers are kept at the other bases. Thus, while fighting the main enemy in a region, the army is also on guard against enemy threats from other regions. If a larger enemy base is detected in another area, the army needs to attack there by dispatching more troops immediately.

In the proposed algorithm AIGA, each soldier represents an individual while the army and troops correspond to the whole population and sub-populations, respectively. As well as greater or stronger enemies are symbolized as the optimal points of optimization problems and greatness of these enemies are defined by the fitness function value. Whilst, main enemy base is represented by global optimum point, smaller bases correspond to local optima points. The search space defines the whole region, and the sub-search space represents the sub-regions where the troops are located. The most important criterion in this method is population number namely the number of soldiers. Depending on the size of the number of soldiers at present, either the entire area or only a limited area can be conquered.

An ordinary stochastic optimization algorithm begins with a defined search space and a certain population size. Apart from other algorithms, the proposed algorithm, AIGA, initiates by dividing the entire search space into smaller sub-search spaces. Since there is no specific information or trail available at the beginning of a search,

the population is partitioned equally into the sub-populations in similar manner for the search space partitioning. Starting from the first sub-search space to the last sub-search space, TGA runs using the number of generations, for each sub-search space and sub-population. After each certain number of generations, the sub-population sizes in each sub-search space are determined using a roulette-wheel method based on evaluation of the population. Since the total population size is constant, the population in some sub-search spaces could be increased, and some could be decreased. Thus, the search continues with a higher population in the sub-search space where the current optimum point is, while in other subspaces the search continues with less individuals. If a new current optimum point is found in a sub-search space with a smaller number of individuals, the majority of the population is directed here to reach a better point in this sub-search space. An illustration of individuals' movement is shown in Figure 1.

In this method, depending on the present situation, it is aimed to search in the sub-search spaces with the sub-population instead of using all population in the entire search space. The method aims to obtain the result in a sub-search space which may contain the highly probable solution. While the majority of the population searches in one subspace, the exploration persists across other sub-search spaces with fewer individuals to prevent becoming confined to a local optimum. The pseudocode code of AIGA is given with Algorithm 1 and 2.

Figure 1. Illustration of AIGA with 4-subspaces over generations.

Algorithm 1: Army-Inspired Genetic Algorithm (AIGA).

```
Set Parameters
Partitioning:
    All Search Space is divided into Sub-Search Spaces. (Number of Sub-Search Spaces is user-defined.)
Distribution:
    All Population is distributed into the sub-space searches. (Initially, equal distribution)
Stopping condition ← false.
while stopping condition==false
    foreach search space
        set options of population size, and initial population.
        run Genetic Algorithm (GA) (details in Alg. 2)
    end
    Gathering:
        Sub-Populations come together.
    Distribution:
        Whole population is distributed into the sub-space searches with the Roulette Wheel Method.
        (Sub-population size of every sub-space search is determined based on current best fitness value.)
    Check if the stopping condition is true.
end
```

Algorithm 2. Genetic Algorithm (GA).

```
Set Initial Population
Evaluation of Fitness Values
Sorting of Fitness Values
Termination criteria ← false.
while Termination criteria is not met
  foreach generation (Generation Number is optional)
    Selection: (Method is optional, Tournament, Roulette Wheel, etc.)
    Crossover: (Method is optional, One point, two points, multi-points, arithmetic, uniform, etc.)
    Mutation: (Mutation Rate, Population Size, Mutation Limits, etc. are optional)
    Evaluation of Fitness Values
    Sorting of Fitness Values
  end
  Check if Termination criteria is met?
end
```

3. Results

In this study, to demonstrate the success of the proposed method, multi-modal benchmark optimization problems that contain global and local optima have been tested. Therefore, two-dimensional multimodal functions having a few local optima were employed for visuality and illustration.

When addressing an optimization challenge, various factors come into play, influencing the algorithm's overall performance. These parameters may vary depending on the algorithm. For example, the success of the conventional genetic algorithm is influenced by various factors, including but not limited to the population size, number of generations, selection method, crossover method, elite population rate, mutation rate, and mutation amount. In fact, an optimization algorithm for two different problems may show different performance with the same parameters because of the nature of the stochastic search. Under this acknowledgment, AIGA has been compared to a traditional genetic algorithm using the same parameters for a fair comparison. For this, Damavandi function [21], which is well known as one of the difficult optimization problems in the literature, and Gaussian-Like function were chosen.

For each case study, same operators and parameters were used in the traditional genetic algorithm and AIGA. For instance, the tournament method was chosen for selection operator, and the arithmetic crossover method is chosen for crossover operator. Furthermore, the traditional genetic algorithm and AIGA share identical settings for parameters such as population size, number of generations, elite population rate, mutation rate, and total number of evaluations.

AIGA is not a new algorithm, but a novel strategy that has not been proposed before. Processing AIGA for a single group corresponds to running a traditional genetic algorithm (TGA). Therefore, the sample optimization problems were processed by increasing the number of sub-spaces to show the effect of AIGA. Accordingly, the results were presented on graphics and tables.

3.1 Case Study 1: Damavandi Function

As the first case study, the multi-model Damavandi function was considered as a test function with 2-dimensions. The equation for the Damavandi function is given in Equation 1.

The boundaries of the defined function have been set as 0 and 14 for each variable, x_1 and x_2. As seen in Figure 2, the function has a local optimum along with the global optimum.

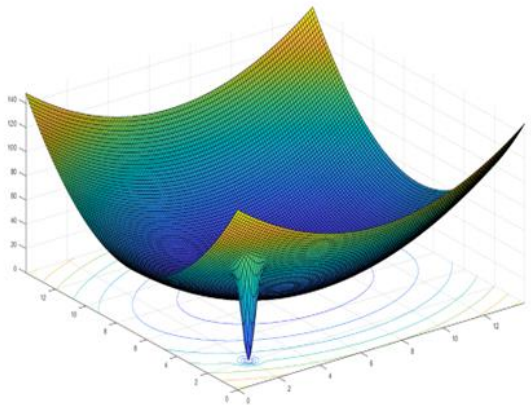

Figure 2. Damavandi function.

$$f(x) = \left[1 - \left|\frac{sin[\pi(x_1 - 2)]sin[\pi(x_2 - 2)]}{\pi^2(x_1 - 2)(x_2 - 2)}\right|^5\right][2 + (x_1 - 7)^2 + (x_2 - 7)^2] \quad (1)$$

Since the area, where the global optimum exists, has small amplitude, the solution to be obtained from any optimization algorithm is highly probable to be stuck in the local optimum. For this reason, Damavandi function is a very exceptional optimization problem to use in comparing or testing the performance of optimization algorithms.

For this optimization problem, the total number of evaluations was set to 20,000. TGA and AIGA were compared under different population sizes from 300 to 1800 by incrementing 100. In a similar manner, the number of groups as a parameter of AIGA were assigned values from 1 to 25. AIGA was run 1,000 times for each population size and the number of groups. The results obtained are presented by both in tabular form and the graphs as the success rate (%) in achieving global optimum for Damavandi function.

In Figure 3, each subplot demonstrates success rates as percentage (y-axis) value versus the number of groups (x-axis) at which AIGA runs at different population sizes on Damavandi function. As seen in Figure 3, the success rate increases (in most cases) as the number of groups increases for each population size. When comparison made between TGA (corresponds to 1-group AIGA) and AIGA (2 or more groups), the increase in success of AIGA is seen prominently as the number of groups increases.

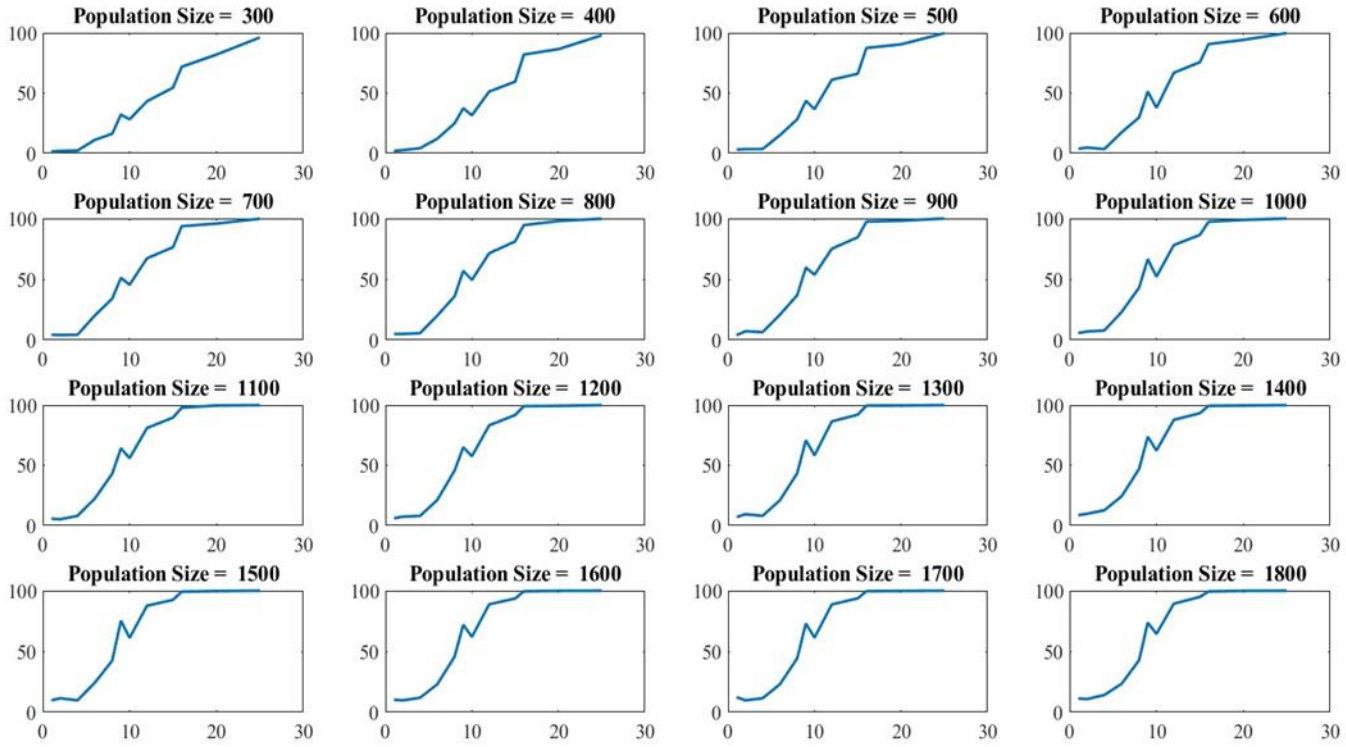

Figure 3. Success Rates (%) vs Number of Groups for Different Population Size

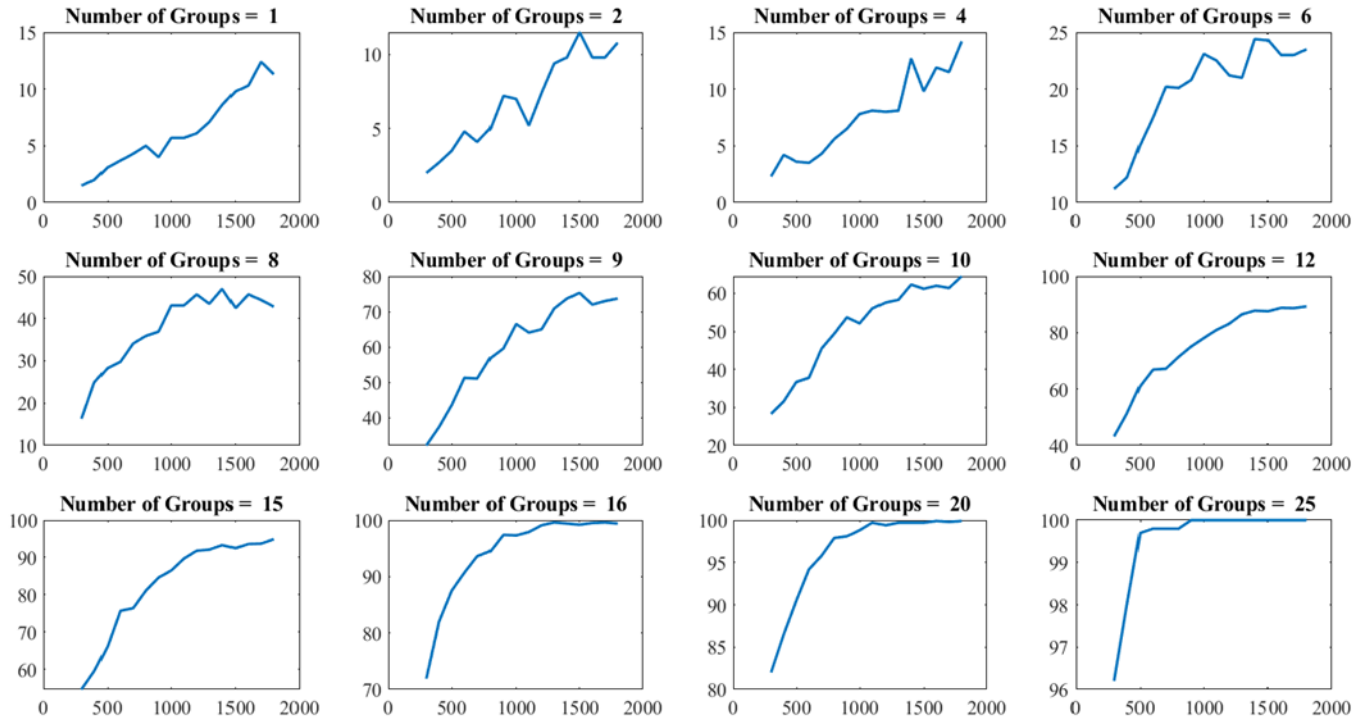

Figure 4. Success rates (%) vs population size for different number of groups.

In Figure 4, each subplot demonstrates success rates as percentage (y-axis) value versus the population size (x-axis) at which AIGA runs at different numbers of groups on Damavandi function. It can be seen in the Figure 4; the success rate increases as population size increases for each group number. However, the increase in success rate here is not as much as the increase in success rate achieved by increasing the number of groups. In multimodal functions, it is normal to expect that for any optimization algorithm the larger population size the better result. Since population is distributed into zones as in AIGA, augmenting the quantity of groups results in a reduction of individuals within each group. Thus, increase in population size becomes more effective in success of finding global optimum as the number of groups increases.

The success rates of AIGA in finding the global optimum of the Damavandi function dependent on both group number and population size are shown in Table 1. As the average success rates are considered, the increase in population size or number of groups or increase in both result with an enhancement in the success rate of finding global optimum. The largest group number exhibits a success rate of 99.60 percent on average, whereas the average success rate for the largest population size is 60.40 percent. Namely, the number of groups as a parameter of AIGA is much more effective than the population size as a decisive parameter to obtain a better success rate. While the population size for 1-group corresponding to TGA was increased from 300 to 1800, the success rate reached 11.3 percent from 1.5 percent. Similarly, while the population size for 25-groups of AIGA was increased from 300 to 1800, the success rate reached 100 percent from 96.2 percent. As a result, AIGA even in small population size performs much better than TGA.

Table 1. Success rates (%) of AIGA for 2d Damavandi function.

Population	Number of Groups												
Size	#1	#2	#4	#6	#8	#9	#10	#12	#15	#16	#20	#25	Average
300	1.5	2.0	2.3	11.2	16.3	32.2	28.3	43.2	54.6	71.9	82.0	96.2	36.8
400	2.0	2.7	4.2	12.2	24.9	37.4	31.6	51.3	59.5	82.0	86.5	98.0	41.0
500	3.1	3.5	3.6	15.1	28.3	43.6	36.7	61.1	66.1	87.5	90.5	99.7	44.9
600	3.7	4.8	3.5	17.5	29.8	51.3	37.8	66.9	75.7	90.7	94.2	99.8	48.0
700	4.3	4.1	4.3	20.2	34.1	51.1	45.5	67.2	76.4	93.6	95.8	99.8	49.7
800	5.0	5.0	5.6	20.1	35.9	56.9	49.4	71.4	81.1	94.5	97.9	99.8	51.9
900	4.0	7.2	6.5	20.8	36.9	59.6	53.7	75.1	84.6	97.4	98.1	100.0	53.7
1000	5.7	7.0	7.8	23.1	43.1	66.5	52.1	78.1	86.5	97.3	98.8	100.0	55.5
1100	5.7	5.2	8.1	22.5	43.1	64.1	56.0	81.0	89.7	97.9	99.7	100.0	56.1
1200	6.1	7.4	8.0	21.2	45.7	65.0	57.6	83.2	91.8	99.1	99.4	100.0	57.0
1300	7.1	9.4	8.1	21.0	43.5	70.9	58.3	86.5	92.1	99.6	99.7	100.0	58.0
1400	8.6	9.8	12.7	24.4	47.0	73.7	62.3	87.8	93.3	99.4	99.7	100.0	59.9
1500	9.8	11.5	9.8	24.3	42.5	75.3	61.2	87.6	92.5	99.2	99.7	100.0	59.5
1600	10.3	9.8	11.9	23.0	45.7	72.0	62.0	88.8	93.6	99.5	99.9	100.0	59.7
1700	12.4	9.8	11.5	23.0	44.4	73.0	61.4	88.7	93.7	99.6	99.8	100.0	59.8
1800	11.3	10.8	14.2	23.5	42.8	73.7	64.5	89.3	94.9	99.4	99.9	100.0	60.4
Average	6.3	6.9	7.6	20.2	37.8	60.4	51.2	75.5	82.9	94.3	96.4	99.6	

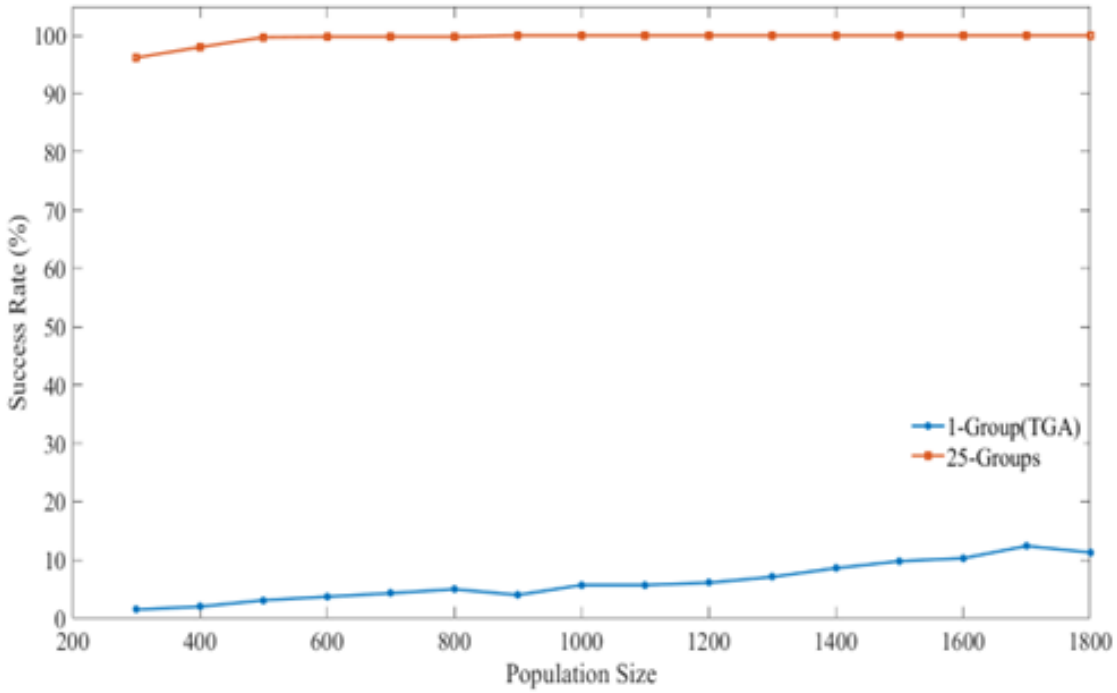

Figure 5. Success rates vs population size for TGA and best number of groups of AIGA.

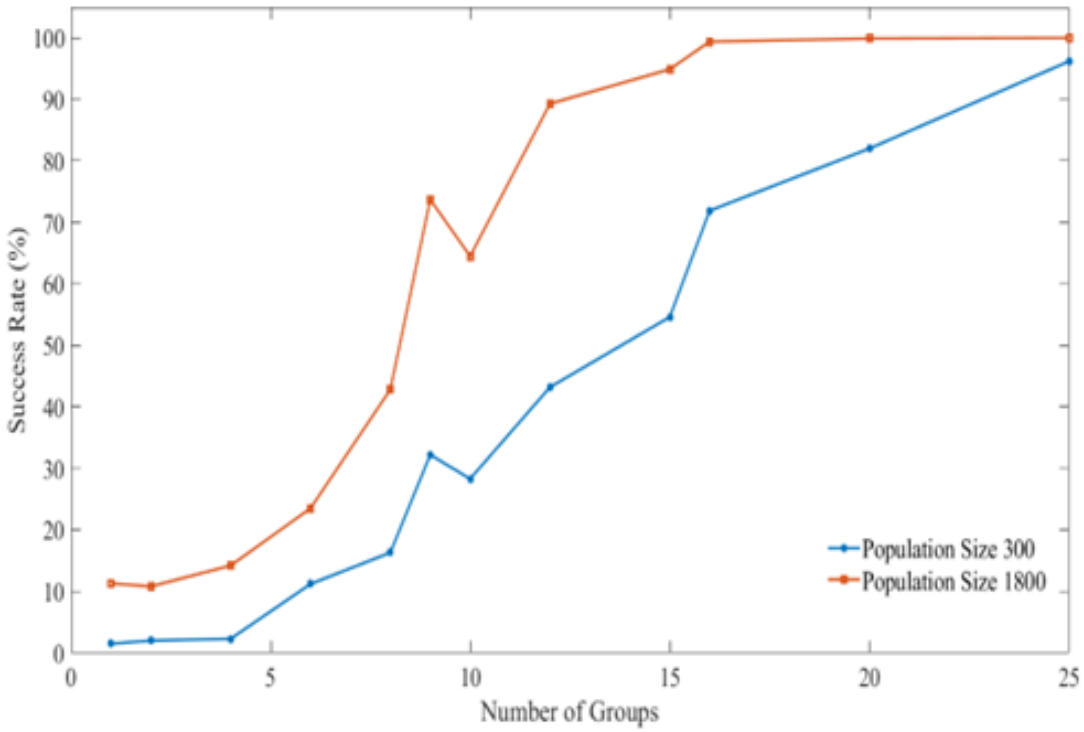

Figure 6. Success rates vs number of groups of AIGA for min. and max. population size.

In Figure 5, two lines on plot show success rates versus population size for the TGA (AIGA; 1-group) and AIGA; 25-groups. Success rates vs population size for TGA and different number of groups of AIGA were shown in subplots of Figure 2, before. Two lines on plot, one corresponds to TGA and other one corresponds to the best group AIGA, were selected to make comparison in Figure 5. Although AIGA distributes the population to different zones depending on the number of groups, it is more successful than TGA even with a small population size.

In Figure 6, two lines on plot show success rates versus the number of groups for the selected minimum and maximum population size as 300 and 1800, respectively. TGA (AIGA; 1-group) has been shown the lowest success rate when compared with all number of groups in both lines.

3.2 Case Study 2: Gaussian-Like Function

As the second case study, a 2-dimensional multimodal test function with different amplitudes and peaks derived from the Gaussian-Like function was used. Kilinc and Caicedo [22] derived the Gaussian function as an example optimization problem to obtain multiple optimal solutions. The equation for the test function is given in Equation 2.

$$f(x) = 4e^{\left(\frac{(-x_1^2-x_2^2)}{50}\right)} - 5e^{\left(\frac{(-x_1^2-x_2^2)}{8}\right)} + 2e^{(-2(x_1-3)^2-2(x_2+2)^2)} + 7e^{\left(\frac{-25(x_1+3)^2-25(x_2-2)^2}{2}\right)} - 3e^{\left(\frac{-25(x_1+3)^2-25(x_2-2)^2}{18}\right)} - 4e^{(-50(x_1-3)^2-50(x_2-2)^2)} \quad (2)$$

The function given in Equation 2 has two local minima with higher amplitude than the global minimum at the intervals of both variables [-5, 5]. It is known that the smaller the amplitude of the global minimum the harder it is to find. Therefore, the test function given in Equation 2 is very plausible for testing optimization algorithms. The graph of the test function is shown in Figure 7.

In this case study, the maximum allowed number of evaluations to achieve the global optimum was taken as 40,000. The population size of both TGA and AIGA were selected as 300, 600, 900, and 1200. In a similar manner, the number of groups as a parameter of AIGA were selected as 1, 4, 9, and 16. AIGA was run 400 times for each population size and the number of groups. The results obtained are presented by both in tabular form and the graphs as the success rate.

In Figure 8, each line on plot demonstrates success rates as percentage (y-axis) value versus the number of groups (x-axis) at which AIGA runs at different population sizes on the Gaussian Like function. Plots in Figure 8 shows that the success of AIGA is seen markedly as the number of groups increases different population sizes.

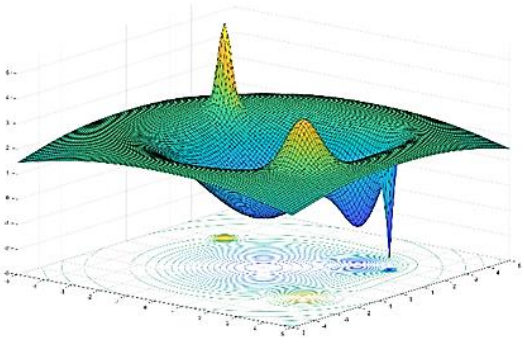

Figure 7. Gaussian-Like Function.

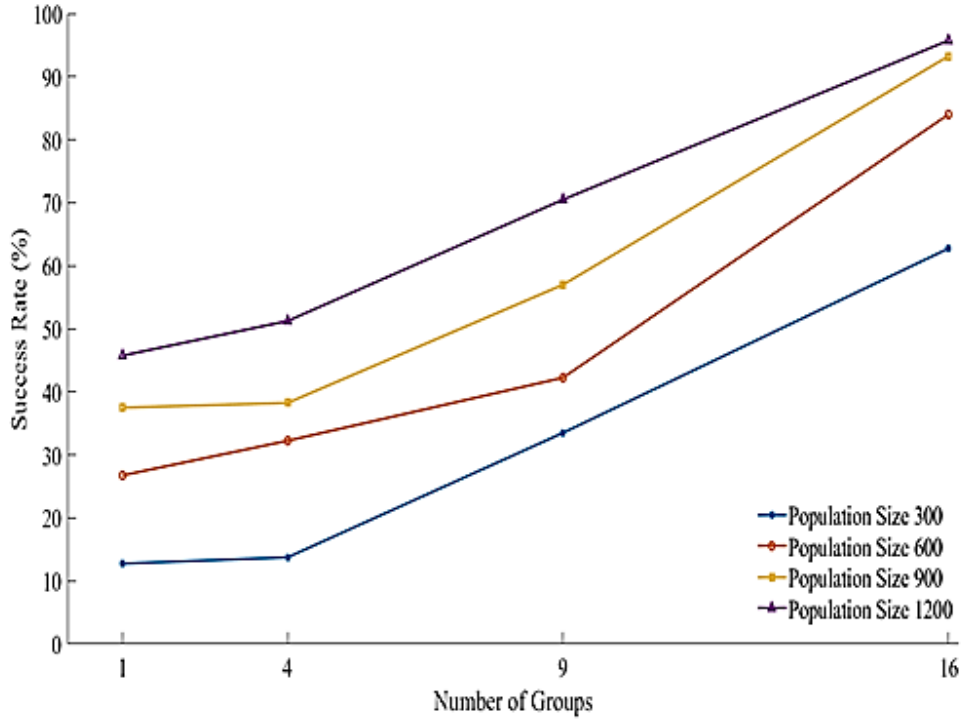

Figure 8. Success rates (%) vs number of groups for different population.

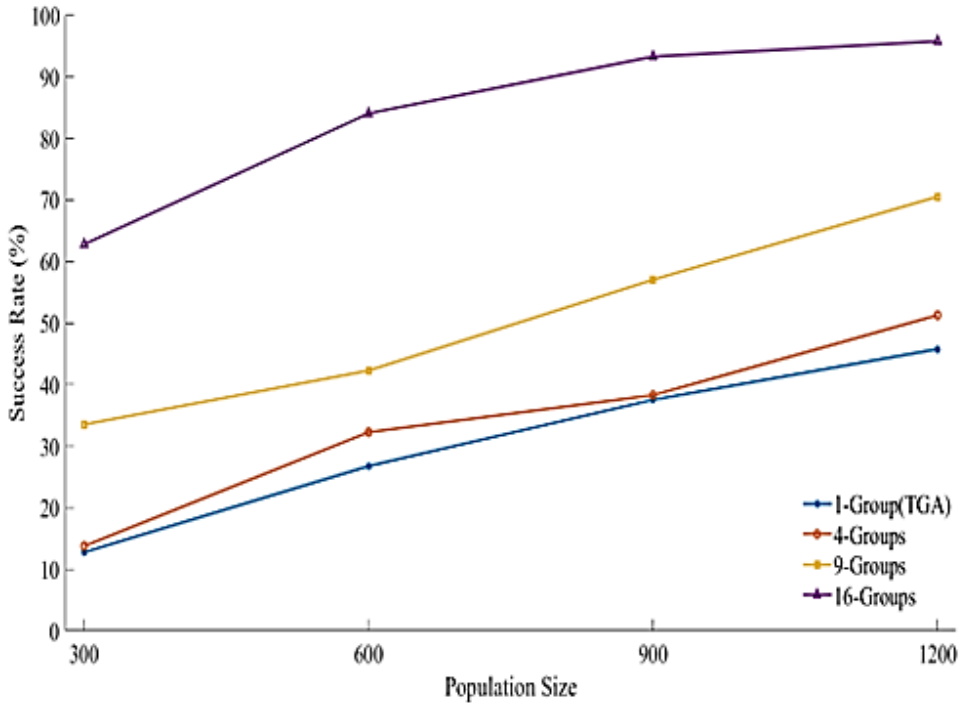

Figure 9. Success rtes (%) vs population size for different number of groups.

In Figure 9, each line on plot demonstrates success rates as percentage (y-axis) value versus the population size (x-axis) at which AIGA runs at different numbers of groups for Gaussian like function. Lines in Fig. 10 show that the success of both TGA (1-group AIGA) and AIGA is enhanced as population size increases for each group number. In the derived 2D gaussian function, the increase in population size was seen that improved the result. Nevertheless, an increase in the number of groups was found to be more effective than that of population size as concluded for Damavandi Function in case study 1. As a result, higher success rate can be achieved to obtain global optimum by using AIGA with the same population size instead of increasing the population size for TGA.

The success rates of AIGA in finding global optimum for the derived 2D Gaussian function dependent on both number of groups and population size are shown in Table 2. In terms of success rate, the increase in population size or number of groups or increase in both result with higher success rate of finding global optimum. While the average success rate for to the largest group number is 89.80 percent, the average success rate for to the largest population size is 44.60 percent. While the population

size for TGA was increased from 300 to 1200, the success rate reached 6.1 percent from 1.5 percent. Similarly, while the population size for Group 16 of AIGA was increased from 300 to 1200, the success rate reached almost 100 percent from 71.9 percent. Therefore, it is concluded that the increase in the number of groups is much more effective than the population size to increase the success rate. Based on the above discussion, it should be noted that 1-group corresponds to TGA, while n-group corresponds to AIGA which the search space is divided into n groups, thus using AIGA means better than using TGA to achieve finding global optima.

Table 2. Success rates (%) of AIGA for 2d Gaussian-like function.

Population Size	Number of Groups				
	#1	#4	#9	#16	Average
300	1.5	2.3	32.2	71.9	27.0
600	3.7	3.5	51.3	90.7	37.3
900	4.0	6.5	59.6	97.4	41.9
1200	6.1	8.0	65.0	99.1	44.6
Average	3.8	5.1	52.0	89.8	

4. Discussion

The performance of the AIGA was tested with widely recognized benchmark functions and compared with existing well-known algorithms such as Grey Wolf Optimizer (GWO) [7], Particle Swarm Optimization (PSO) [8], Firefly Algorithm [9], Bat Algorithm [10], Crow Search Algorithm [11], Cuckoo Search Algorithm [12], Flower Pollination Algorithm (FPA)[14]. Comparisons were carried out using functions with 2 and 30 dimensions having a significant search space size difference. For optimization, complex functions with different mathematical characteristics were selected. For instance, Damavandi function is continuous, differentiable, non-scalable and multimodal, Schwefel function is continuous, differentiable, scalable and has many local optima, Griewank function is continuous, differentiable, non-separable and has many local optima, Rosenbrock function is continuous, differentiable, non-separable and unimodal.

Firstly, Damavandi, Griewank, Schwefel, Rosenbrock, and Gaussian functions set to two dimensions shown in Table 3 were run 30 times with below algorithms.

The average and standard deviation of the obtained results for each function and algorithm were presented in the Table 3. Damavandi and Gaussian functions have been studied in detail above in the success of AIGA over TGA. In comparison to other algorithms, it is understood from the evaluation of average and standard deviation results (Table 4) that AIGA performs mostly better in capturing the global optimum.

Table 3. Benchmark functions used for 2-dimension experiments.

Function	Equation	dim	Range	f_{min}
Damavandi	$f(x)=\left[1-\left\lvert\frac{sin[\pi(x_1-2)]sin[\pi(x_2-2)]}{\pi^2(x_1-2)(x_2-2)}\right\rvert^5\right][2+(x_1-7)^2+(x_2-7)^2]$	2	[0,14]	0
Griewank	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{400}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	2	[-600,600]	0
Schwefel	$f(x)=418.9829d-\sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	2	[-512, 512]	0
Rosenbrock	$f(x)=\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	2	[-5,5]	0
Gaussian	$f(x)=4e^{\left(\frac{(-x_1^2-x_2^2)}{50}\right)}-5e^{\left(\frac{(-x_1^2-x_2^2)}{8}\right)}+2e^{(-2(x_1-3)^2-2(x_2+2)^2)}+7e^{\left(\frac{-25(x_1+3)^2-25(x_2-2)^2}{2}\right)}-3e^{\left(\frac{-25(x_1+3)^2-25(x_2-2)^2}{18}\right)}-4e^{(-50(x_1-3)^2-50(x_2-2)^2)}$	2	[-5,5]	0

Table 4. Comparison of algorithms for selected benchmark functions (dim = 2).

Function / Algorithm	Damavandi		Griewank		Schwefel		Rosenbrock		Gaussian	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
AIGA	0.6667	0.9589	0.0000	0.0000	0.0000	0.0000	0.0093	0.0162	-2.6560	0.0000
GWO	1.8674	0.5047	0.0025	0.0035	3.9491	21.6237	0.0650	0.2179	-2.6008	0.3023
PSO	1.9333	0.3651	0.0003	0.0013	3.9480	21.6238	33.2036	140.0460	-2.6560	0.0000
Firefly	1.9552	0.2452	0.0019	0.0031	68.5925	46.6370	55.0196	74.5664	-1.5939	0.3217
BAT	1.9747	0.3066	0.3271	0.2342	24.9053	38.2936	642.8919	518.1514	-1.4556	0.3472
Crow S.	1.6735	0.6231	0.0003	0.0003	0.0000	0.0000	0.0096	0.0151	-2.6560	0.0000
Cuckoo S.	0.2000	0.6103	0.0009	0.0012	0.0002	0.0003	1.5223	1.3912	-2.6560	0.0001
FPA	0.4811	0.7417	0.0076	0.0036	0.2090	0.2028	8.3209	6.8515	-2.6278	0.0271

Lastly, Griewank, Schwefel, Rastrigin, Michalewicz and Styblinski-Tang functions set to thirty dimensions shown in the Table 5 were run 30 times with below algorithms as made above.

The mean and the variability of the obtained results for each function defined in a much larger search space, and for each algorithm are shown in Table 6. Likewise, AIGA outperformed over the other algorithms in case of

significantly larger (higher dimension) search space as well. The large differences in average values and standard deviation seen in Table 6 indicate that other algorithms are trapped more on local optima.

This proposed method is more effective in many engineering problems where the number of variables is high and a definitive result cannot be achieved, and in problems where there is a possibility of getting stuck in a local result. Planar and space truss structures optimization can be given as an example of concrete engineering problems in these studies.

Table 5. Benchmark functions used for 30-dimension experiments.

Function	Equation	dim	Range	f_{min}
Griewank	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{400}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600,600]	0
Schwefel	$f(x)=418.9829d-\sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-512, 512]	0
Rastrigin	$f(x)=10d+\sum_{i=1}^{d}\left[x_i^2-10\cos(2\pi x_i)\right]$	30	[-5.12, 5.12]	0
Michalewicz	$f(x)=-\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	30	$[0,\pi]$	-1.8013
Styblinski-Tang	$f(x)=\frac{1}{2}\sum_{i=1}^{d}\left(x_i^4-16x_i^2+5x_i\right)$	30	[-5,5]	-39.16599d

Table 6. Comparison of algorithms for selected benchmark functions (dim = 30).

Function / Algorithm	Griewank		Schwefel		Rastrigin		Michalewicz		Styblinski-Tang	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
AIGA	0.0090	0.0027	94.7817	99.0926	0.0117	0.0068	-29.5855	0.0322	-1174.9822	0.0009
GWO	0.0025	0.0047	5933.4899	795.9782	2.1056	3.3369	-17.1622	3.5241	-1017.4006	50.0187
PSO	0.0094	0.0103	5605.3116	665.5943	41.5892	16.2007	-26.6819	0.9912	-1035.9739	31.9540
Firefly	0.0298	0.0161	9236.3183	393.6007	18.7905	9.5962	-17.7670	2.0266	-750.5202	0.1449
BAT	185.3600	51.3961	4862.3753	4673.7892	319.2250	24.9510	-8.3810	0.7271	-669.7952	46.0701
Crow S.	0.8102	0.1110	5374.9974	684.3050	8.2851	6.5580	-19.2515	1.9559	-1012.5750	27.0560
Cuckoo S.	19.0544	3.0362	4273.5146	159.2521	146.1697	9.0925	-16.1591	0.5792	-969.3548	12.0875
FPA	155.0082	22.9206	6322.0616	186.1977	256.9353	9.6748	-11.7977	0.3635	-765.4912	18.3458

5. Conclusion

In this study, a new strategy proposed to be used for optimization algorithms. The new strategy has been tested on a genetic algorithm and aim was achieved by overcoming the problem of trapping the local optimum. In other words, the proposed method AIGA is not a new algorithm, rather is a new methodology that can be used in any optimization algorithm. Similar problems exist in other optimization algorithms such as trapping local optimum and premature convergence problems in genetic algorithms. By employing this suggested approach, exploration will persist across all regions within the search space, ensuring that the quest for the global optimum point extends to the majority of the population.

The performance of AIGA is verified using different optimization problems revealing the weakness of TGA in the case studies. For optimization problems, 2-dimensional multimodal Damavandi and Gaussian-Like functions having very small amplitude of global optimum and particularly local optima with larger amplitude have been selected. Finding the global optimum in stochastic optimization algorithms poses a significant challenge due to the low likelihood or difficulty in escaping local optima. The success of AIGA regardless of the precision of the result has been tested by observing if it reaches the global optimum in case studies. When validating case studies, consistency was maintained across all parameters and operators for both TGA and AIGA, with a fixed number of fitness function evaluations. In this context, detailed conclusions have been acquired as below:

In general, the probability of finding the global optimum increases as an increase in population size. In the proposed algorithm AIGA, even with fixed population size, the probability of reaching the global optimum significantly increases much more as the number of groups increases. It has been concluded that the increase in the number of groups is more effective than the increase in population size, thus AIGA is superior to TGA. Based on the results of this study, it can be stated that this approach can also be applied to improve other similar optimization algorithms.

Author contributions

Muslum Kilinc: Conceptualization, Software, Methodology, Validation, Writing-Original draft preparation. **Emrah Atilgan:** Methodology, Validation, Writing-Original draft preparation. **Cengiz Atis:** Validation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. Atilgan, E., & Hu, J. (2018). First-principle-based computational doping of SrTiO 3 using combinatorial genetic algorithms. Bulletin of Materials Science, 41(1), 1. https://doi.org/10.1007/s12034-017-1515-9
2. S., V. C. S., & S., A. H. (2022). Nature inspired meta heuristic algorithms for optimization problems. Computing, 104(2), 251-269. https://doi.org/10.1007/s00607-021-00955-5
3. Fister Jr, I., Yang, X. S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. Neural and Evolutionary Computing, 80(3), 116-122. https://doi.org/10.48550/arXiv.1307.4186
4. Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. Future Computing and Informatics Journal, 3(2), 231-246. https://doi.org/10.1016/j.fcij.2018.06.001
5. Yang, X. S. (2020). Nature-inspired optimization algorithms: Challenges and open problems. Journal of Computational Science, 46, 101104. https://doi.org/10.1016/j.jocs.2020.101104
6. Stork, J., Eiben, A. E., & Bartz-Beielstein, T. (2022). A new taxonomy of global optimization algorithms. Natural Computing, 21(2), 219-242. https://doi.org/10.1007/s11047-020-09820-4
7. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007
8. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95-International Conference on Neural Networks, 4, 1942-1948. https://doi.org/10.1109/ICNN.1995.488968
9. Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.
10. Yang, X. S., & Hossein Gandomi, A. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29(5), 464-483. https://doi.org/10.1108/02644401211235834
11. Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Computers & Structures, 169, 1-12. https://doi.org/10.1016/j.compstruc.2016.03.001
12. Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), 210-214. https://doi.org/10.1109/NABIC.2009.5393690
13. Chechkin, A. V., Gonchar, V. Y., Klafter, J., & Metzler, R. (2006). Fundamentals of Lévy flight processes. Fractals, Diffusion, and Relaxation in Disordered Complex Systems: Advances in Chemical Physics, Part B, 439-496.
14. Yang, X. S. (2012). Flower pollination algorithm for global optimization. In International Conference on Unconventional Computing and Natural Computation, 7445, 240-249. https://doi.org/10.1007/978-3-642-32894-7_27
15. Rocha, M., & Neves, J. (1999). Preventing premature convergence to local optima in genetic algorithms via random offspring generation. In Multiple Approaches to Intelligent Systems: 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-99, Cairo, Egypt, May 31-June 3, 1999. Proceedings 12, 127-136. https://doi.org/10.1007/978-3-540-48765-4_16
16. Dang, D. C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., ... & Sutton, A. M. (2017). Escaping local optima using crossover with emergent diversity. IEEE Transactions on Evolutionary Computation, 22(3), 484-497. https://doi.org/10.1109/TEVC.2017.2724201
17. Doerr, B. (2020). Does comma selection help to cope with local optima?. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, 1304-1313. https://doi.org/10.1145/3377930.3389823
18. Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., & Trubenová, B. (2018). How to escape local optima in black box optimisation: when non-elitism outperforms elitism. Algorithmica, 80, 1604-1633. https://doi.org/10.1007/s00453-017-0369-2
19. Sharma, P., & Raju, S. (2024). Metaheuristic optimization algorithms: A comprehensive overview and classification of benchmark test functions. Soft Computing, 28(4), 3123-3186. https://doi.org/10.1007/s00500-023-09276-5
20. Cheng, R., Li, M., Tian, Y., Xiang, X., Zhang, X., Yang, S., ... & Yao, X. (2018). Benchmark functions for the cec'2018 competition on many-objective optimization.
21. Deb, L. (1993). Multimodal deceptive functions. Complex Systems, 7, 131-153.
22. Kilinc, M., & Caicedo, J. M. (2019). Finding plausible optimal solutions in engineering problems using an adaptive genetic algorithm. Advances in Civil Engineering, 2019(1), 7475156. https://doi.org/10.1155/2019/7475156