Modified Grasshopper Optimization Algorithm Based Genetic Algorithm for Global Optimization Problems: The System of Nonlinear Equations Case Study

Hala A. Omar
Menoufia University
Mohammed El-Shorbagy (mohammed_shorbagy@yahoo.com)
Prince Sattam bin Abdulaziz university https://orcid.org/0000-0002-8115-0638

Research Article

Keywords: Grasshopper optimization algorithm (GOA), genetic algorithm (GA), global optimization, the system of nonlinear equations

Posted Date: April 12th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-382227/v1

Modified Grasshopper Optimization Algorithm Based Genetic Algorithm for Global Optimization Problems: The System of Nonlinear Equations Case Study

Hala A. Omar[1,2] , M.A. El-Shorbagy[3,1,*]

[1]Department of Basic Engineering Science, Faculty of Engineering, Menoufia University, Shibin El Kom, Egypt
[2]Department of mathematics, Umm AL-Qura University, Saudi Arabia.
[3]Department of Mathematics, College of Science and Humanities in Al-Kharj, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia

Abstract

Grasshopper optimization algorithm (GOA) is one of the promising optimization algorithms for optimization problems. But, it has the main drawback of trapping into a local minimum, which causes slow convergence or inability to detect a solution. Several modifications and combinations have been proposed to overcome this problem. In this paper, a modified grasshopper optimization algorithm (MGOA) based genetic algorithm (GA) is proposed to overcome this problem. Modifications rely on certain mathematical assumptions and varying the domain of the C_{max} control parameter to escape from the local minimum and move the search process to a new improved point. Parameter C is one of the most important parameters in GOA where it balances the exploration and exploitation of the search space. These modifications aim to lead to speed up the convergence rate by reducing the repeated solutions and the number of iterations. The proposed algorithm will be tested on the 19 main test functions to verify and investigate the influence of the proposed modifications. In addition, the algorithm will be applied to solve 5 different cases of nonlinear systems with different types of dimensions and regularity to show the reliability and efficiency of the proposed algorithm. Good results were achieved compared to the original GOA.

Keywords: Grasshopper optimization algorithm (GOA), genetic algorithm (GA), global optimization, the system of nonlinear equations.

1. Introduction

The collective behavior resulting from the social insects operating under extremely few rules can be defined as metaheuristic algorithms (MAs). MAs' famous algorithms coming from the animal's world, such as schools of fish, the flocks of birds, and bugs' swarms. Furthermore, MAs are known as computational models that mimic systems of the natural swarm. To date, most MAs have been recommended and successfully applied for the solution of optimization problems in the literature. Examples of MAs models are: genetic algorithm [1,2], ant colony optimization [3], particle swarm optimization [4-6], artificial bee colony [7], cat swarm optimization [8], bacterial foraging [9], firefly algorithm [10], glowworm swarm optimization [11], krill herd algorithm [12], monkey algorithm [13], sine cosine algorithm [14], cuckoo search algorithm [15], grasshopper optimization algorithm [16], salp swarm algorithm [17], slime mould algorithm [18], harris hawks optimization [19], gradient-based optimizer [20], equilibrium optimizer algorithm [21,22], etc.

One of the novel MAs based on the swarming nature of grasshoppers is the grasshopper optimization algorithm (GOA). In GOA, the optimal global optimum optimization values of the problem are primarily dependent on the forces of social interaction. It is commonly used in a number of optimization problems due to its fast implementation and high precision, robustness, and performance. But GOA has some drawbacks, including 1) an unbalancing of exploiting and exploring processes; 2) an unstable rate of convergence, and 3) falling into the local optima. So, there are many modifications of GOA solve these limitations in the literature [23-32]. In [24] GOA is enhanced, to overcome the above shortcomings, by using a nonlinear convergence parameter, niche mechanism, and the β-hill climbing technique. In [26] GOA, the authors employed the idea of chaos to create a uniformly distributed population, to increase the consistency of initial populations and the capacity to look for a new space, and the leveraging existing search space. In order to resolve the shortcomings of a traditional GOA and to detect the autism spectrum disorder at all times of life, the authors in [27] suggested a modified GOA. In [28] modified GOA is proposed as a new search method combined with a convolutional neural network (CNN) to solve modeled problems and to retrieve similar images efficiently, despite total search in the database. In order to create a more acceptable trade-off between diversification and intensification and produce substantially better performance than traditional gray wolf optimization (GWO) and GOA, hybridized GWO with GOA was suggested in [30]. In [32] dynamic population quantum binary GOA is proposed as an enhanced GOA for feature selection based on shared knowledge and rough

set theory. In [33] an improved GOA was suggested, using the nonlinear comfort zone parameter to enable the use of the iterations of the algorithm and the Lévy flight method to maximize randomness and extend the local search range. Moreover, a random hopping technique to help the algorithm leap from the local optimum has been implemented.

As we have seen from the survey, there are many modifications made to GOA to overcome the abovementioned shortcomings. Consequently, work should be done to introduce new methods to enhance the performance of traditional GOA. From this motivation, this paper proposed a modified grasshopper optimization algorithm (MGOA) to overcome the shortcomings of the original GOA and improve its accuracy. There is a significant parameter *C* in the original GOA. This parameter is a reducing coefficient that decreases three zones: the comfort zone, the zone of repulsion, and the zone of attraction. *C* decreases in proportion to the number of iterations used to balance exploration and exploitation. It is calculated based on the parameters C_{max} (upper bound of C), C_{min} (lower bound of *C*), and the maximum number of iterations. The proposed modifications depend on varying the upper bound C_{max} through the genetic algorithm (GA) to help GOA out of the local optima when it falls into it, then *C* is updated according to its classical formula.

The main academic contributions of this paper are:

1. Introducing a local search mechanism based on a genetic algorithm (GA) to bring GOA out from the local minimum.
2. Improving the overall performance of GOA by updating the parameter value C_{max} after each GA stage.
3. Preventing the divergence of objective function value from the best reached value by controlling the space of the expected calculated objective function. Even with a strong local minimum if no noticeable improvement in the objective function occurred, the modifications transfer the search process to another part of the domain taking GOA out of the local minimum.
4. Preventing time-consuming, calculation burden and wasting iterations by limiting the number of repeated solutions yield from GOA to decide whether trapped into local minimum.
5. Preventing GA from working as a global search or dominates the search process by defining and controlling the search space of GA parameters and keeping tracking of the parameter C_{max}.

The rest of the paper is presented as follows; Section 2 is presented the preliminaries about the problem. Section 3 introduces genetic algorithms (GA) and grasshopper optimization algorithm (GOA) with explaining the proposed algorithm MGOA in detail. Section 4 presents the numerical results with discussions. Finally, the conclusion and future works are given in Section 5.

2. Preliminaries

2.1 Global Optimization

Optimization is the search process for the optimum parameters of a given objective function. The mathematical formula can be written as follows for the single objective optimization problem:

Find $x = (x_1, x_2, ..., x_n)$ such that:

$$\begin{aligned} &\text{Min } f(x), \\ &\text{Subject to:} \\ &\quad L_d \le x_d \le U_d \ \forall\ d = 1, 2, ..., n; \end{aligned} \tag{1}$$

where x is the solution vector, $f(x)$ is an objective function, L_d and U_d represent the lower and the upper bounds, respectively, for decision variables $x_d \forall\ d = 1, 2, ..., n$.

2.2 The System of Nonlinear Equations

The system of nonlinear equations (SNEs) is defined mathematically as:

$$\text{SNEs}: \begin{cases} f_1(x) = 0 \\ f_2(x) = 0 \\ \quad \vdots \\ f_i(x) = 0 \end{cases}; \tag{2}$$

where each function $f_i \forall i = 1, \ldots, m$ is a nonlinear function, which maps the vector $x = (x_1, x_2, \ldots, x_n)$ of the n-dimensional space R^n to real line. Some of the functions may be linear and others nonlinear. The solution for a nonlinear system is to find solutions so that each of the above functions $f_i \forall i = 1, \ldots, m$ is equal to zero.

Definition 1: If $\forall i, i = 1, \ldots, m, f_i(x) = 0$, then the solution $x = (x_1^*, x_2^*, \ldots, x_n^*)$ is called the optimal solution of the SNEs.

There are many methods that convert the SNEs to an optimization problem [34-37]. To solve SNEs, it is usually transformed into an equivalent single unconstrained objective optimization

problem where, the objective function is represented as the sum of squared residuals of nonlinear equations. problem can be presented as:

$$min\, F(x) = min\left(\sqrt{\sum_{i=1}^{m} f_i^2(x))}\right) = min\|f_i(x)\|, x = (x_1, x_2, \ldots, x_n); \quad (3)$$

where $F(x)$ is the objective function that will be minimized.

3. The Proposed Algorithm

This section presents an introduction of genetic algorithms (GA) and grasshopper optimization algorithm (GOA). In addition, the proposed algorithm is discussing in detail.

3.1 Genetic algorithms (GA)

The invention of genetic algorithms (GA) was dated back to the 1960s by Holland and further described by Goldberg [38]. The GA was successfully applied to problems in many areas, including optimization design, neural networks, fuzzy logic control, planning, expert systems, and many more [39]. GA codes the solution as a chromosome to any optimization problem. It then determines the first population of those chromosomes that is part of the problem's solution space. The search space is thus defined as the space for the solution in which every chromosome represents each feasible solution. Before the search begins, the initial population is picked by random chromosomes from the search space. The individuals would then be chosen competitively by means of calculations based on their fitness which calculated by a particular objective function.

Genetic search operators including selection, crossover, and mutation are then used one after the other to produce new chromosomes generation. This process is repeated until the end criterion has been met and the finished solution is the best chromosome of the latest generation. The general GA pseudo-code is displayed in Figure 1.

Generate an initial population.
Evaluate fitness of individuals in the population;
Do:
    Select parents from the population;
    Recombine parents to produce children;
    Evaluate fitness of the children;
    Replace some or all of the population by the children;
while a satisfactory solution has been found.

Fig. 1. The general GA pseudo-code

3.2 Grasshopper optimization algorithm (GOA)

GOA is a recent optimization technique based on swarm algorithm depending on simulating the swarm nature of the grasshoppers. The main feature of the grasshopper's swarm is described in two phases. The first one is larval phase where the steps are small, and the movement is slow. However, the second one is long-range and fast in adulthood phase.

Another important feature is food seeking. The process of foraging can be divided into two branches exploration and exploitation.

In the GOA, each grasshopper represents a solution in the population. The grasshopper's position calculations depend upon three types of forces. These forces are, the social interaction between each one and other grasshoppers S_i, the gravity force G_i and the wind movement A_i.

The resultant force affects each grasshopper can be defined as:

$$X_i = S_i + G_i + A_i \tag{4}$$

The social interaction force between each grasshopper and other grasshoppers can be defined as:

$$S_i = \sum_{j=1}^{N} s(d_{ij})\, \widehat{d_{ij}}; \tag{5}$$

where d_{ij} is the distance between the grasshopper *i* and grasshopper *j*.

There are two main types of forces between grasshoppers, the attraction force and repulsion force. The function (*s*) represents the strength of these two-social forces. It is defined as:

$$s(r) = f e^{-\frac{r}{l}} - e^{-r}; \tag{6}$$

where *f, l* are the intensity of the attraction and the attractive length scale. The changing of these parameters affects the social behavior of the grasshoppers.

The interval [0, 2.079] represents the distance where a repulsion force will occur between two grasshoppers. It should be mentioned that, when the distance between two grasshoppers is 2.079, there will be no social force between grasshoppers which will form the comfortable zone. However, If the distance will be more than 2.079 the attraction force will increase, then the attraction force will decrease gradually when it reaches to 4. The distance between every two grasshopper affects the calculations of social forces. When the distance between the grasshoppers is increased than 10, their function (*s*) may fail to apply the forces between all of them. As a result, the distance of grasshoppers is mapped in the interval [1,4]. The second affected force is the gravity force which is defined as:

$$G_i = -g\widehat{e_g}; \tag{7}$$

where g is the gravitational constant and $\widehat{e_g}$ is a center of earth unity victor.

The wind direction can be calculated as:

$$A_i = u\widehat{e_w}; \tag{8}$$

where u is a constant drift and $\widehat{e_w}$ is a wind direction unity vector.

The position of the grasshopper can be calculated as:

$$X_i = \sum_{j=1}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\widehat{e_g} + u\widehat{e_w}. \tag{9}$$

To achieve convergence and prevent the grasshoppers from reaching the comfort zone before reaching the solution, Eq.9 can be represented as:

$$X_i = C\sum_{j=1}^{N} C\left(\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T_d}; \tag{10}$$

where $ub_d, lb_d$are the upper and lower bound and $\widehat{T_d}$ is the target value in the dth dimension. C is a decreasing parameter for controlling the zone of attraction, the zone of repulsion, and the comforting zone. It decreases proportionally to the number of iterations with a balance to the exploration and exploitation of the algorithm. This constant can be calculated as:

$$C = C_{max} - l\frac{C_{max} - C_{min}}{L} \tag{11}$$

where C_{max} and C_{min} are the maximum and minimum values, l is the current iteration and L is the maximum number of iterations.

The steps of GOA algorithm can be summarized as:

1- **Initialization**. In this step, the values of the parameters are defined. These values include, C_{max}, C_{min}, f, l and the maximum number of iterations.

2- **Initial population and evaluation**. The initial population is generated randomly then, each candidate is evaluated. The fitness of each grasshopper is corresponding to the value of objective function at this grasshopper.

3- **Assigning the best solution**. After evaluating all the solutions, the best solution is defined as (the target position) and it's corresponding fitness is assigned as (the target fitness).

4- **Updating the decreasing coefficient parameter**. The value of the decreasing coefficient C is then calculated at each iteration depending on Eq. 11

5- **Updating solution**. Each solution in the population will be updated as shown in Eq. 10.

6- **Solution boundaries violation**. After updating the solution, all solutions should be examined to be within the predetermined boundaries. If it is not, it rests to its domain.

7- **Examining all candidate solutions**. Steps from 3 to 6 are repeated for all members in the population.
8- **The best reached solution is detected**. Both target position and target fitness will be evaluated, and the best global solution is assigned.
9- **Termination criteria and the best solution**. all steps are repeated until either the termination criterion or the maximum number of iterations is reached. Then, the optimum solution is displayed.

3.3 Modified grasshopper optimization algorithm (MGOA)

Figure 2 displays the flow chart of the proposed algorithm. As shown in the figure, GOA algorithm starts the optimization process by using steps in subsection 3.2. If the termination criterion is reached, the algorithm terminates, and the best solution $\widehat{T_d}$ its corresponding objective value are displayed as the desired solution. Otherwise, the algorithm would be checked whether it is trapping into a local minimum. If the answer is no, the optimization process will be completed with the GOA. However, if the answer is yes, the genetic algorithm (GA) is applied to help GOA get out of the local minimum. This goal will be achieved by detecting another new best target position $\widehat{T_d}$ and defining a modified value of the parameter C_{max}. Again, if the termination criterion is reached, the algorithm terminates and the best solution $\widehat{T_d}$and the corresponding objective value are displayed. Otherwise, the target position $\widehat{T_d}$, fitness, and C_{max} will be updated and GOA will continue the optimization process based on these updated values. These steps are repeated until the defined termination criterion is reached and the optimal solution is obtained.

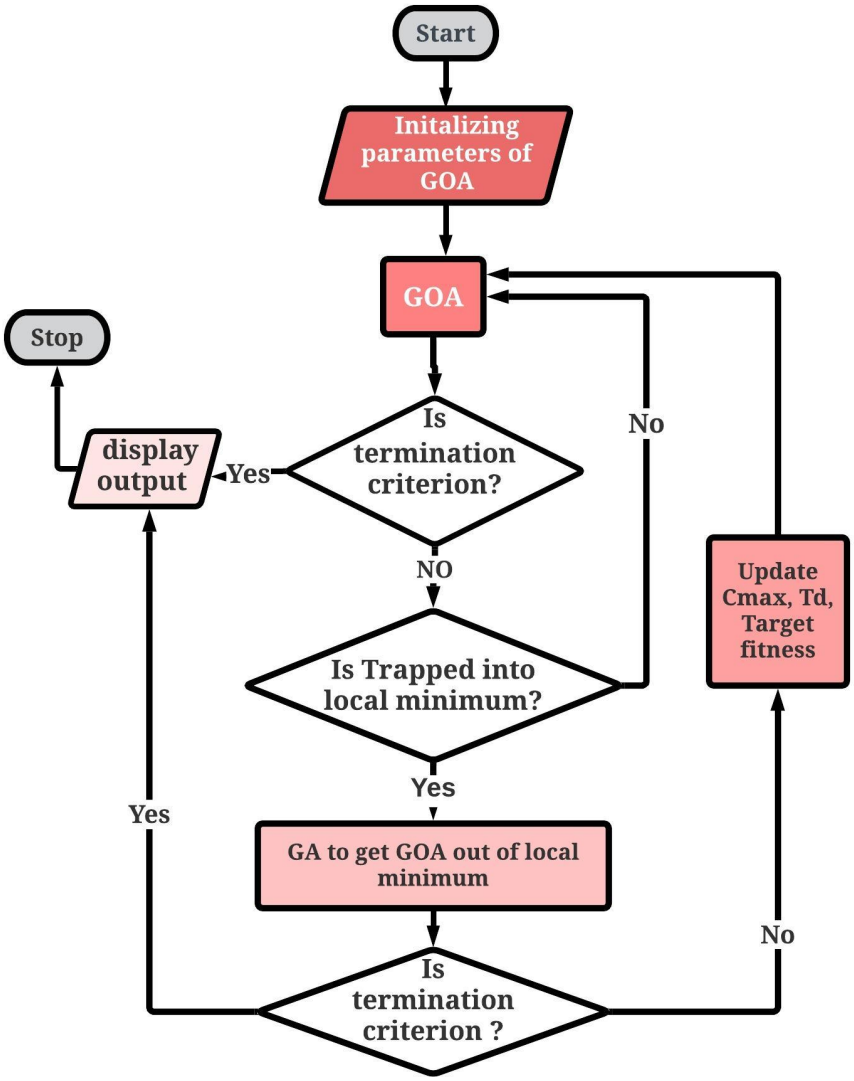


Fig.2. Flowchart of the Proposed Algorithm

3.3.1 GA stage to get GOA out of the local minimum

In this stage, the optimization problem, i.e. variables and objective functions, are defined based on certain assumptions and mathematical formulations. As shown in Figure 3, suppose GOA stuck at a local minimum point with a target position X_1, fitness value of F_1, and value of the parameter $C = C_1$. GA aims to transfer the search process to a better point to get a new $\widehat{T_d}$ and target fitness. Assuming the next point yielded from GA is (C_2, F_{2P}) as shown Figure 3. Hence, according to the figure, the value of C will change. Where, C_2 is the new value of parameter C and , F_{2P} is the preasumed enhanced obective value.

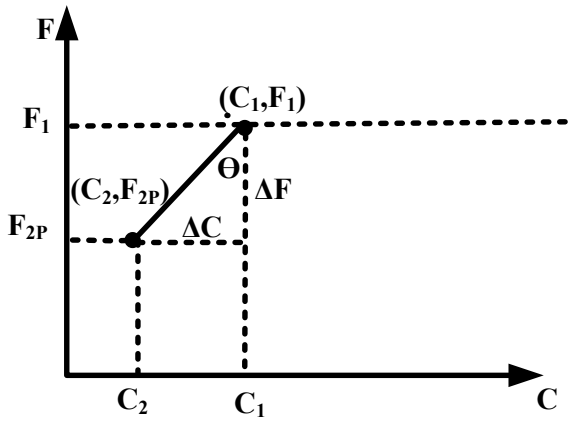


Fig.3. Predicted objective function against change in *C*

According to Figure 3, The change in parameter *C* can be calculated as:

$$\Delta C = \Delta F tan(\theta) \tag{12}$$

As $F_{2P} < F_1$, it can be expressed as:

$$F_{2P} = \alpha F_1,\ \alpha \in [0,1) \tag{13}$$

Then, the presumed change in fitness function can be calculated as:

$$\Delta F = F_{2P} - F_1 = \alpha F_1 - F_1 = (\alpha - 1)F_1 \tag{14}$$

According to the values of $\{\theta, \alpha\}$, the change in parameter *C* can be determined and the new value of *C* can be calculated as:

$$C_2 = C_1 + \Delta C \tag{15}$$

To ensure that the new point is always better than the previous one, the domain of the angle (θ) is restricted to the interval $[\pi, 2\pi]$.

As shown in Figure 4 if the old point has target position (solution) of $X_1 = [X_{11}X_{21} \dots \dots \dots X_{n1}]$ at $C = C_1$ where, *n* is the number of variables in the system.

At the new point where $C = C_2$, the value of each variable can increase or decrease according to the value of assumed angle (ß). So that the new candidate solution can be calculated as:

$$\Delta X_1 = \Delta C \tan(\beta) \tag{16}$$

$$X_{12} = X_{11} + \Delta X_1 \tag{17}$$

For n variables in the system the new component of candidate solution or new point can be calculated as:

$$X_{i2} = X_{i1} + \Delta X_i = X_{i1} + \Delta C \tan(\beta_i)\, for i = 1,2, \dots \dots, n \quad (18)$$

The new solution position will be:

$$X_2 = [\, X_{12} X_{22} \dots . X_{n2}] \quad (19)$$

Then, the value of actual fitness value will be calculated by substituting with the new point X_2 in the objective function (*F)* of the original system. It can be represented as:

$$F_{2A} = F(X_2) \quad (20)$$

According to the previous equations and assumptions, the variables in the genetic algorithm will be$[\theta, \alpha, \beta_1, \beta_2, \dots, \beta_n]$ where$\theta \in [\pi, 2\pi], \alpha \in [0,1)$, and $\beta_i \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

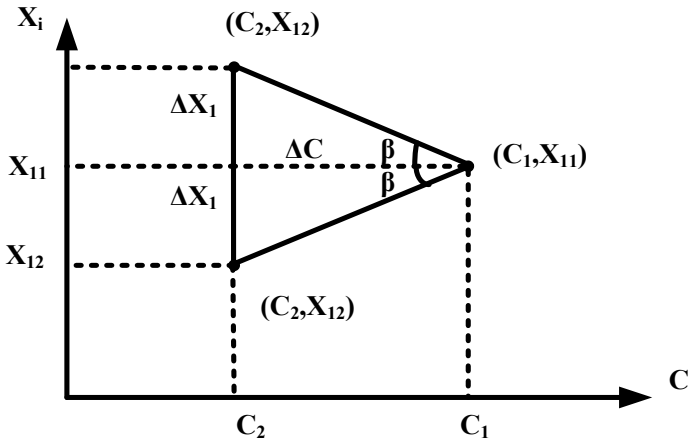


Fig.4. Predicted change in X_1 according to change in *C*

The parameters in GA will be adapted to enforce the value of actual fitness to be equal to the presumed(desired) improved fitness value yielded from Eq.13. The objective function of GA can be represented as:

$$\min_{\theta, \alpha, \beta_i, (i=1:n)} \|F_{2A} - F_{2P}\| \quad (21)$$

After applying GA, if the new target fitness achieved the termination criterion, the algorithm will terminate, and the final solution is displayed. Else, the new value of target position (X_2) will be entered as $\widehat{T_d}$, the new fitness value (F_{2A}) will be the target fitness and the new value of parameter (C_2)will be the value of parameter C_{max} and they will be entered to GOA. The new parameters which are introduced to GOA are $\widehat{T_d} = X_2$, TargetFitness $= F_{2A}$, and $C_{max} = C_2$.

Then, the calculations of GOA will start again, and the parameter C will decrease according to the formula of Eq.11 in the classical GOA based on the updated value of C_{max}.

4. Numerical Results and Discussions

In this section, 19 test functions are used to investigate the performance of the proposed algorithm. These test functions are unimodal, multimodal, and composite. The mathematical formulation of these test functions is available in [16].The results are compared with those obtained by the original GOA to show the benefits of the proposed modifications, their influence on reaching an optimal solution. Furthermore, 5 systems of nonlinear equations are solved as case study for MGOA [40]. The proposed algorithm is coded in MATLAB. 2020.a, implemented on the computer with Intel(R) Core (TM) i7 CPU, 3.2 GHz, and 16 GB RAM.

For computational studies, a population size equals 20 and50 generations are used, crossover fraction is 0.8, migration interval is 20, migration factor is 0.2 and all other parameters and functions were as default in (GA Tool) in MATLAB. 2020.a.Also, the termination criterion for both algorithms (GOA and MGOA) is defined as:

$$\delta = \left| \left\| F_{optimum} \right\| - \left\| F_l \right\| \right| \leq Tolerence = 1e-06; \quad (22)$$

where $\left\| F_{optimum} \right\|$ is the optimum value of the objective function which is (0) in all test functions and nonlinear system cases. While, $\| F_l \|$ is the calculated objective function at each iteration l.

It should be noted here that, both algorithms (GOA and MGOA) have the same maximum number of iterations which was identified to (7000) and all results have been taken from the first run. In addition, when either of them has met the termination criterion, the calculations were terminated, and the number of consumed iterations was displayed.

Furthermore, the mechanism of MGOA to decide whether GOA has been trapped into local minimum or not is to count a limit number of iterations which produced the same solution. This number was limited to 3 consecutive iterations in MGOA Finally, In Eq.13, F_{2P} is calculated as $F_{2P} = \alpha F_1$ where $\alpha \in [0,1)$. The upper limit of the domain of α should not be exactly 1 ($\alpha < 1$) to avoid continuous trapping into the same local minimum point and wasting more iterations without significant improvement in the objective function. In the MGOA, the domain of α was defined as [0,0.9].It should be mentioned here, the value of parameter C_{max} was fixed in GOA and initiated to MGOA as 1 (would change according to GA stage) and the value of C_{min} was 1e-05 in both algorithms.

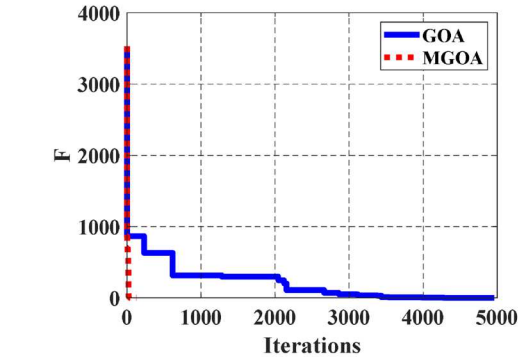
4.1 Results

Table 1 shows the results of unimodal functions where it presents comparing between the original GOA and the proposed MGOA in terms of their obtained solution and number of iterations. While Figure 5 show the convergence curves of the function value. Besides, in test function (F_5)the resulted objective values were too large so, they were rationalized by the maximum reached value (F_{max}) to show the results of both algorithms more clearly.

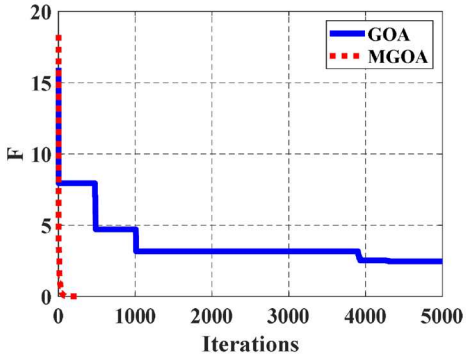
As shown in the table and figures, MGOA found the optimal solutions after smaller number of iterations than GOA. The details of iterations in MGOA was represented in the table. For each function, both numbers of iterations using GOA and MGOA were displayed in the table. For example, in F1 the GOA found the solution after 4960 iterations while MGOA found the solution after 131 iterations where 37 of them were using GA stage to overcome trapping into local minimum. In addition, it is evident in some cases that GOA could not find a solution until it reached the maximum number of iterations as in cases of test functions (F_2 and F_7). All results and figures illustrate how MGOA converged faster and more reliable than original GOA.

Table 1: Results of unimodal functions

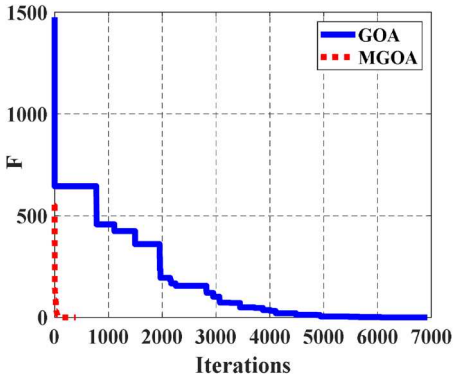
Unimodal functions		**GOA**			**MGOA**				
No.	**Optimum value**	**Found the solution**	$\|\|F\|\|$	**Number of iterations**	**Found the solutions**	$\|\|F\|\|$	**Number of iterations**	**GOA iterations**	**GA iterations**
F_1	**0**	Yes	1.2×10^{-6}	4960	Yes	9.9×10^{-7}	131	94	37
F_2	**0**	No	2.337	7000	Yes	6.7×10^{-7}	290	195	95
F_3	**0**	Yes	8.6×10^{-7}	6920	Yes	7.8×10^{-7}	394	263	131
F_4	**0**	Yes	8.9×10^{-7}	6999	Yes	4.7×10^{-7}	262	178	84
F_5	**0**	Yes	0.201	7000	Yes	1.6×10^{-7}	346	231	115
F_6	**0**	Yes	8.1×10^{-7}	6642	Yes	7.6×10^{-7}	263	176	87
F_7	**0**	No	0.012	7000	Yes	4.2×10^{-7}	80	54	26
F_8	**0**	Yes	4.9×10^{-7}	6998	Yes	1.1×10^{-7}	39	27	12



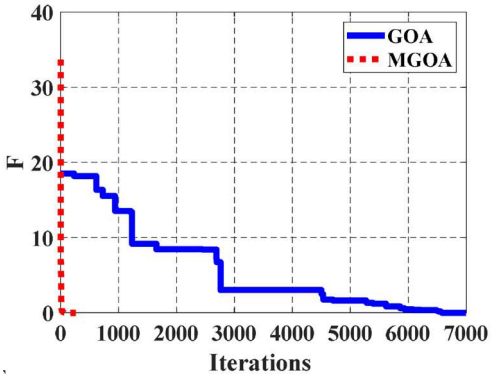
F_1



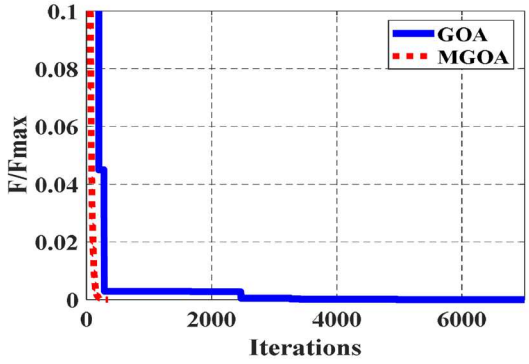
F_2



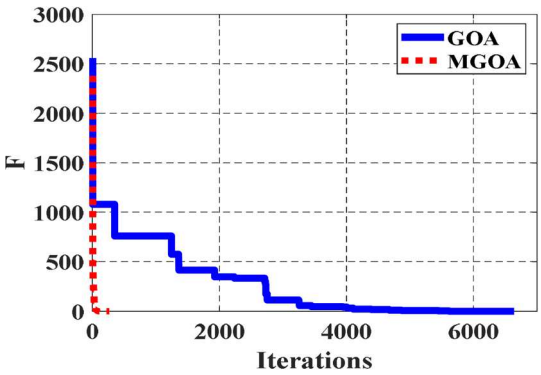
F_3



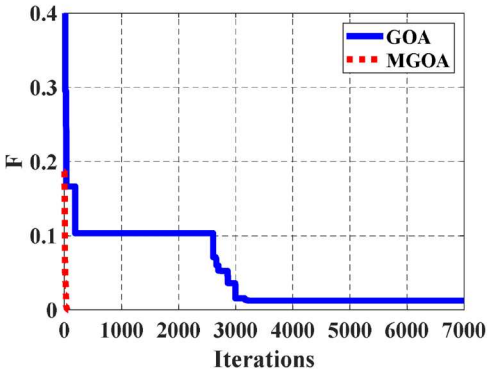
F_4



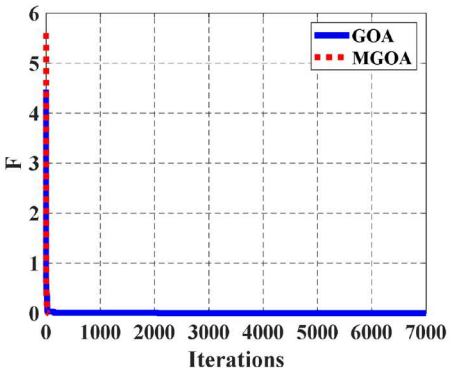
F_5



F_6



F_7



F_8

Fig. 5. Convergence curves for unimodal functions

For the multimodal test functions, Table 2 shows the results of these functions where it presents the results of both GOA and MGOA. Figure 6 show the convergence curves of the function value of these test functions. To show the convergence curves of the two algorithms more clearly, the figures were drawn as *r* versus *F*; where *r* is ratio between the iteration and the whole number of iterations for each algorithm, and *F is* the value of the objective function. It should be mentioned here, *r* was calculated after the termination of each algorithm and according to its results. The ratio *r* is represented as:

$$r = \frac{iteration}{The\ whole\ number\ of\ iterations\ consumed\ by\ both\ algorithm} \tag{23}$$

As shown in Table 2 and Figure 6, MGOA detect the solution consuming smaller number of iterations than successful trials of original GOA. In addition, the horizontal segments in convergence curves of GOA show the wasted iterations in remaining in the same local minimum. On the other hand, MGOA overcame this drawback and reached to an optimum solution by continuously minimizing the objective function. The results and figures illustrate the effectiveness of the proposed GA stage with its parameters and objective function to improve the performance of GOA.

Table 2: Results of multimodal test functions

Multimodal **functions**		**GOA**			**MGOA**				
No.	**Optimum value**	**Found the solution**	$\|\|F\|\|$	**Number of iterations**	**Found the solutions**	$\|\|F\|\|$	**Number of iterations**	**GOA iterations**	**GA iterations**
F_9	**0**	No	3.07	7000	Yes	9.5×10^{-7}	416	278	138
F_{10}	**0**	Yes	7.4×10^{-6}	7000	Yes	6.3×10^{-7}	278	186	92
F_{11}	**0**	No	0.07	7000	Yes	9.02×10^{-7}	416	278	138
F_{12}	**0**	Yes	8.9×10^{-7}	6458	Yes	2.8×10^{-7}	218	146	72
F_{13}	**0**	Yes	9.4×10^{-7}	6865	Yes	7.7×10^{-7}	323	216	107

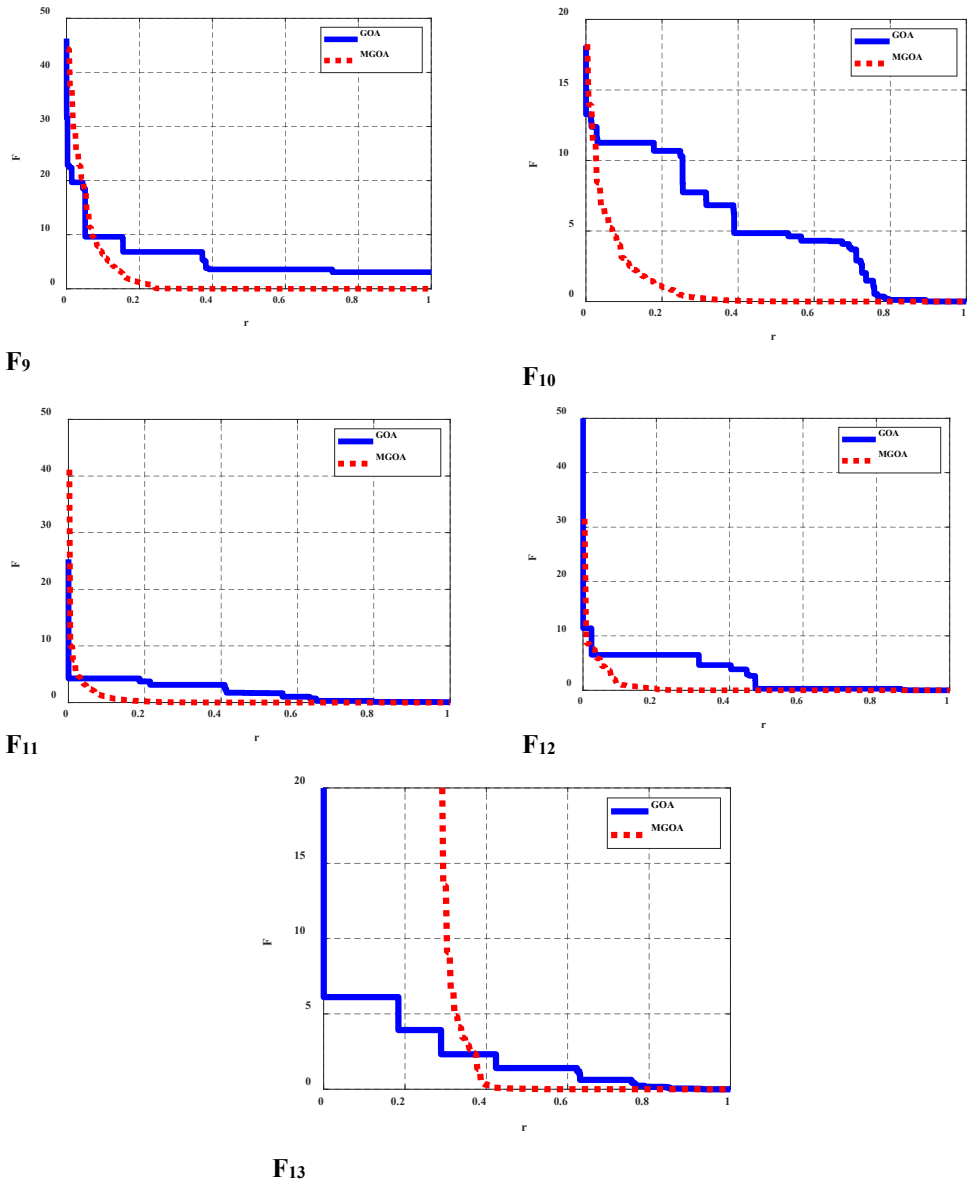


Fig. 6. Convergence curve for multimodal test functions

For the composite test functions, Table 3 shows the results of these functions. Figure 7 show the curves of objective value of absolute error function (F) versus the parameter (C)in both algorithms. The aim of plotting this relation is to illustrate the performance of MGOA in changing the parameter C_{max} to enhance the convergence to the optimum solution. On the contrary, the figures show that the classical decreasing of C_{max} leads to trapping into local minimum. For example, in test function $F_{17,}$ in GOA the value of parameter C_{max} was 1 and it decreases according to Eq.11. However, in MGOA it starts with 1 until the GOA fallen into local

minimum then, its domain was edited using GA stage, C_{max} increased to 1.05 then, it started decreasing again from this updated value according to Eq.11. While, in test function $F_{18,}$ the value C_{max} continuously decreased until it reached almost 0.85 then, GOA trapped into a local minimum and GA stage was needed. After applying GA stage, the value of C_{max} was updated to almost 0.95 and introduced with other parameters to GOA again. However, in test functions F_{15} and $F_{16,}$ the value of C_{max} was updated by decreasing but in different slope than resulted by Eq.11. It is worthy here to state that, the main target of GA stage is to update the value of C_{max} whatever increasing or decreasing to escape from the local optima. Finally, both table and figures show that how the proposed MGOA led to continuous decreasing of the objective function value until the desired tolerance and optimum solution were achieved.

Table 3: Results of Composite test functions

Composite functions		**GOA**			**MGOA**				
No.	**Optimum value**	**Found the solution**	$\|\|F\|\|$	**Number of iterations**	**Found the solutions**	$\|\|F\|\|$	**Number of iterations**	**GOA iterations**	**GA iterations**
F_{14}	**0**	No	0.99	7000	Yes	9.9×10^{-7}	399	267	133
F_{15}	**0**	No	0.21	7000	Yes	9.7×10^{-7}	239	160	79
F_{16}	**0**	Yes	2.5×10^{-7}	4166	Yes	1.1×10^{-7}	68	46	22
F_{17}	**0**	No	0.39	7000	Yes	9.8×10^{-7}	373	249	124
F_{18}	**0**	No	3	7000	Yes	9.2×10^{-7}	434	290	144
F_{19}	**0**	Yes	3.7×10^{-5}	7000	Yes	9.7×10^{-7}	45	31	14

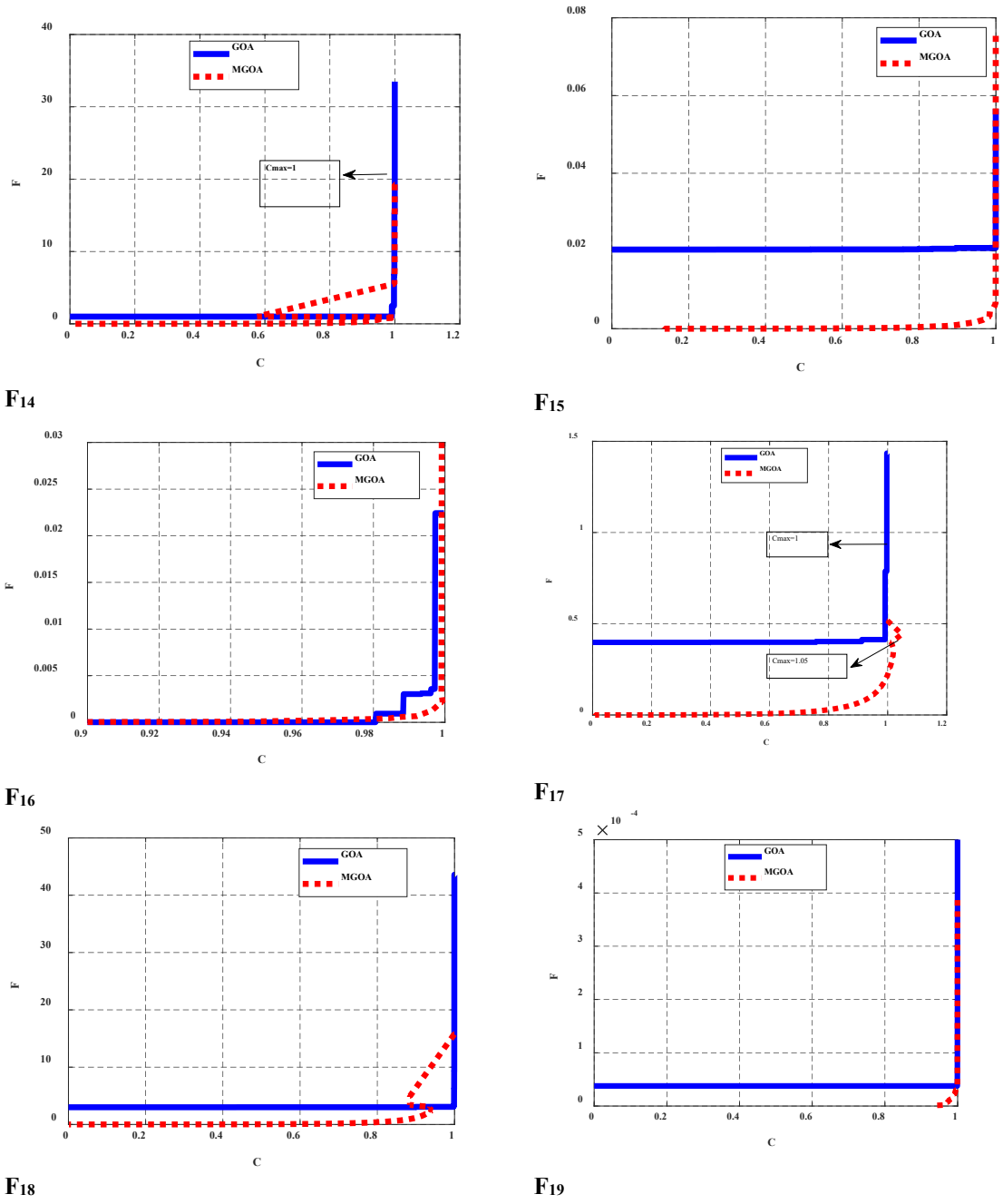
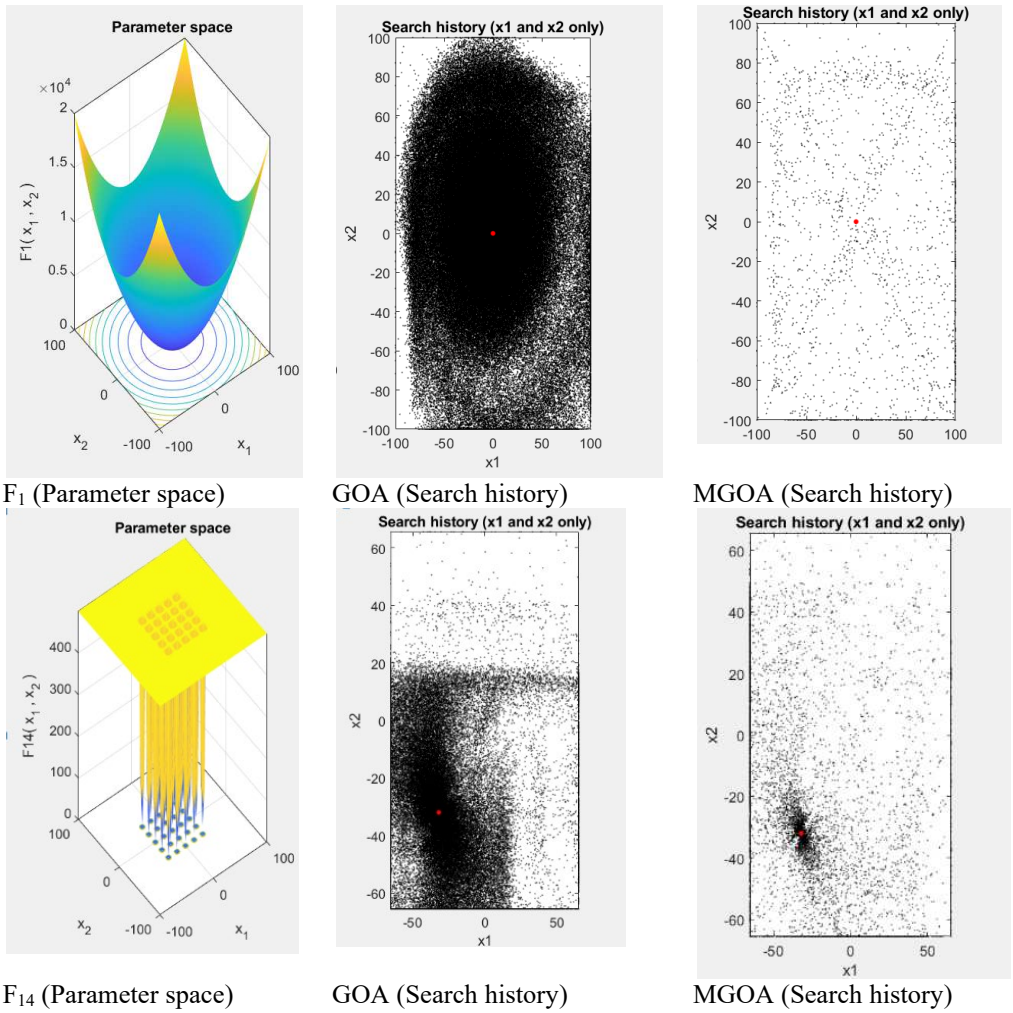


Fig. 7. Change of *F* versus *C* for composite test functions

4.2. Result analysis and Discussion

In this subsection, the important remarks on the proposed algorithm and analysis of results are discussed.

1- MGOA does not use GOA and GA to solve the optimization problem and to find an optimal solution. But, the main algorithm is GOA and when it falls into a local minimum, the GA stage intervenes to get it out of the local minimum and transfer it to a better solution. Then GOA continues the search process again based on the updated values of the new parameters (C_{max}, $\widehat{T_d}$ and target fitness) yielded from the GA stage.
2- GA operations did not waste time or iterations as seen in results where the total number of iterations in MGOA including GA and GOA was smaller than the iterations using the original GOA.
3- liberating the domain of parameter C_{max} enables MGOA to get GOA out of the local minimum and to achieve a balance between exploration and exploitation.
4- In addition, GA is not concerned with solving the original optimization problem since it is only used when GOA is fallen into a local minimum So, in MGOA, GA acts as a local search algorithm, not a global one.
5- For further illustration, Figure 8 shows the parameter space and search history for the two test functions F_1 and F_{14} using GOA for MGOA.
    a) It is obvious from Figure 8 that the GOA distribution of grasshoppers around the solution was so condensed because of remaining in local minimum for large periods or a large number of iterations. Thus, GOA in F_1 was able to find a solution after many iterations. While, for F_{14}, it was unable to detect a solution because of trapping into the local minimum and spending all iterations at the same point.
    b) On the contrary, MGOA got rid of several local minimum points and continues to explore the search space until an optimum solution is found for both the F_1 and F_{14} test functions in a smaller number of iterations. In addition, as trapping is eliminated at the local minimum as quickly as possible using the GA stage, the distribution of grasshoppers is more widespread around the optimal solution point, allowing MGOA to explore and exploit more quickly.



F_1 (Parameter space) GOA (Search history) MGOA (Search history)

F_{14} (Parameter space) GOA (Search history) MGOA (Search history)

Fig. 8. Parameter space and search history for F_1 and F_{14} using GOA for MGOA

4.3. Case Study: Solving nonlinear systems of equations

In this subsection, MGOA will be applied on various types of nonlinear systems of equations. All details of these nonlinear systems can be found in [40]. First, the system of nonlinear equations is converted to an optimization problem according to Eq. 2. Then, this optimization problem is solved by the modified MGOA. The results will be compared Newton's method (NM) and Levenberg Marquardt algorithm (LMA) [41-43].In each case *n* represents the number of variables however, *m* is the number of equations. The descriptions of these nonlinear equation systems are as follows:

1- **Freudenstein and Roth function:** This system is square with$n = m = 2$,

$$\begin{aligned} f_1(x) &= -13 + x_1 + \left((5 - x_2)x_2 - 2\right)x_2 \\ f_2(x) &= -29 + x_1 + \left((x_2 + 1)x_2 - 14\right)x_2 \end{aligned} \tag{24}$$

2- **Kowalik and Osborne function:** This system is non-square with$n = 4, m = 11$,

$$f_i(x) = y_i - \frac{x_1\left(u_i^2 + u_i x_2\right)}{u_i^2 + u_i x_3 + x_4}, \quad i = 1, 2, ..., m; \tag{25}$$

where $y_i's$ and $u_i's$ are constants, their definition can be found in [40].

3- **Biggs EXP6 function:** This system is square with $n = 6, m = n$,

$$\begin{aligned} f_i(x) &= x_3 e^{-t_i x_1} - x_4 e^{-t_i x_2} + x_6 e^{-t_i x_5} - y_i, \\ y_i &= e^{-t_i} - 5e^{-10t_i} + 3e^{4t_i},\ i = 1, 2, \ldots, m; \end{aligned} \tag{26}$$

where$t_i = 0.1i$.

4- **Osborne 1 function:** This system is non-square with $n = 5, m = 33$,

$$\begin{aligned} f_i(x) &= y_i - \left(x_1 + x_2 e^{-t_i x_4} + x_3 e^{-t_i x_5}\right),\ \ i = 1, 2, \ldots, m, \\ t_i &= 10(i - 1); \end{aligned} \tag{27}$$

where $y_i's$are constants, its values can be found in [40].

5- **Penalty function II**: This system is non-square with $n = 10, m = 2n$,

$$\begin{aligned} f_1(x) &= x_1 - 0.2, \\ f_i(x) &= a^{1/2}\left(e^{x_i/10} + e^{x_{i-1}/10}\right) - y_i,\ \forall\ 2 \le i \le n, \\ f_i(x) &= a^{1/2}\left(e^{x_{i-n+1}/10} - e^{i/10}\right),\ \forall\ n < i < 2n, \\ f_{2n} &= \left(\sum_{j=1}^{n}(n - j + 1)x_j^2\right) - 1, \\ y_i &= e^{i/10} + e^{(i-1)/10},\ a = 10^{-5}. \end{aligned} \tag{28}$$

Table.4 shows the results of these nonlinear systems using MGOA compared with Newton's method (NM) and Levenberg-Marquardt (LM) algorithm as the most popular conventional algorithms in solving nonlinear systems. While Figure 9 shows the convergence curves of the three algorithms.

Table. 4 Results of application systems

Nonlinear system of equations	Algorithm	Found solution	$\|\|F\|\|$	iterations
Freudenstein and Roth function:	**MGOA**	Yes	0	150
	NM	No	0.02	1000
	LM	No	6.99	1000
Kowalik and Osborne function	**MGOA**	Yes	9.4×10^{-09}	427
	NM	No	0.025	1000
	LM	No	0.074	1000
Biggs EXP6 function	**MGOA**	Yes	7.4×10^{-09}	161
	NM	No	0.04	1000
	LM	No	0.1	1000
Osborne 1 function	**MGOA**	Yes	9.4×10^{-09}	544
	NM	No	0.54	1000
	LM	No	1.53	1000
Penalty function II	**MGOA**	Yes	9.1×10^{-09}	596
	NM	No	0.02	1000
	LM	No	0.017	1000

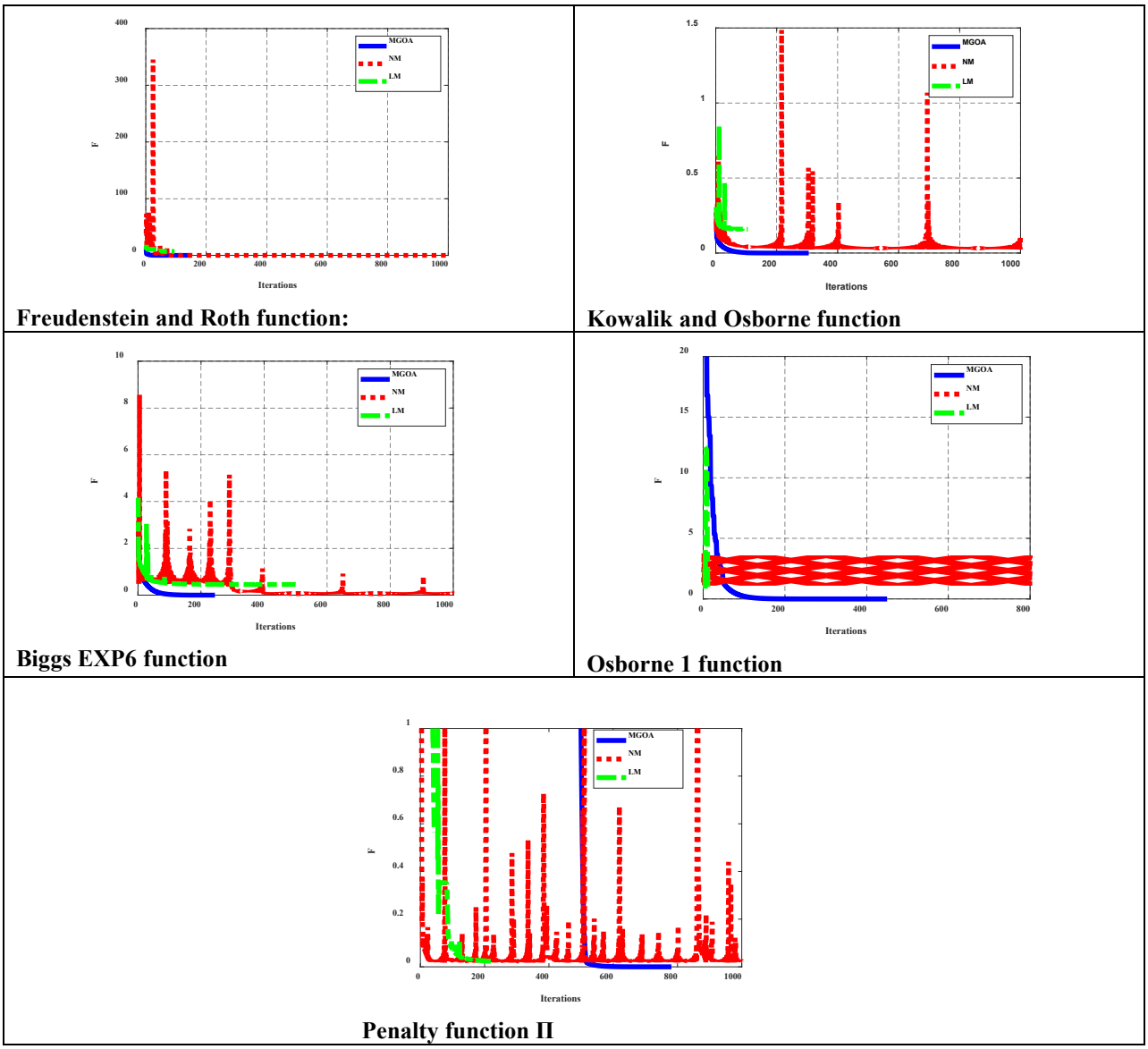


Fig. 9. Convergence curves for solving nonlinear system of equations

Table 4 and Figure 9 show the ability of the proposed algorithm MGOA to detect a solution despite the singularity of these systems because it does not depend on the derivatives. Both NM and LM failed to obtain the solution in addition to calculation burden due to calculations of the jacobian matrices. This indicates the ability of MGOA to solve even singular square or non-square nonlinear systems effectively.

5. Conclusions

In this paper, a modified grasshopper optimization algorithm (MGOA) is proposed, where the classic grasshopper optimization algorithm (GOA) is modified based on a genetic algorithm (GA). The aim of the modification is to take out GOA from trapping into local minimum and to improve both the convergence rate and the speed by reducing the number of repeated solutions and decreasing the number of iterations required. The modifications based on some mathematical assumptions and relations which depend on releasing the calculations of the parameter C to move to a new improved point in the search space. MGOA was tested on the same 19 test functions that were solved by the original GOA to investigate and verify the influence and responses of the proposed modifications. The results showed a significant improvement in the acceleration of convergence and the success of MGOA in finding the optimal solution compared to the classic GOA. In addition, MGOA was used to solve 5 applications of nonlinear systems based on transforming their equations into a single unconstrained optimization problem with the objective function of minimizing overall residual function. The MGOA demonstrated distinctive success in solving the various cases of square, non-square, and singular non-linear systems, whereas classical methods such as Newton's method or the Levenberg-Marquardt algorithm failed to detect the solution. All results and comparisons ensure the effectiveness, reliability, and validity of the modified algorithm.

In future works, the proposed approach can be modified to solve other optimization problems such as constrained optimization problems, nonlinear bilevel programming problems, interval quadratic programming problems, data clustering problems, etc. Hence large-scale engineering problems such as resource allocation issues, economical load transmission problems, unit commitment issues, wind farm optimization of wind turbines, real-time applications, etc. can be considered. Finally, the proposed algorithm can be developed in order that it can solve multiobjective optimization problems.

Data Availability

All data used to support the findings of this study are included within the article.

Funding Statement

The authors received no specific funding for this work.

Compliance with ethical standards

I. Conflict of interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

II. Ethical approval

This study is not supported by any sources or any organizations.

III. Human and animal rights

This article does not contain any studies with human participants or animals performed by any of the authors.

Authors' Contributions

Conceptualization, M.A.E.; Investigation, M.A.E. and H.A.O.; Methodology, H.A.O.; Writing—original draft, H.A.O.; Writing—review & editing, M.A.E. and H.A.O.; M.A.E. and H.A.O. have read and agreed to the published version of the manuscript.

References

[1] A.A. Mousa, M.A. El-Shorbagy and M. A. Farag, "K-means-Clustering Based Evolutionary Algorithm for Multi-objective Resource Allocation Problems," *Applied Mathematics & Information Sciences*, vol. 11, no. 6, pp. 1-12, 2017.

[2] Ali M. Abdelsalam and M.A. El-Shorbagy, "Optimization of wind turbines siting in a wind farm using genetic algorithm based local search," *Renewable Energy*, vol. 123, pp. 748-755, 2018.

[3] M. Dorigo and T. Stützle, "Ant Colony Optimization," MIT Press, Cambridge, 2004.

[4] M.A. El-Shorbagy and Aboul Ella Hassanien, "Particle Swarm Optimization from Theory to Applications," *International Journal of Rough Sets and Data Analysis (IJRSDA),* vol. 5, no. 2, pp. 1-24, 2018.

[5] M.A. El-Shorbagy and A.A. Mousa, "Chaotic Particle Swarm Optimization for Imprecise Combined Economic and Emission Dispatch Problem," *Review of Information Engineering and Applications*, vol. 4, no. 1, pp. 20-35, 2017.

[6] A.A. Mousa and M.A. El-Shorbagy, "Enhanced Particle Swarm Optimization Based Local Search for Reactive Power Compensation Problem," *Applied Mathematics*, vol. 3, pp. 1276-1284, 2012.

[7] D. Karaboga, "An Idea Based On Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, vol. 200, pp. 1-10 2005.

[8] S.-C. Chu, P.-W.Tsai and J.-S. Pan, "Cat swarm optimization," In *Pacific Rim* international *conference on artificial intelligence*, Springer, Berlin, Heidelberg, 2006, pp. 854-858.

[9] W. Zhao and L. Wang, "An effective bacterial foraging optimizer for global optimization," *Information Sciences*, vol. 329, pp.719-735, 2016.

[10] M.A. Elsisy, D.A. Hammad and M.A. El-Shorbagy, "Solving Interval Quadratic Programming Problems by Using the Numerical Method and Swarm Algorithms," *Complexity*, vol. 2020, pp 1 – 11, Sep. 2020.

[11] M. Marinaki and Y. Marinakis, "A Glowworm Swarm Optimization Algorithm for the Vehicle Routing Problem with Stochastic Demands," *Expert Systems with Applications*, vol. 46, pp. 145-163, 2016.

[12] A. L. Bolaji, M. A. Al-Betar, M. A. Awadallah, A. T. Khader and L. M. Abualigah, "A comprehensive review: Krill Herd algorithm (KH) and its applications," *Applied Soft Computing*, vol. 49, pp. 437–446, 2016.

[13] Y. Zhou, X. Chen and G. Zhou, "An improved monkey algorithm for a 0-1 knapsack problem," *Applied Soft Computing*, vol. 38, pp. 817-830, 2016.

[14] Y. Abo-elnaga and M.A. El-Shorbagy, "Multi-Sine Cosine Algorithm for Solving Nonlinear Bilevel Programming Problems," *International* Journal *of Computational Intelligence Systems*, vol. 13, no. 1, pp 421–432, 2020.

[15] M. Shehab, A. T. Khader, M., Laouchedi and O. A. Alomari, "Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization," *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2395-2422, 2019.

[16] S. Saremi, S. Mirjalili and A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application," *Advances in Engineering* Software, vol. 105, pp. 30–47, 2017.

[17] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163-191, 2017.

[18] S. Li., H. Chen, M.Wang, A. A. Heidari and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," Future Generation Computer Systems, vol. 111, pp. 300-323, 2020.

[19] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, "Harris hawks optimization: algorithm and applications," Future Generation Computer Systems, vol. 97, pp. 849-872, 2019.

[20] I. Ahmadianfar, O. Bozorg-Haddad, X. Chu, "Gradient-based optimizer: A new Metaheuristic optimization algorithm," *Information Sciences*, vol. 540, pp. 131-159, 2020.

[21] M.A. El-Shorbagy, A. A. Mousa, Constrained Multiobjective Equilibrium Optimizer Algorithm for Solving Combined Economic Emission Dispatch Problem, Complexity, Jan. 2021, Volume 2021, pp. 1 - 14.

[22] A. A. Mousa, M.A. El-Shorbagy, I. Mustafa, H. Alotaibi, Chaotic Search Based Equilibrium Optimizer for Dealing with Nonlinear Programming and Petrochemical Application, Processes, Jan. 2021, Volume 9, Issue 2, p. 200.

[23] Algamal, Z. Y., Qasim, M. K., Lee, M. H., & Ali, H. T. M. Improving grasshopper optimization algorithm for hyperparameters estimation and feature selection in support vector regression. Chemometrics and Intelligent Laboratory Systems, Volume 208, 15 January 2021, 104196

[24] Feng H, Ni H, Zhao R, Zhu X. An Enhanced Grasshopper Optimization Algorithm to the Bin Packing Problem. Journal of Control Science and Engineering. 2020 Mar 31;2020.

[25] Alphonsa, MM Annie, and N. MohanaSundaram. "A reformed grasshopper optimization with genetic principle for securing medical data." Journal of Information Security and Applications 47 (2019): 410-420.

[26] Dwivedi S, Vardhan M, Tripathi S. An Effect of Chaos Grasshopper Optimization Algorithm for Protection of Network Infrastructure. Computer Networks. Volume 176, 20 July 2020, 107251

[27] Goel, N., Grover, B., Gupta, D., Khanna, A. and Sharma, M., 2020. Modified Grasshopper Optimization Algorithm for detection of Autism Spectrum Disorder. Physical Communication, Volume 41, August 2020, 101115.

[28] Sezavar A, Farsi H, Mohamadzadeh S. A Modified Grasshopper Optimization Algorithm Combined with CNN for Content Based Image Retrieval. International Journal of Engineering. 2019 Jul 1;32(7):924-930.

[29] Luo, Jie, Huiling Chen, Yueting Xu, Hui Huang, and Xuehua Zhao. "An improved grasshopper optimization algorithm with application to financial stress prediction." Applied Mathematical Modelling 64 (2018): 654-668.

[30] Purushothaman, R., S. P. Rajagopalan, and Gopinath Dhandapani. "Hybridizing Gray Wolf Optimization (GWO) with Grasshopper Optimization Algorithm (GOA) for text feature selection and clustering." Applied Soft Computing 96 (2020): 106651.

[31] Taher, M.A., Kamel, S., Jurado, F. and Ebeed, M., 2019. Modified grasshopper optimization framework for optimal power flow solution. Electrical Engineering, 101(1), pp.121-148.

[32] Wang, D., Chen, H., Li, T., Wan, J. and Huang, Y., 2020. A novel quantum grasshopper optimization algorithm for feature selection. International Journal of Approximate Reasoning, 127, pp.33-53.

[33] Zhao R, Ni H, Feng H, Song Y, Zhu X. An improved grasshopper optimization algorithm for task scheduling problems. Int. J. Innov. Comput. Inf. Control.. 2019 Oct;15(5):1967-87.

[34] P.Y. Nie, A null space method for solving system of equations, Appl. Math. Comput. 149 (1) (2004) 215–226.

[35] P.Y. Nie, An SQP approach with line search for a system of nonlinear equations, Math. Comput. Modell. 43 (3–4) (2006) 368–373.

[36] C. Grosan, A. Abraham, A new approach for solving nonlinear equations systems, IEEE Trans. Syst. Man Cybern.-Part A: Syst. Humans 38(3) (2008) 698–714.

[37] A. Pourrajabian, R. Ebrahimi, M. Mirzaei, M. Shams, Applying genetic algorithms for solving nonlinear algebraic equations, Applied Mathematics and Computation 219 (2013) 11483–11494.

[38] D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, MA, 1989.

[39] K.F. Man, K.S. Tang, S. Kwong, Genetic Algorithms: Concepts and Designs, Springer, London, (1999).

[40] Jorge J. More Burton S. Ga Rbow Kenneth E. Hillstrom, " Testing Unconstrained Optimization Software", Argonne National Laboratory Argonne, Illinois 60439 , Applied Mathematics Division Technical Memorandum No. 324, (1978).

[41] Éric Walter, "Numerical Methods And Optimization", Springer International Publishing Switzerland, (2014).

[42] Broyden, C.G.: "A Class Of Methods For Solving Nonlinear Simultaneous Equations.", Math. Comp. V. 19(92), 577–593, (1996).

[43] Jarry, P. And Beneat, J.N. "Appendix B Levenberg–Marquardt–More", Optimization Algorithm" , 239-244, (2016).

Figures

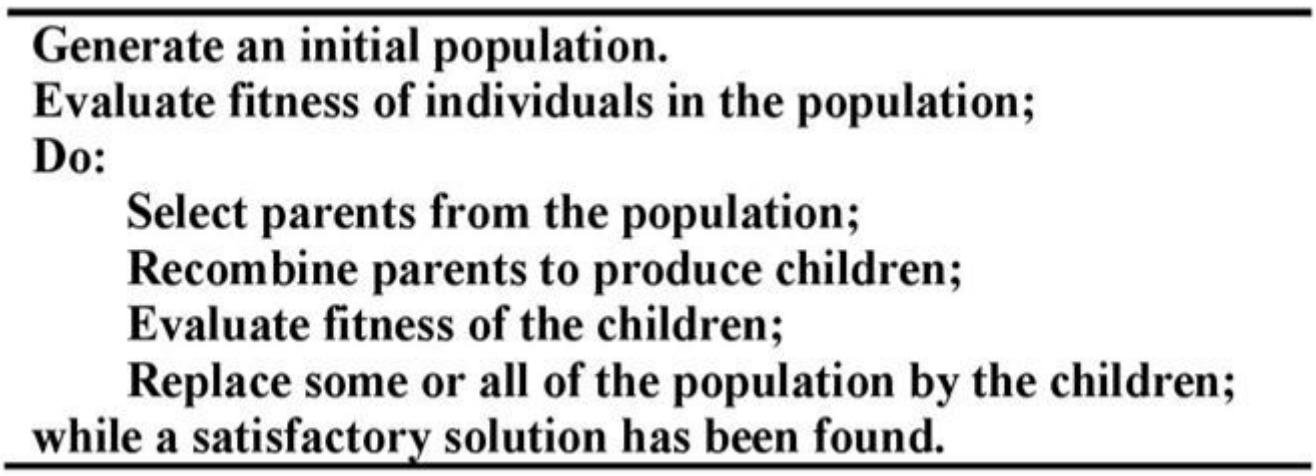


Figure 1

The general GA pseudo-code

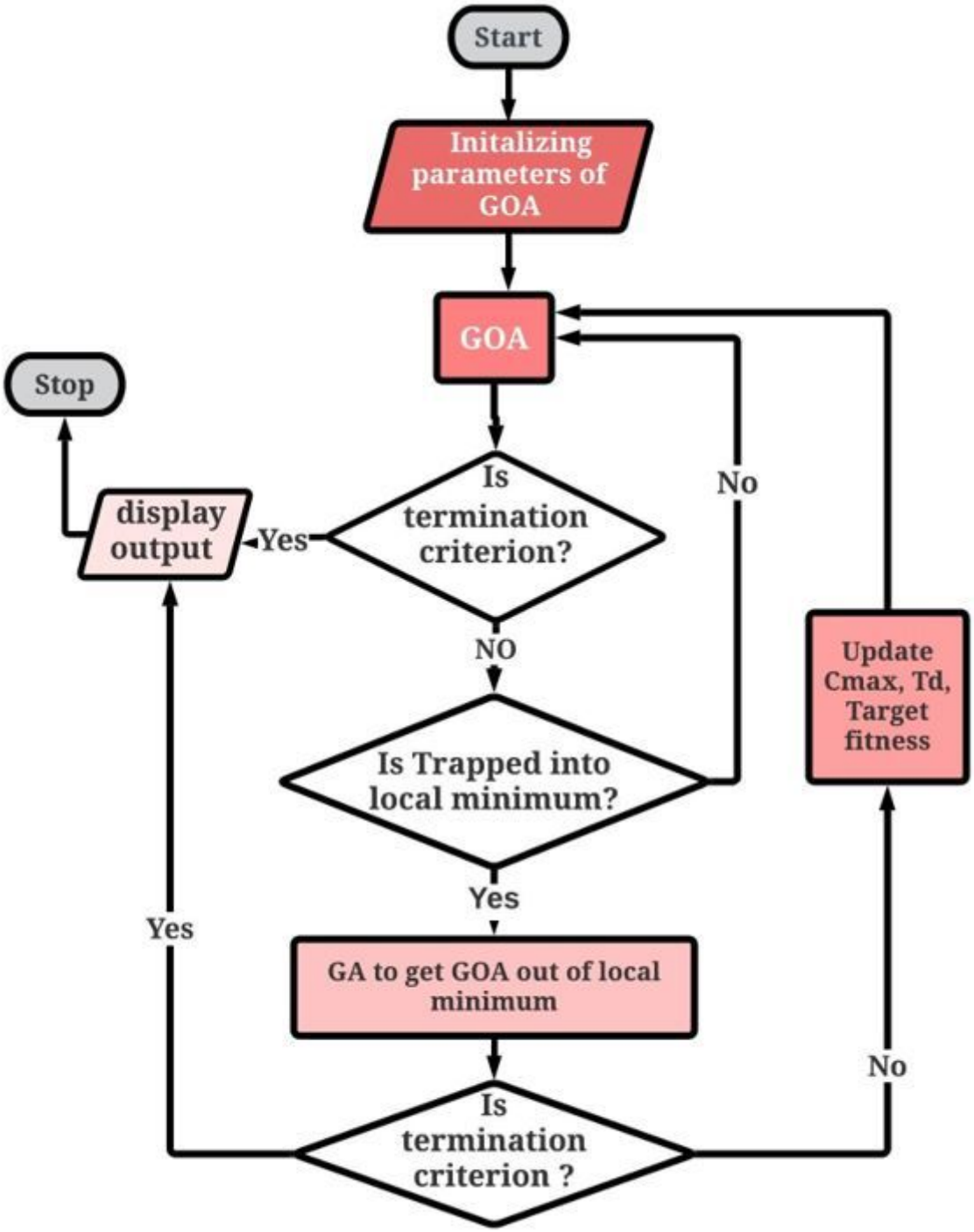


Figure 2

Flowchart of the Proposed Algorithm

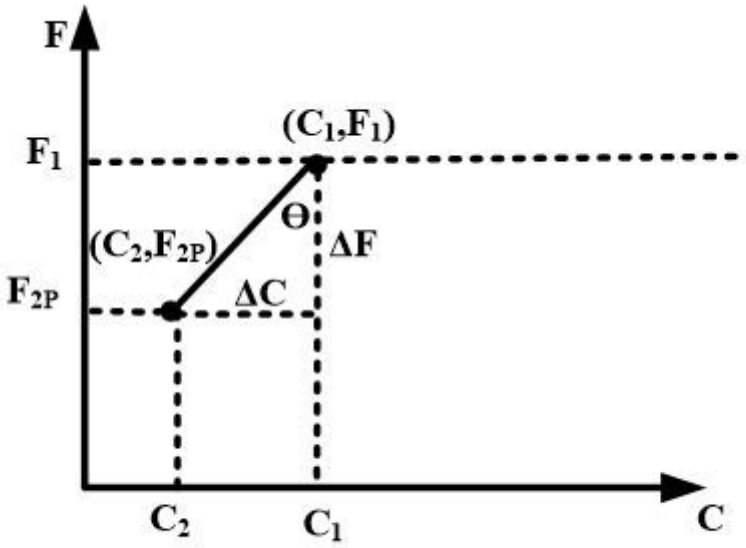


Figure 3

Predicted objective function against change in C

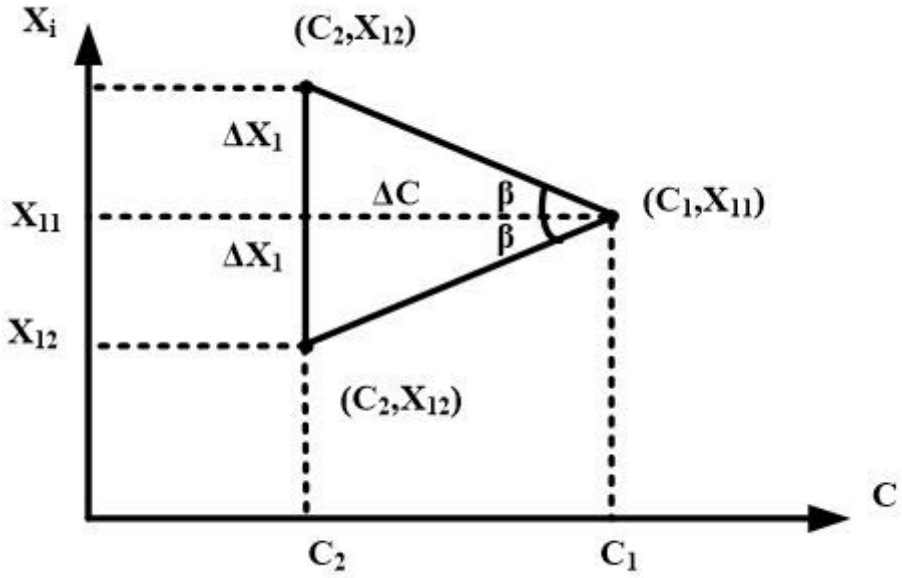


Figure 4

Predicted change in X_1 according to change in C

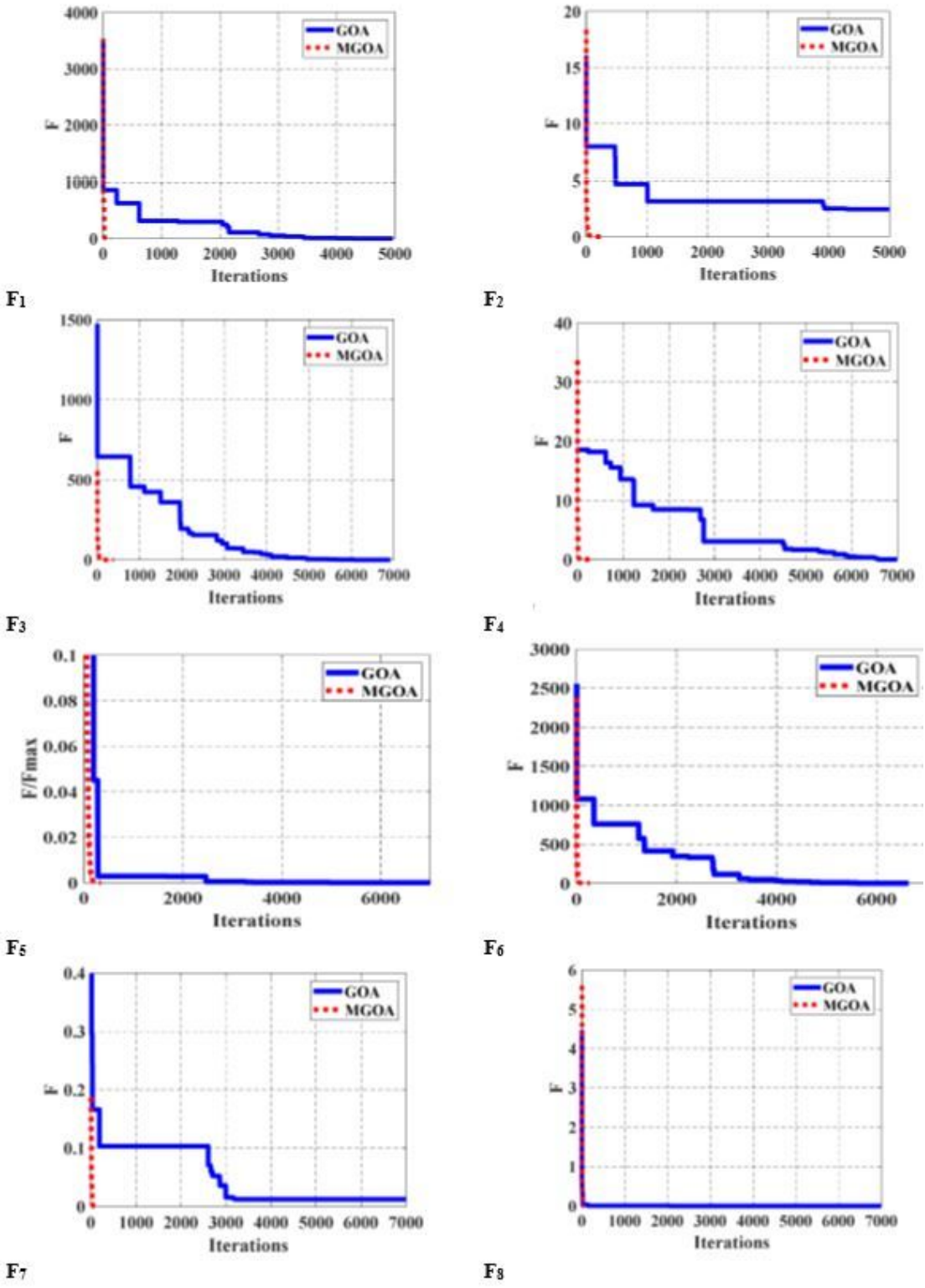


Figure 5

Convergence curves for unimodal functions

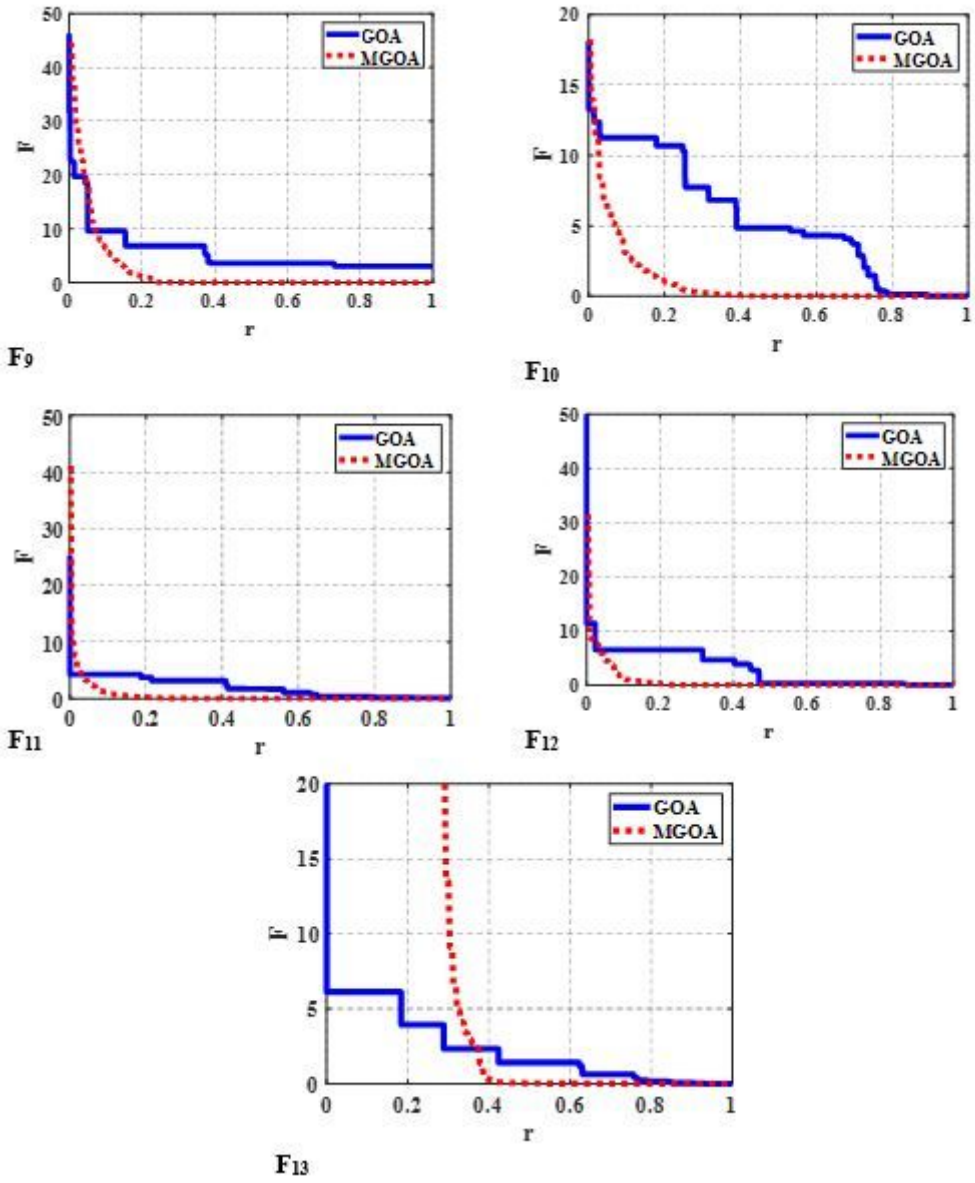


Figure 6

Convergence curve for multimodal test functions

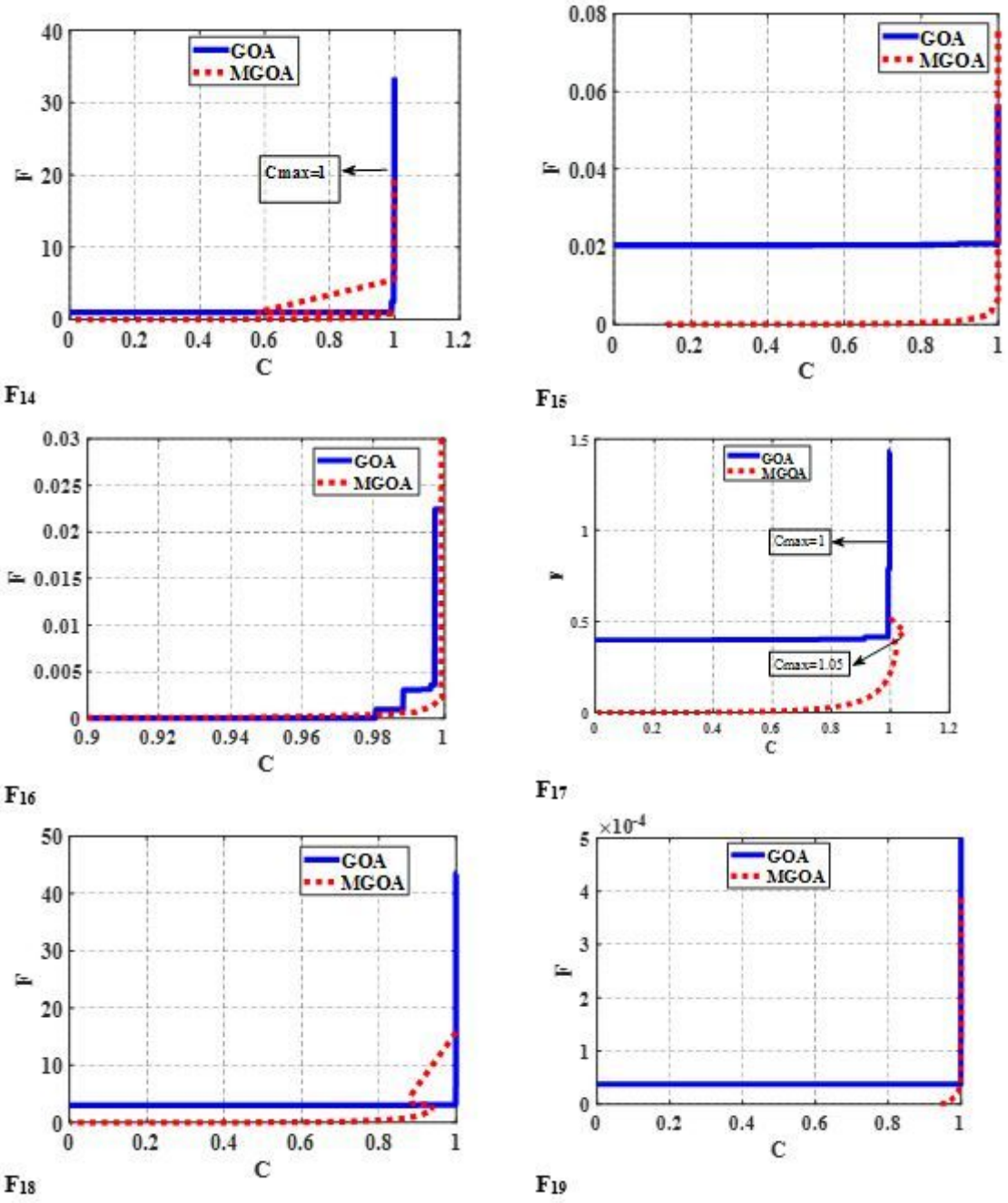


Figure 7

Change of F versus C for composite test functions

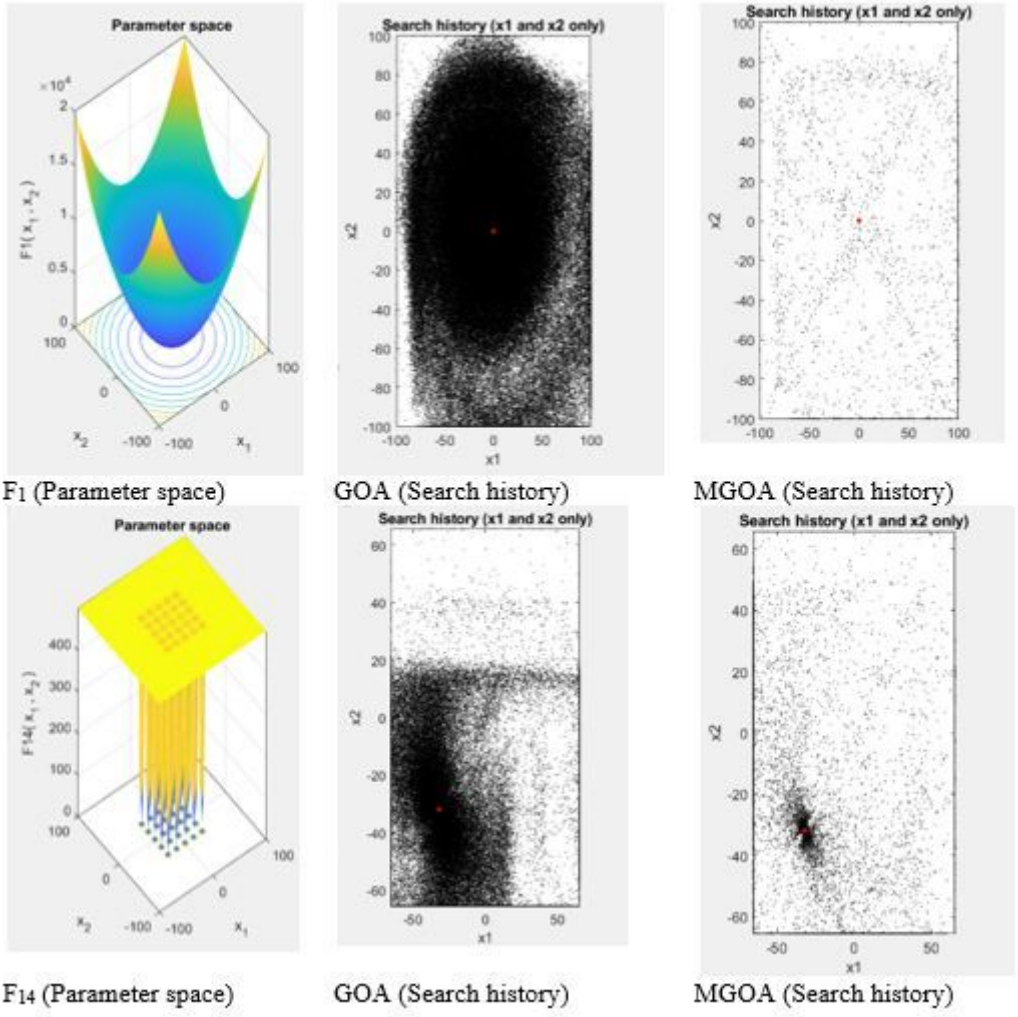


Figure 8

Parameter space and search history for F1 and F14 using GOA for MGOA

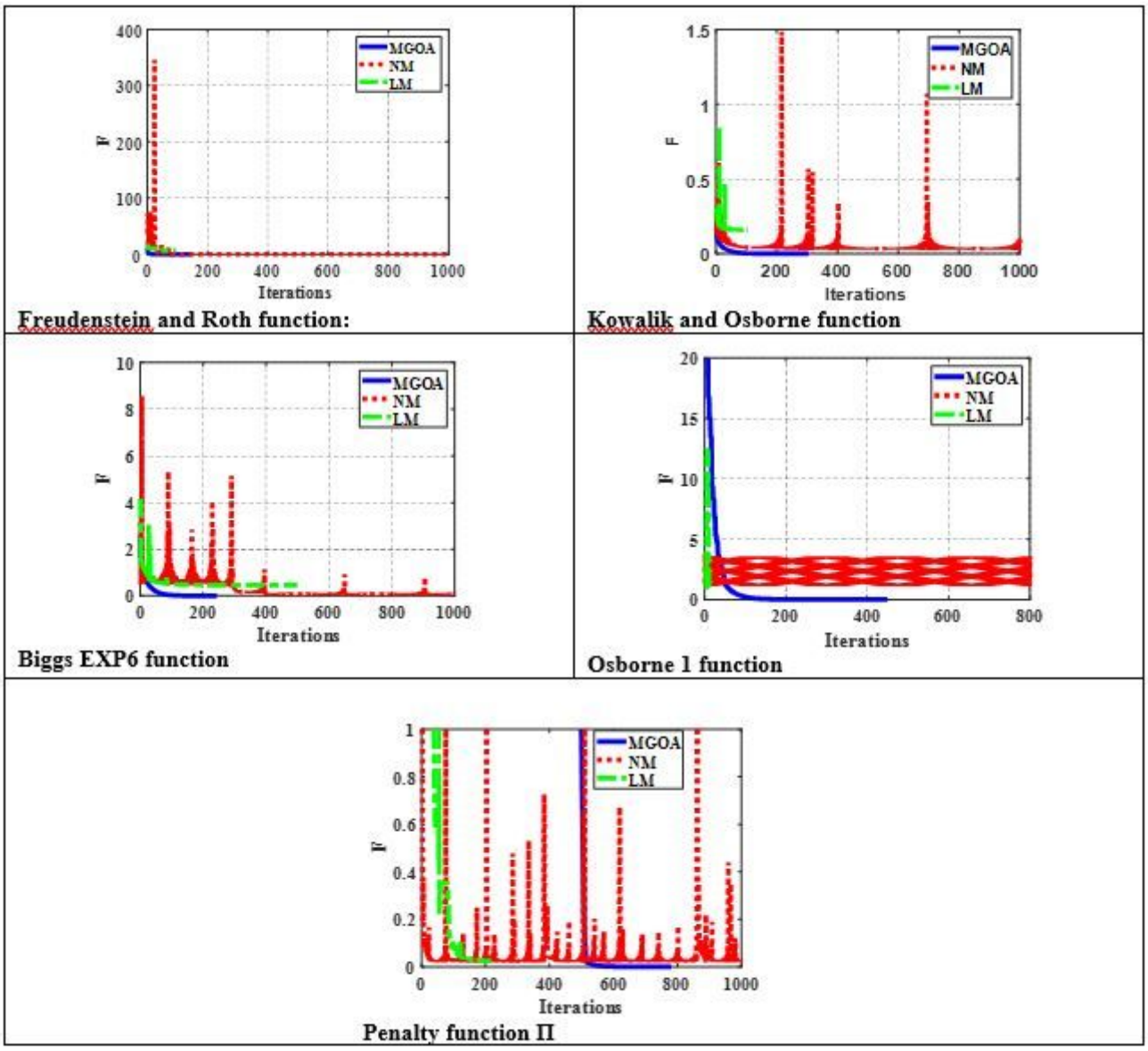


Figure 9

Convergence curves for solving nonlinear system of equations