Modified grasshopper optimization algorithm-based genetic algorithm for global optimization problems: the system of nonlinear equations case study

Hala A. Omar¹ ∙ M. A. El-Shorbagy²,¹

Accepted: 23 May 2022 / Published online: 20 June 2022 © The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
Grasshopper optimization algorithm (GOA) is one of the promising optimization algorithms for optimization problems. However, 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 were suggested to overcome this problem. This paper presents a modified grasshopper optimization algorithm (MGOA)-based genetic algorithm to overcome this problem. Modifications rely on certain mathematical assumptions and varying the domain of the control parameter, $C_{\text{max}}$, to escape from the local minimum and move the search process to an improved point. Parameter $C$ is one of the essential parameters in GOA, where it balances the exploration and exploitation of the search space. These modifications aim to speed up the convergence rate by reducing the repeated solutions and the number of iterations. Both the original GOA and the proposed algorithms are tested with 19 main test functions to investigate the influence of the proposed modifications. In addition, the algorithm will be applied to solve five different cases of nonlinear systems with different types of dimensions and regularity to show the reliability and efficiency of the proposed algorithm. Promising results are achieved compared to the original GOA. The proposed approach shows an average percentage of improvement of 96.18 as illustrated in the detailed results.

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 come from the animal 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 to solve optimization problems in the literature. Examples of MAs models are: genetic algorithm (Mousa et al. 2017; Abdelsalam and El-Shorbagy 2018; Moslehi and Haeri 2020), ant colony optimization (Moslehi and Haeri 2020), particle swarm optimization (Dorigo and Stützle 2004; El-Shorbagy and Hassanien 2018; El-Shorbagy and Mousa 2017), artificial bee colony (Mousa and El-Shorbagy 2012), cat swarm optimization (Karaboga 2005), bacterial foraging (Chu et al. 2006), firefly algorithm (Zhao and Wang 2016), glowworm swarm optimization (Elsisy et al. 2020), krill herd algorithm (Marinaki and Marinakis 2016), monkey algorithm (Bolaji et al. 2016), sine cosine algorithm (Zhou et al. 2016), cuckoo search algorithm (Abo-elnaga and El-Shorbagy 2020), grasshopper optimization algorithm (Shehab et al. 2019), salp swarm algorithm (Saremi et al. 2017), slime mould algorithm (Mirjalili et al. 2017), Harris hawks optimization (Li et al. 2020), gradient-based optimizer (Heidari et al. 2019), equilibrium optimizer
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 primarily depend on social interaction forces. It is commonly used in several optimization problems due to its fast implementation and high precision, robustness, and performance (Meraihi et al. 2021; Sezavar et al. 2019a). On the other hand, GOA has some drawbacks, including:

1. An unbalancing of exploiting and exploring processes.
2. An unstable convergence rate.
3. Falling into the local optima.

So, different GOA modifications solve these limitations in the literature (Algamal et al. 2021; Feng et al. 2020). In (Alphonsa and MohanaSundaram 2019), GOA is enhanced by using a nonlinear convergence parameter, niche mechanism, and the β-hill-climbing technique to overcome the above shortcomings. In (Dwivedi et al. 2020), the authors employed the idea of chaos to create a uniformly distributed population, increase the consistency of initial populations and the capacity to look for a new space, and leveraging the existing search space. To resolve the shortcomings of a traditional GOA and to detect the autism spectrum disorder at all times of life, the authors in Goel et al. 2020 suggested a modified GOA. In (Sezavar et al. 2019b), modified GOA is proposed as a new search method combined with a convolutional neural network (CNN) to solve modeled problems and retrieve similar images efficiently. In (Purushothaman et al. 2020), hybridized gray wolf optimization (GWO) with GOA was suggested to create more acceptable trade-off between diversification and intensification and produce substantially better performance than traditional GWO and GOA. In (Taher et al. 2019), dynamic population quantum binary GOA is proposed as an enhanced GOA for feature selection based on shared knowledge and rough set theory. In (Wang et al. 2020), improved GOA was suggested, using the nonlinear comfort zone parameter to enable the iterations of the algorithm and the Lévy flight method to maximize randomness and extend the local search range. Moreover, a random hopping technique helps the algorithm to leap from the local optimum has been implemented in Zhao et al. (2019).

This revealed that GOA has undergone several modifications to address the aforementioned flaws. Consequently, efforts 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. The original GOA has a significant parameter $C$ performing as a reducing coefficient that decreases three zones: the comfort zone, the zone of repulsion, and the attraction zone. The coefficient $C$ decreases according to the number of iterations to balance exploration and exploitation in the grasshopper algorithm. It is calculated based on the parameters $C_{\text{max}}$ (upper bound of $C$), $C_{\text{min}}$ (lower bound of $C$), and the maximum number of iterations. The proposed modifications depend on varying the upper bound $C_{\text{max}}$ through the genetic algorithm (GA) to help GOA out of the local optima when it falls into it; then, the parameter $C$ is updated according to its classical formula.

This paper aims to:

1. Introduce 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 $C_{\text{max}}$ after each GA stage.
3. Prevent the divergence of objective function value from the best-reached value by controlling the expected calculated objective function space. 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. Prevent time-consuming, calculation burden and wasting iterations by limiting the number of repeated solutions yielded from GOA by decide whether it trapped into the local minimum or not.
5. Prevent GA from working as a global search or dominating the search process by defining and controlling the search space of GA parameters and keeping track of the parameter $C_{\text{max}}$.

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

2 Preliminaries

2.1 Global optimization

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

Find \( x = (x_1, x_2, \ldots, x_n) \) such that:

\[
\begin{align*}
\text{Min } f(x), \\
\text{Subject to :} \\
L_d \leq x_d \leq U_d \ \forall \ d = 1, 2, \ldots, n;
\end{align*}
\]

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

### 2.2 The System of Nonlinear Equations

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

\[
\text{SNEs : } \begin{cases} 
    f_1(x) = 0 \\
    f_2(x) = 0 \\
    \vdots \\
    f_m(x) = 0
\end{cases}
\]

where each function \( f_i, i = 1, \ldots, m \) is a nonlinear function, which maps the vector \( x = (x_1, x_2, \ldots, x_n) \) of the \( n \)-dimensional space \( \mathbb{R}^n \) to the 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 \( f_i, i = 1, \ldots, m \) 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.

Different methods convert the SNEs to an optimization problem (Zhao et al. 2019; Nie 2004, 2006; Grosan and Abraham 2008). 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. The problem can be represented as:

\[
\begin{align*}
\min F(x) & = \min \left( \sum_{i=1}^{m} f_i^2(x) \right) = \min \|f_i(x)\|, x \\
& = (x_1, x_2, \ldots, x_n);
\end{align*}
\]

where \( F(x) \) is the objective function.

### 3 The proposed algorithm

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

#### 3.1 Genetic algorithms (GA)

The invention of genetic algorithms (GA) was dated back to the 1960s by Holland and is further described by Goldberg (More et al. 1978). The GA was successfully applied to problems in many areas, including optimization design, neural networks, fuzzy logic control, planning, expert systems, and many more (Failed 2014). 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 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 using calculations based on their fitness which is 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 Fig. 1.

#### 3.2 Grasshopper optimization algorithm (GOA)

GOA is a recent swarm algorithm that depends on simulating the swarming nature of the grasshoppers. The main feature of the grasshopper's swarm is described in two phases. The first one is the larval phase, where the steps are small and slow, while, in the second adulthood phase, the steps are long range and fast. Food source seeking is another important characteristic of the swarming of grasshoppers. 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

---

**Fig. 1** The general GA pseudo-code

---

\[\text{Generate an initial population.} \]

\[\text{Evaluate fitness of individuals in the population;} \]

\[\text{Do:} \]

\[\text{Select parents from the population;} \]

\[\text{Recombine parents to produce children;} \]

\[\text{Evaluate fitness of the children;} \]

\[\text{Replace some or all of the population by the children;} \]

\[\text{while a satisfactory solution has been found.} \]
social interaction between each one and other grasshoppers $S_i$, the gravity force $G_i$, and the wind movement $A_i$.

The resultant force that affects each grasshopper is defined as:

$$X_i = S_i + G_i + A_i$$  \hspace{1cm} (4)

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

$$S_i = \sum_{j=1}^{N} s(d_{ij})d_{ij};$$  \hspace{1cm} (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:

$$s(r) = fe^{-r} - e^{-r};$$  \hspace{1cm} (6)

where $f$ and $l$ 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. If the distance is more than 2.079, the attraction force will increase. Then, the attraction force will gradually decrease when it reaches 4. The distance between every two grasshoppers affects the calculations of social forces. When the distance between the grasshoppers is more than 10, their function ($s$) may fail to apply the forces between them. As a result, the distance of grasshoppers is mapped in the interval (Mousa et al. 2017; Dorigo and Stützle 2004). The second affected force is the gravity force which is defined as:

$$G_i = -g \hat{e}_g;$$  \hspace{1cm} (7)

where $g$ is the gravitational constant and $\hat{e}_g$ is the center of earth unity vector.

The nymph grasshoppers’ movements correlated with wind direction because they have no wings. The wind direction can be calculated as:

$$A_i = u \hat{e}_w;$$  \hspace{1cm} (8)

where $u$ is a constant drift and $\hat{e}_w$ is a wind direction unity vector.

The position of the grasshopper can be calculated as:

$$X_i = \sum_{j=1}^{N} s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} - g \hat{e}_g + u \hat{e}_w.$$

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

$$X_i = C \sum_{j=1}^{N} C \left(\frac{ub_d - lb_d}{2} s\left(\frac{x_j - x_i}{d_{ij}}\right) \frac{x_j - x_i}{d_{ij}}\right) + \hat{T}_d.$$

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

$$C = C_{\text{max}} - l C_{\text{max}} - C_{\text{min}} \frac{1}{L}$$

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

The steps of the GOA algorithm can be summarized as:

1. **Initialization** In this step, the values of the parameters are defined. These values include $C_{\text{max}}, C_{\text{min}}, F, l$, and the maximum number of iterations.
2. **Population initialized** The initial population of grasshoppers is generated randomly.
3. **Evaluation** The fitness of each candidate (grasshoppers) in this population is evaluated.
4. **Assign the best solution** After evaluating all the solutions, the best solution is defined as the target position ($\hat{T}_d$), and its corresponding fitness is assigned as the target fitness.
5. **Updating the decreasing coefficient parameter** The value of the decreasing coefficient $C$ is then calculated at each iteration depending on Eq. (11).
6. **Updating solution** Each solution in the population will be updated as in Eq. (10).
7. **Solution boundaries violation** After updating the solution, all solutions should be examined to be within the predetermined boundaries (feasible). If any solution violates its upper or lower bounds, it is returned to its domain.
8. **Repeated the main loop** Steps 3–7 are repeated until the termination condition is fulfilled.
9. **Termination criteria** The algorithm is terminated, and the best solution is displayed when either the termination condition or the maximum number of iterations is met.

### 3.3 Modified grasshopper optimization algorithm (MGOA)

Figure 2 displays the flow chart of the proposed algorithm. The figure shows that the GOA algorithm starts the optimization process by using steps in subsection 3.2. If the termination criterion is met, the algorithm is terminated, and the best solution $c_{Td}$ with its objective value is 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. 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 $\hat{T}_d$, defining a modified value of the parameter $C_{\text{max}}$. Again, if the termination criterion is reached, the algorithm terminates, and the best solution $\hat{T}_d$ and its corresponding objective value is displayed. Otherwise, the target position $\hat{T}_d$, its fitness, and $C_{\text{max}}$ 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.

#### 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, is defined based on certain assumptions and mathematical formulations. As shown in Fig. 3, suppose GOA is 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 $\hat{T}_d$, and target fitness. Assuming the next point yielded from GA is $(C_2, F_{2p})$ as shown in Fig. 3. Hence, according to the figure, the value of $C$ will be changed where $C_2$ is the new value of parameter $C$ and $F_{2p}$ is the predetermined enhanced objective value.

According to Fig. 3, the change in parameter $C$ can be calculated as:

$$\Delta C = \Delta F \tan(\theta)$$ (12)

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

$$F_{2p} = \alpha F_1, \alpha \in (0, 1)$$ (13)

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

$$\Delta F = F_{2p} - F_1 = \alpha F_1 - F_1 = (\alpha - 1)F_1$$ (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:

$$C_2 = C_1 + \Delta C$$ (15)
Fig. 4 Predicted change in $X_i$ according to changes in $C$

To ensure that the new point is always better than the previous one, the domain of the angle ($\theta$) is restricted to the interval $[\pi, 2\pi]$. According to the previous equations and assumptions, the variables in the genetic algorithm will be $[\theta, x, \beta_1, \beta_2, \ldots, \beta_n]$ where $\theta \in [\pi, 2\pi]$, $x \in [0, 1]$, and $\beta_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

As shown in Fig. 4, if the old point has a target position (solution) of $X_1 = [X_{11}, X_{21}, \ldots, X_{n1}]$ 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 the assumed angle ($\beta$). So that the new candidate solution can be calculated as:

$$\Delta X_1 = \Delta C \tan(\beta)$$

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

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

$$X_{i2} = X_{i1} + \Delta X_i = X_{i1} + \Delta C \tan(\beta_i)$$

$$\text{for } i = 1, 2, \ldots, n$$

The new solution position will be:

$$X_2 = [X_{12}, X_{22}, \ldots, X_{n2}]$$

Then, the value of actual fitness value of the new point $X_2$ is calculated as:

$$F_{2A} = F(X_2)$$

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:

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

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

For more illustration of the methodology and its steps, Fig. 5 contains the pseudo-code of the MGOA.

Figure 5 summarizes the MGOA algorithm in the following steps:

1. The standard GOA algorithm is used to begin the search process until it is unable to improve the obtained best solution in the population without meeting the termination condition, where GOA is stuck in a local minimum.
2. As detailed in Sect. 3.3.1, the GA seeks to pull GOA out of the local minimum.
3. At the end of the GA loop, $C_{\text{max}}$, the best solution and its corresponding best fitness value are updated.
4. The GOA search continues based on the updated parameter and best solution until the optimal solution is found or another local minimum is reached.
5. If GOA becomes trapped in a new local minimum, the GA loop will be repeated.
6. These steps are repeated until the termination criterion is met or the maximum number of iterations is reached.

4 Numerical results and discussions

The performance of the proposed algorithm is investigated in this section using the 19 test functions that were assessed using the original GOA algorithm (Shehab et al. 2019). These test functions are unimodal, multimodal, and composite. The mathematical formulation of these test functions is available in Shehab et al. (2019). The results will be compared with those obtained by the original GOA to show the proposed modifications’ benefits and their influence on reaching an optimal solution. Furthermore, five systems of
nonlinear equations are solved as a case study for MGOA (Broyden 1996). The proposed algorithm was coded in MATLAB. 2020a and implemented with a Core (TM) i7, Intel(R) CPU with 3.2 GHz and 16 GB RAM.

For computational studies, a population size of 20 and 50 generations is used, crossover fraction is 0.8, and all other parameters and functions were default as in GA tool. Also, the termination criterion for both algorithms (GOA and MGOA) is defined as:

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

where $\| F_{\text{optimum}} \|$ 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

---

Fig. 5 Pseudo-code of MGOA

```
Generate the initial population of grasshoppers $P_l(i = 1,2,...,n)$ randomly
Initialize $C_{\text{min}}, C_{\text{max}},$ and maximum number of iterations $t_{\text{max}}$
Evaluate the fitness $F(P_i)$ of each grasshopper $P_i$
$T_d = \text{the best solution, and } F_T = \text{the best fitness}$
while $(t < t_{\text{max}})$ or termination do
  Update $C$ using the following equation
  $$C = C_{\text{max}} - \frac{t}{C_{\text{max}}} C_{\text{min}}$$
  for $i = 1$ to $N$ (all $N$ grasshoppers in the population), do
    Normalize the distance between grasshoppers in the range $[1,4]$
    Update the position of the current grasshopper
    Bring the current grasshopper back if it goes outside the boundaries
  end for
  if there is a better solution
    Update $T_d$
    $t = t + 1$
  Else
    Apply GA to get out of local minima by:
    - $C_{\text{max}} = C_0$
    - Define the GA variables' domains:
      $\alpha \in [0,1], \theta \in [\pi, 2\pi], f_p \in \left[ \frac{\pi}{2}, 2^{\frac{\pi}{2}} \right], j = 1,2 \text{ to (m) number of variables}$
    - Generate random initial population of (GA).
    - For each individual, do
      $$\Delta F = (\alpha - 1)F_T$$
      $$\Delta C = \Delta F \tan(\theta)$$
      $$X = T_d^{\circ} + \Delta C \left[ \tan(\theta_1) \tan(\theta_2) \tan(\theta_3) \ldots \tan(\theta_m) \right]$$
      $$F = F(X)$$
      $$F_{\text{presumed}} = \alpha F_T$$
      - Apply GA operations to get:
        $$\min_{\theta, \alpha, \beta, \gamma} \| F - F_{\text{presumed}} \|$$
      - Use the optimum parameters: $\alpha_{\text{opt}}, \theta_{\text{opt}}, \beta_{\text{opt}}, \gamma_{\text{opt}}$ to calculate:
        $$X_{2A} = T_d^{\circ} + \Delta C \left[ \tan(\theta_{1\text{opt}}) \tan(\theta_{2\text{opt}}) \tan(\theta_{3\text{opt}}) \ldots \tan(\theta_{m\text{opt}}) \right]$$
        $$F_{2A} = \alpha_{\text{opt}} F_T$$
    - Update $C_{\text{max}} = C_0 + \Delta C$
    - Update the best solution ($T_d = X_{2A}$), $F_T = F_{2A}$
  if termination
    end while
  Return the best solution $T_d$
Else
  Return to the GOA (step 5) with the updated $C_{\text{max}},$ the best solution, and best fitness. continue GOA search.
end while
Return the best solution $T_d$
```
has met the termination criterion, the calculations terminates, and the number of consumed iterations is displayed.

Furthermore, the mechanism of MGOA to decide whether GOA is trapped into the local minimum or not is to count a limited number of iterations that 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 \( \alpha \) should not be precisely 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 \( \alpha \) is defined as \([0,0.9]\). Finally, it should be mentioned that the parameter \( C_{\text{max}} \) equals to 1, and the parameter \( C_{\text{min}} \) equals to 1e-05.

### 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 Fig. 6 shows the convergence curves of the function value. Besides, in test function \( F_3 \), the resulting objective values were too large so, they were rationalized by the maximum reached value \( (F_{\text{max}}) \) to show the results of both algorithms more clearly.

As shown in Table 1, MGOA found the optimal solutions after fewer iterations than GOA. For example, after 4960 iterations, the GOA found the solution in \( F_1 \). While MGOA found it after 131 iterations, 37 of them were used by the GA stage to avoid trapping into a local minimum. In addition, the GOA could not find a solution until it reached the maximum number of iterations as in cases of test functions \( (F_2 \) and \( F_7)) \). The results illustrate how the MGOA converged faster and was more reliable than the original GOA. Table 1’s last column indicates the suggested algorithm’s improvement as a percentage relation, which is determined as follows:

\[
\text{Percentage improvement} = \frac{\text{GOA iterations} - \text{MGOA iterations}}{\text{GOA iterations}} \times 100 \tag{23}
\]

As shown in Fig. 6, the straight segments in GOA curves represent the periods of non-improving of the objective function due to trapping in a local minimum. In some cases, the GOA succeeded in getting out of the local minimum. During the computations, however, it falls into other local minimums until it reaches the optimal solution as illustrated in \( (F_1, F_3, F_4, F_5, F_6, \) and \( F_8) \). In addition, GOA was unable to solve the two cases \( (F_2 \) and \( F_7) \) until the maximum number of iterations had been used. On the other hand, MGOA improved all results significantly as indicated in Table 1. This revealed that MGOA guides GOA to eliminate the local minimum and enhance the search results, reducing the number of iterations and, as a result, time, by preventing iterations from being used without improvement and convergence to the solution.

For multimodal test functions, Table 2 shows the results of these functions, presenting the results of both GOA and

| Unimodal functions | GOA | MGOA | Percentage improvement |
|-------------------|-----|------|------------------------|
| No Optimum value | Found the solution | \( |F| \) | Number of iterations | Found the solution | \( |F| \) | The whole number of iterations | GOA iterations | GA iterations |
| \( F_1 \) 0 | Yes | \( 1.2 \times 10^{-6} \) | 4960 | Yes | \( 9.9 \times 10^{-7} \) | 131 | 94 | 37 | 97.35% |
| \( F_2 \) 0 | No | 2.337 | 7000 | Yes | \( 6.7 \times 10^{-7} \) | 290 | 195 | 95 | 95.86% |
| \( F_3 \) 0 | Yes | \( 8.6 \times 10^{-7} \) | 6920 | Yes | \( 7.8 \times 10^{-7} \) | 394 | 263 | 131 | 94.31% |
| \( F_4 \) 0 | Yes | \( 8.9 \times 10^{-7} \) | 6999 | Yes | \( 4.7 \times 10^{-7} \) | 262 | 178 | 84 | 96.26% |
| \( F_5 \) 0 | Yes | 0.201 | 7000 | Yes | \( 1.6 \times 10^{-7} \) | 346 | 231 | 115 | 95.06% |
| \( F_6 \) 0 | Yes | \( 8.1 \times 10^{-7} \) | 6642 | Yes | \( 7.6 \times 10^{-7} \) | 263 | 176 | 87 | 96.04% |
| \( F_7 \) 0 | No | 0.012 | 7000 | Yes | \( 4.2 \times 10^{-7} \) | 80 | 54 | 26 | 98.85% |
| \( F_8 \) 0 | Yes | \( 4.9 \times 10^{-7} \) | 6998 | Yes | \( 1.1 \times 10^{-7} \) | 39 | 27 | 12 | 99.44% |
Modified grasshopper optimization algorithm-based genetic algorithm for global optimization...

\[ F_1 \]  
\[ F_2 \]  
\[ F_3 \]  
\[ F_4 \]  
\[ F_5 \]  
\[ F_6 \]  
\[ F_7 \]  
\[ F_8 \]
MGOA. Table 2 illustrates that MGOA reached the optimum solution for all cases in fewer iterations. However, GOA solved only three test functions with consuming a high number of iterations. The last column represents the percentage improvements in these test cases according to Eq. (23).

Figure 7 shows the convergence curves of the function value of these test functions. The figures are drawn as \( r \) versus \( F \), where \( r \) is the current iteration’s ratio to the algorithm’s maximum number of iterations. \( F \) is the value of the objective function to show the convergence curves of the two algorithms more clearly. It is worth mentioning that \( r \) was calculated after the termination of each algorithm and according to its results. The ratio \( r \) is represented as:

\[
    r = \frac{\text{iteration}}{\text{The whole number of iterations consumed by both algorithm}}
\]

As shown in Table 2 and Fig. 7, MGOA reached the solution in fewer iterations than that of the 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 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 3 presents the results of the composite test functions, along with the percentage of results improvements. As shown in Table 3, MGOA found the optimum solution for all cases in fewer iterations, whereas GOA only resolved two test functions after consuming a large number of iterations.

Table 2 Results of multimodal test functions

| Multimodal functions | GOA | MGOA |
|----------------------|-----|------|
| No Optimum value     | Found the solution | || Found the solutions | Number of iterations | Number of iterations | GOA iterations | GA iterations | Percentage improvement |
| F9 0                 | No   | 3.07 | 7000 | Yes | 9.5 × 10^{-7} | 416 | 278 | 138 | 94.06% |
| F10 0                | Yes  | 7.4 × 10^{-6} | 7000 | Yes | 6.3 × 10^{-7} | 278 | 186 | 92 | 96.03% |
| F11 0                | No   | 0.07 | 7000 | Yes | 9.02 × 10^{-7} | 416 | 278 | 138 | 94.06% |
| F12 0                | Yes  | 8.9 × 10^{-7} | 6458 | Yes | 2.8 × 10^{-7} | 218 | 146 | 72 | 96.62% |
| F13 0                | Yes  | 9.4 × 10^{-7} | 6865 | Yes | 7.7 × 10^{-7} | 323 | 216 | 107 | 95.29% |

Figure 8 shows show the curves of the objective value of absolute error function \( F \) versus the parameter \( C \) in both algorithms. The aim of plotting this relationship is to show how changing the parameter \( C_{\text{max}} \) can improve MGOA’s performance in terms of convergence to the best solution. In addition, it shows that the classical decreasing \( C_{\text{max}} \) leads to trapping into the local minimum. For example, in the test function \( F_{17} \), in GOA, the value of parameter \( C_{\text{max}} \) starts with 1, decreasing according to Eq. (11) during the optimization process. However, MGOA starts with 1 until the GOA falls into a local minimum. Its domain was then edited using the GA stage whereby the parameter \( C_{\text{max}} \) increased to 1.05 then and then started to decrease again from this updated value according to Eq. (11). In test function \( F_{18} \), the value \( C_{\text{max}} \) continuously reduced until it reached almost 0.85; then, GOA was trapped into a local minimum, and the GA stage was needed. By applying the GA stage, the value of \( C_{\text{max}} \) was updated to nearly 0.95 and introduced with other parameters to GOA again. Furthermore, in the test functions \( F_{15} \) and \( F_{16} \), the value of \( C_{\text{max}} \) was updated by decreasing but in a different slope than resulted by Eq. (11). It is worthy here to state that the main target of the GA stage is to update the value of \( C_{\text{max}} \) by increasing or decreasing it to escape from the local optima. Finally, these results show that how the proposed MGOA led to continuous decreasing of the objective function value until the desired tolerance and optimum solution were achieved.

4.2 Analysis of results and discussion

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

1. MGOA does not use both GOA and GA to solve the optimization problem and to find an optimal solution.
However, the main algorithm is GOA, and when it falls into a local minimum, the GA stage intervenes to get GOA 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 ($C_{\text{max}}$, $\hat{T}_d$, 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 $C_{\text{max}}$ helps 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 applied when GOA is fallen into a local minimum. So, in MGOA, GA acts as a local search algorithm, not a global one.

5. The proposed hybridization or modification achieves the aimed enhancement in the GOA performance, where the average percentage of improvement in MGOA for all 19 test cases illustrated above is about 96.18% which is calculated as:

\[
\text{Average improvement percentage} = \frac{\sum_{i=1}^{19} \text{Percentage improvement}}{19} \tag{25}
\]

6. For further illustration, Fig. 9 shows the parameter space and search history for the two test functions $F_{1}$ and $F_{14}$, using GOA and MGOA.

a) It is evident from Fig. 9 that the GOA distribution of grasshoppers around the solution was so condensed because of remaining in local minimum for extensive periods or a large number of iterations. After 4960 iterations, GOA in $F_{1}$ was, however, able to identify a solution. While in $F_{14}$, GOA 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 rapidly.

4.3 Case study: solving nonlinear systems of equations

In this subsection, MGOA will be applied to various types of nonlinear systems of equations. All details of these nonlinear systems can be found in More et al. (1978). First, the system of nonlinear equations is converted to an optimization problem according to Eq. (2). Then, this optimization problem is solved by the proposed algorithm MGOA. The results will be compared to Newton’s method (NM) (Nie 2004, 2006; Grosan and Abraham 2008) and Levenberg–Marquardt’s algorithm (LMA) (Failed 2014; Broyden 1996; Jarry and Beneat 2016). 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 $n = m = 2$.
   \[
   f_1(x) = -13 + x_1 + ((5 - x_2)x_2 - 2)x_2 \\
   f_2(x) = -29 + x_1 + ((x_2 + 1)x_2 - 14)x_2 \tag{26}
   \]

2. **Kowalik and Osborne function:** This system is non-square with $n = 4, m = 11$.
   \[
   f_i(x) = y_i - \frac{\overline{u_i^2} + u_i x_2}{u_i^2 + u_i x_3 + x_4}, \quad i = 1, 2, \ldots, m; \tag{27}
   \]
   where $\overline{y}_i$ and $u_i$'s are constants and their definition can be found in Broyden (1996).

3. **Biggs EXP6 function:** This system is square with $n = 6, m = n$.
where $t_i = 0.1i$.

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

$$f_i(x) = x_3 e^{-t_{i1}} - x_4 e^{-t_{i2}} + x_6 e^{-t_{i8}} - y_i,$$

where $t_i = 0.1i$.

$$y_i = e^{-h} - 5e^{-10h} + 3e^{hi}, i = 1, 2, \ldots, m;$$

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

$$f_i(x) = y_i - (x_1 + x_2 e^{-t_{i1}} + x_3 e^{-t_{i1}}), i = 1, 2, \ldots, m,$$

where $t_i = 10(i - 1)$;

$$y_i/10$$ are constants and its values can be found in More et al. (1978)
\[ f_1(x) = x_1 - 0.2, \]
\[ f_i(x) = a^\frac{1}{2} \left( e^{x_{i-1}/10} + e^{x_i/10} \right) - y_i, \quad \forall 2 \leq i \leq n, \]
\[ f_i(x) = a^\frac{1}{2} \left( e^{x_{i-1}/10} - e^{i/10} \right), \quad \forall n < i < 2n, \]
\[ f_{2n} = \left( \sum_{j=1}^{n} (n-j+1)x_j^2 \right) - 1, \]
\[ y_i = e^{i/10} + e^{(i-1)/10}, \quad a = 10^{-5}. \]

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. Figure 10 shows the convergence curves of the three algorithms. Table 4 clarifies how the MGOA algorithm solved all nonlinear systems in fewer iterations. In addition, despite the high ability of the LM algorithm to deal with nonlinear systems, it failed to detect the solution in cases of singularity. However, MGOA overcame this issue as it does not require derivatives calculations.

Table 4 and Fig. 10 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. However, both NM and LM failed to obtain the
solution in addition to the calculation burden due to calculations of the Jacobian matrices. The results show that MGOA successfully handles nonlinear systems that are even singular square or non-square.

### 5 Conclusions

In this paper, a modified grasshopper optimization algorithm (MGOA) is proposed by modifying the classic grasshopper optimization algorithm (GOA) and the genetic algorithm (GA). The modification aims to save the GOA from falling into local minimum and to improve the convergence rate. The MGOA achieves that by reducing the number of repeated solutions and decreasing the number of spent iterations. These modifications are based on some mathematical assumptions and relations that depend on releasing the calculations of the parameter $C$ to move to a new and improved point in the search space. MGOA was tested on the same 19 test functions that the original GOA solved 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. The test functions’ results illustrate via tables and figures how the MGOA succeeded in reaching the optimum solution in fewer iterations than the GOA. Besides, the convergence curves show how the GOA consumes time in trapping into the local minima. Hence, the MGOA terminates this sticking in the local minimum and moves the GOA to a new better search space by using the GA. The proposed hybridization achieves the aimed enhancement in the GOA performance, where the average percentage of improvement in MGOA was about 96.18. Also, MGOA was used to solve five 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 remarkable success in solving the various square, non-square, and singular nonlinear systems cases, 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. Large-scale engineering challenges, such as resource allocation difficulties, economical load transfer problems, unit commitment concerns, wind farm optimization of wind turbines, and real-time applications, may all be addressed using the suggested approach. Finally, it can be extended to solve multiobjective optimization problems.

| 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 | $9.4 \times 10^{-9}$ | 427 |
| NM | No | 0.025 | 1000 |
| LM | No | 0.074 | 1000 |
| Biggs EXP6 function | MGOA | Yes | $7.4 \times 10^{-9}$ | 161 |
| NM | No | 0.04 | 1000 |
| LM | No | 0.1 | 1000 |
| Osborne 1 function | MGOA | Yes | $9.4 \times 10^{-9}$ | 544 |
| NM | No | 0.54 | 1000 |
| LM | No | 1.53 | 1000 |
| Penalty function II | MGOA | Yes | $9.1 \times 10^{-9}$ | 596 |
| NM | No | 0.02 | 1000 |
| LM | No | 0.017 | 1000 |
Author contributions MAE contributed to conceptualization; MAE and HAO contributed to investigation, writing—review and editing, have read, and agreed to the published version of the manuscript; and HAO contributed to methodology and writing—original draft.

Funding The authors received no specific funding for this work.

Data availability All data used to support the findings of this study are included in the article.

Declarations

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

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

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

References

Abdelsalam AM, El-Shorbagy MA (2018) Optimization of wind turbines siting in a wind farm using genetic algorithm based local search. Renew Energy 123:748–755
Abo-elnaga Y, El-Shorbagy MA (2020) Multi-sine cosine algorithm for solving nonlinear bilevel programming problems. Int J Comput Intell Syst 13(1):421–432
Ahmadianfar I, Bozorg-Haddad O, Chu X (2020) Gradient-based optimizer: a new Metaheuristic optimization algorithm. Inf Sci 540:131–159
Algamal ZY, Qasim MK, Lee MH, Ali HTM (2021) Improving grasshopper optimization algorithm for hyperparameters estimation and feature selection in support vector regression. Chemom Intell Lab Syst 208:104–125
Alphonsa MMA, MohanaSundaram N (2019) A reformed grasshopper optimization with genetic principle for securing medical data. J Inf Secur Appl 47:410–420
Bolaji AL, Al-Betar MA, Awadallah MA, Khader AT, Abualigah LM (2016) A comprehensive review: Krill Herd algorithm (KH) and its applications. Appl Soft Comput 49:437–446
Broyden CG (1996) A class of methods for solving nonlinear simultaneous equations. Math Comp 19(92):577–593
Chu S-C, Tsai P-W, Pan J-S (2006) Cat swarm optimization. Pacific rim international conference on artificial intelligence. Springer, Berlin, Heidelberg, pp 854–858
Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge
Dwivedi S, Vardhan M, Tripathi S (2020) An effect of chaos grasshopper optimization algorithm for protection of network infrastructure. Comput Netw 176:107251
El-Shorbagy MA, Hassanien AE (2018) Particle swarm optimization from theory to applications. Int J Rough Sets Data Anal (IJRSDA) 5(2):1–24
El-Shorbagy MA, Mousa AA (2017) Chaotic particle swarm optimization for imprecise combined economic and emission dispatch problem. Rev Inf Eng Appl 4(1):20–35
El-Shorbagy MA, Mousa AA (2021) Constrained multiobjective equilibrium optimizer algorithm for solving combined economic emission dispatch problem. Complexity 2021:1–14
Elisy MA, Hammad DA, El-Shorbagy MA (2020) Solving interval quadratic programming problems by using the numerical method and swarm algorithms. Complexity 2020:1–11
Éric Walter (2014) Numerical methods and optimization. Springer International Publishing Switzerland
Feng H, Ni H, Zhao R, Zhu X (2020) An enhanced grasshopper optimization algorithm to the bin packing problem. J Control Sci Eng 2020:1–16
Goel N, Grover B, Gupta D, Khanna A, Sharma M (2020) Modified grasshopper optimization algorithm for detection of autism spectrum disorder. Phys Commun 41:101115
Grosan C, Abraham A (2008) A new approach for solving nonlinear equations systems. IEEE Trans Syst Man Cybern Part A Syst Humans 38(3):698–714
Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafjarah M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872
Jarry P, Beneat JN (2016) Appendix B levenberg–marquardt–more. In: Optimization algorithms. pp 239–244
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech Rep TR06 Erciyes Univ Eng Fac Comput Eng Dep 200:1–10
Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: a new method for stochastic optimization. Futur Gener Comput Syst 111:300–323
Marinaki M, Marinakis Y (2016) A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. Expert Syst Appl 46:145–163
Meraityi R, Gabis AB, Mirjalili S, Ramdane-Cherif A (2021) Grasshopper optimization algorithm: theory, variants, and applications. IEEE Access 9:50001–50024
Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
More JJ, Garbow BS, Hillstrom KE (1978) Testing unconstrained optimization software, Argonne national laboratory argonne, Illinois 60439. In: Applied Mathematics Division Technical Memorandum, 324
Moslehi F, Haeri A (2020) A genetic algorithm-based framework for mining quantitative association rules without specifying minimum support and minimum confidence. Sci Iran 27(3):1316–1332
Mousa AA, El-Shorbagy MA (2012) Enhanced particle swarm optimization based local search for reactive power compensation problem. Appl Math 3:1276–1284
Mousa AA, El-Shorbagy MA, Farag MA (2017) K-means-clustering based evolutionary algorithm for multi-objective resource allocation problems. Appl Math Inf Sci 11(6):1–12
Nie PY (2004) A null space method for solving system of equations. Appl Math Comput 149(1):215–226
Nie PY (2006) An SQP approach with line search for a system of nonlinear equations. Math Comput Model 43(3–4):368–373
Purushothaman R, Rajagopaln SP, Dhandapani G (2020) Hybridizing gray wolf optimization (gwo) with grasshopper optimization algorithm (gga) for text feature selection and clustering. Appl Soft Comput 96:106651
Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
Sezavar A, Farsi H, Mohamadzadeh S (2019a) A modified grasshopper optimization algorithm combined with cnn for content based image retrieval. Int J Eng 32(7):924–930
Sezavar A, Farsi H, Mohamadzadeh SA (2019b) Modified grasshopper optimization algorithm combined with CNN for content based image retrieval. Int J Eng 32(7):924–930
Shehab M, Khader AT, Laouchedi M, Alomari OA (2019) Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization. J Supercomput 75(3):2395–2422
Taher MA, Kamel S, Jurado F, Ebeed M (2019) Modified grasshopper optimization framework for optimal power flow solution. Electr Eng 101(1):121–148
Wang D, Chen H, Li T, Wan J, Huang Y (2020) A novel quantum grasshopper optimization algorithm for feature selection. Int J Approx Reason 127:33–53
Zhao W, Wang L (2016) An effective bacterial foraging optimizer for global optimization. Inf Sci 329:719–735
Zhao W, Ni H, Feng H, Song Y, Zhu X (2019) An improved artificial bee colony algorithm for the vehicle routing problem with time windows. Appl Soft Comput 75:829–842
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.