Neural Computing and Applications (2022) 34:7711–7731
https://doi.org/10.1007/s00521-021-06885-9

ORIGINAL ARTICLE

Adaptive grey wolf optimizer

Kazem Meidani[1] · AmirPouya Hemmasian[1] · Seyedali Mirjalili[2,3] · Amir Barati Farimani[1,4,5]

Received: 17 August 2021 / Accepted: 19 December 2021 / Published online: 11 January 2022

Abstract
Swarm-based metaheuristic optimization algorithms have demonstrated outstanding performance on a wide range of optimization problems in both science and industry. Despite their merits, a major limitation of such techniques originates from non-automated parameter tuning and lack of systematic stopping criteria that typically leads to inefficient use of computational resources. In this work, we propose an improved version of grey wolf optimizer (GWO) named adaptive GWO which addresses these issues by adaptive tuning of the exploration/exploitation parameters based on the fitness history of the candidate solutions during the optimization. By controlling the stopping criteria based on the significance of fitness improvement in the optimization, AGWO can automatically converge to a sufficiently good optimum in the shortest time. Moreover, we propose an extended adaptive GWO ($AGWO^4$) that adjusts the convergence parameters based on a three-point fitness history. In a thorough comparative study, we show that AGWO is a more efficient optimization algorithm than GWO by decreasing the number of iterations required for reaching statistically the same solutions as GWO and outperforming a number of existing GWO variants.

Keywords Metaheuristic optimization · Adaptive optimization · Grey wolf optimizer · Fitness-based adaptive algorithm

1 Introduction

Optimization problems in various fields of science and engineering always require better and more novel optimization algorithms that can improve our ability to find the optimal value and corresponding decision variables of some objective function. Metaheuristic algorithms have attracted ever-increasing interest in the previous decade due to their superiority and efficiency in solving difficult optimization problems [1, 2]. Such methods are known to be stochastic, simple, flexible, derivation-free, and global search algorithms. Their simplicity and flexibility have paved the way to be applied in a vast variety of problems in both science and industry.

Unlike gradient-based methods which require an explicit form of the objective function and its derivative, these methods can operate on black-box functions or functions that have unknown or expensive derivatives. At last, by employing multiple agents as the population, they are famous to avoid local optima and converge to the global optimum. Metaheuristic algorithms are generally classified into nature-inspired and non-nature-based methods. Many nature-inspired algorithms have been introduced that mathematically model some phenomena in nature to use it

✉ Amir Barati Farimani
barati@cmu.edu

Kazem Meidani
mmeidani@andrew.cmu.edu

AmirPouya Hemmasian
ahemmasi@andrew.cmu.edu

Seyedali Mirjalili
ali.mirjalili@gmail.com

[1] Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

[2] Centre for Artificial Intelligence Research and Optimization, Torrens University Australia, Adelaide, Australia

[3] Yonsei Frontier Lab, Yonsei University, Seoul, Republic of Korea

[4] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA

[5] Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

for optimization problems. Genetic Algorithm (GA) [3], Particle Swarm Optimization (PSO) [4], Differential Evolution [5], and some other algorithms are well regarded among scientists in various areas. There have been a lot of new variants proposed for each of these algorithms to improve their performance when showing poor results on specific types of problems.

Based on No Free Lunch (NFL) theorem [6], there is no single metaheuristic method that can be proposed to outperform all other algorithms for solving every optimization problem in the world. Therefore, many researchers are actively working to create and improve metaheuristic algorithms that can solve the problems more efficiently. It is worth mentioning that these algorithms do not guarantee to reach the global minimum and they are all based on systematic trial and error. Most of these methods employ a population of agents rather than a single agent to parallelize the search in the space and increase the ability of the model to avoid local minima.

Metaheuristic algorithms include a wide variety of methods proposed in different contexts, with different origins and applications. The origins of a lot of such methods are inspirations coming from mother nature. Aside from algorithms mimicking human intelligence, nature-inspired algorithms have been roughly categorized based on their origin of inspiration into three classes of Evolutionary algorithms (EA), Physics-inspired algorithms, and Swarm Intelligence (SI). Algorithms like Tabu Search (TS) [7] and Iterated Local Search (ILS) [8] are based on strategies to avoid previously visited bad locations during the search for the optimum or to improve hill-climbing algorithm to avoid getting stuck at local minima. Evolutionary algorithms get inspiration from the natural competition and selection in the process of natural evolution. Some of the well-known algorithms in this group are Differential Evolution (DE) [5], Genetic Algorithm (GA) [3], Evolution Strategies (ES) [9], and biogeography-based optimizer (BBO) [10]. Physics-inspired algorithms are inspired by the laws of physics such as gravity in Gravitational Search Algorithm (GSA) [11] or big bang-big crunch (BB-BC) [12].

The last group which we focus on is Swarm Intelligence (SI). SI algorithms are inspired by swarm movements in various animals in nature such as ant colony optimization (ACO) [13], Artificial Bee Colony [14], Particle Swarm Optimization (PSO) [15], Cuckoo Search (CS) [16], and Bacterial Foraging Optimization (BFO) [17]. For example, in ACO, the behavior of ants to find the shortest available path between their colony and the food source is investigated. Many other algorithms have been recently proposed that model and imitate the group behavior of various swarms and groups of animals in nature. Grey Wolf Optimizer (GWO) [18], Whale Optimization Algorithm (WOA) [19], Moth-flame Optimization (MFO) [20], Dragonfly Algorithm (DA) [21], Grasshopper Optimization Algorithm (GOA) [22], and the Ant Lion Optimizer (ALO) [23] are some well-known examples of SI algorithms.

Grey Wolf Optimizer or GWO [18] is one of the recent swarm-based metaheuristic algorithms that has attracted a lot of attention and been used in many applications. As the name suggests, this algorithm is inspired by the hunting behavior of grey wolf packs. Having three leaders that navigate the group toward the optimal values has made this method a successful algorithm in avoiding local minima with fast convergence. However, there have been some drawbacks associated with this algorithm such as lack of diversity, exploration/exploitation imbalance, and premature local minima convergence in large-scale problems [24–26]. There has been an active line of research on how to improve the original algorithm that brought about several variants of GWO [27, 28]. Modifications to the update steps of GWO, and specifically the convergence parameters have tried to improve the ability of GWO in dealing with real-world applications. We discuss these variants in Sect. 2.

The contribution of this study is to overcome the aforementioned weaknesses in two ways. First, by defining a fitness-based stopping criterion that observes the significance of improvements during the optimization process and makes the optimization efficient by stopping when the improvements become negligible from the function value viewpoint. Second, by introducing an exploration/exploitation parameter that is independent of the iteration number and is automatically adjusted with the behavior of the optimization in the recent iterations. We combine these two contributions in a novel fitness-dependent adaptive version of GWO named Adaptive GWO (AGWO). In addition, a three-point fitness-based extension (AGWO^4) is proposed that improves AGWO's performance, specifically on unimodal functions.

The structure of this paper is as follows: Some related works and variants of GWO are mentioned in Sect. 2, and their novelties and limitations are discussed. In Sect. 3, the original methodology of the grey wolf optimizer (GWO) is reviewed and different steps of the algorithm are elaborated upon and their importance is discussed. Section 4 explains the proposed algorithm and its implementation details. The reasonings are provided to support the algorithm and its extended version (AGWO^4). The performance of AGWO is evaluated using several standard experiments on benchmark test functions in Sect. 5. The convergence curves, as well as performance tables are provided to compare the proposed algorithm with the original GWO algorithm and some of its recent variants. At last, the

results are discussed and some future directions are suggested in Sect. 6.

2 Related works

Recently, there have been many studies to enhance the GWO algorithm in different ways. Some variants proposed some adjustment strategies for the GWO parameters, i.e., A and C. Some other works integrate novel or currently existing operators such as local search methods to improve GWO's performance. Finally, combining GWO with other existing metaheuristic algorithms, i.e. hybrid algorithms, is another way to improve GWO's characteristics such as its exploration/exploitation balance. Another aspect of related studies to this work is adaptive tuning of the parameters. Adaptive methods have been previously used in combination with other metaheuristic algorithms and have shown promising enhancements to them. Although our proposed fitness-dependent adaptive algorithm is essentially different from the previous methods, we discuss several adaptive algorithms along with their mechanisms and features.

Modifications to the update scheme and parameter adjustments can improve GWO's performance on some types of problems. The weighted distance average of the three best solutions as opposed to the simple average of them is proposed as wdGWO [29]. Mittal et al. [30] proposed mGWO in which the exploration is enhanced by modifying the parameters of convergence to nonlinear form. Modified augmented Lagrangian with improved grey wolf optimizer (MAL-IGWO) [31] also employs a nonlinear adjustment for the exploration/exploitation parameter a of the GWO and shows a better balance in constrained optimization problems. Another augmented GWO algorithm also modifies the parameter a to increase the possibility of exploration in comparison with exploitation and is best suited for small population search. These nonlinear modifications, however, are successful in improving algorithm's performance on some particular sets of problems. For example, mGWO's exploration favorable change enhances convergence performance on unimodal functions but is less effective for more complex multimodal functions. Besides parameter update equations, Fuzzy logic is also used for the dynamic adaptation of GWO parameters and the update rule for the position of agents [32, 33]. Relative function values of agents in the population are used in an adaptive way to modify the parameter a for obtaining better solutions in partial discharge optimization problem [34]. However, it uses fitness values in each single iteration, and similar to other variants, it is strongly dependent on the iteration number. EE-GWO [35] is an exploration-enhanced GWO algorithm that applies modifications on both the update step and the nonlinear parameter setting of exploration/exploitation parameter a. EE-GWO is shown to suit best for high-dimensional complex problems rather than more simple unimodal functions. Random Opposition Based Learning GWO (ROL-GWO) [36] modifies the parameter C instead of A to improve the algorithm by increasing exploration. In Enhanced GWO (EGWO), the diversity is increased with opposition-based learning, and also the parameter a is adjusted to oscillate for the first part of the optimization and attain a constant value in the rest of the iterations [37]. Tuning the parameter adjustment form and their additional variables is another downside of these methods. Tuning is usually done by considering a set of functions as training functions to adjust these parameters. This will bias the algorithms toward performing well for specific sets of functions.

Some other variants hybridize GWO with some other local search or metaheuristic algorithms to improve its performance. A new search strategy named Dimension Learning-based Hunting (DLH) is introduced in I-GWO [38] to enhance the global and local search in GWO. Although not evaluated on a lot of benchmark functions, the integration of local search algorithms like Powell local search optimization and Pattern Search algorithms have shown promising results in different applications [39, 40]. Lévy flight is integrated with GWO (LGWO) [25] to avoid local stagnation and improve exploration. With the same goal, Cellular GWO (CGWO) with topological structure is introduced that considers topological neighbors for each wolf that also enhances subgroup exploitation [26]. Evolutionary Population Dynamics (EPD) operator is combined with GWO (EPD-GWO) [41] to relocate bad agents to better locations in the search space. However, the balance of the exploitation with exploration, specifically in unimodal and hybrid functions, may need further improvements. Hybrid GWO (HGWO) employs crossover and mutation operators along with GWO to solve economic dispatch problem [42]. GWO is equipped (E-GWO) with tournament selection, crossover, and mutation in companion with a sinusoidal bridging mechanism to improve its local minima avoidance in multimodal functions. Again, the survival of fittest (SOF) idea in biological evolution is added to the vanilla GWO to form another improved version of GWO (IGWO) [43]. The differential evolution and GWO hybrid method is combined with a change of GWO basic update algorithm, i.e., hierarchy structure, to enhance its local optima avoidance ability in hierarchy strengthened GWO (HSGWO) [24]. Quasi-Oppositional Based Learning (Q-OBL) theory has also been incorporated with GWO to form QO-GWO algorithm [44]. In another similar work, the information sharing strategy of the Artificial Bee Colony (ABC) algorithm is integrated with the hierarchical leadership of GWO to boost the exploration of GWO in

complex problems (ABC-GWO) [45]. A hybrid of harmony search with GWO (GWO-HS) is used to solve the parameter selection problem in harmony search [46]. A categorization of nature-inspired metaheuristic algorithms along with subcategories of swarm intelligence and different variants of GWO is depicted in Fig. 1.

However, there are some limitations with the implementation and applications of these algorithms. Usually, the performance of these algorithms is evaluated not only by their ability to converge to a good minimum but also by their convergence speed and computational efficiency. However, the maximum number of iterations should be empirically provided to the algorithm which is not an obvious choice for many optimization problems. In fact, an unknown objective function has an unknown complexity, while the required time strongly depends on the objective function. Two examples of objective functions with different degrees of complexity are depicted in Fig. 2. We expect the first function (top in Fig. 2) to need much less time for a good convergence in comparison with the second function (bottom in Fig. 2). The parameter selection and adjustment, however, are prior to interaction with the objective functions. Selecting too short or too long iterations results in premature convergence and the waste of computational time, respectively. Various types of stopping criteria have been used to realize the convergence of the optimization, and also several adjustments as mentioned above can be made to improve the balance of the exploration and exploitation during the optimization. However, none of these algorithms can automatically adjust the convergence parameters based on the objective function and still need prior input of maximum iteration. In fact, all of the previously introduced variants of GWO are dependent on the iteration number.

Adaptive methods have been introduced for some of the metaheuristic algorithms to set the involved parameters based on the behavior of the agents in either the fitness values or their positions. Various parameters used in evolutionary algorithms can be controlled using adaptive methods [47]. Adaptive Particle Swarm Optimization (APSO) [48] introduces an average distancing method that can tune the exploration/exploitation balance and the weights based on relative positions of particles. In APSO, the relative positions of agents and their formation with respect to the best points are exploited as useful information on the level of the optimization and adjustment of the PSO weights. Therefore, instead of direct fitness values of the agents over the iterations, their mirrored positions projected on the search space are considered for adaptive tuning. The fitness history values, however, can be informative about the behavior and geometry of the objective function throughout the optimization which is not considered in APSO. Another novel fitness-based parameter setting is used for updates in adaptive cuckoo search (ACS) [49]. In the ACS, the fitness values of different agents in the population are compared with the worst and the best agents to adjust the movements in each direction of the space for a given agent. However, the fitness history is not used and it is still dependent on the iteration number.

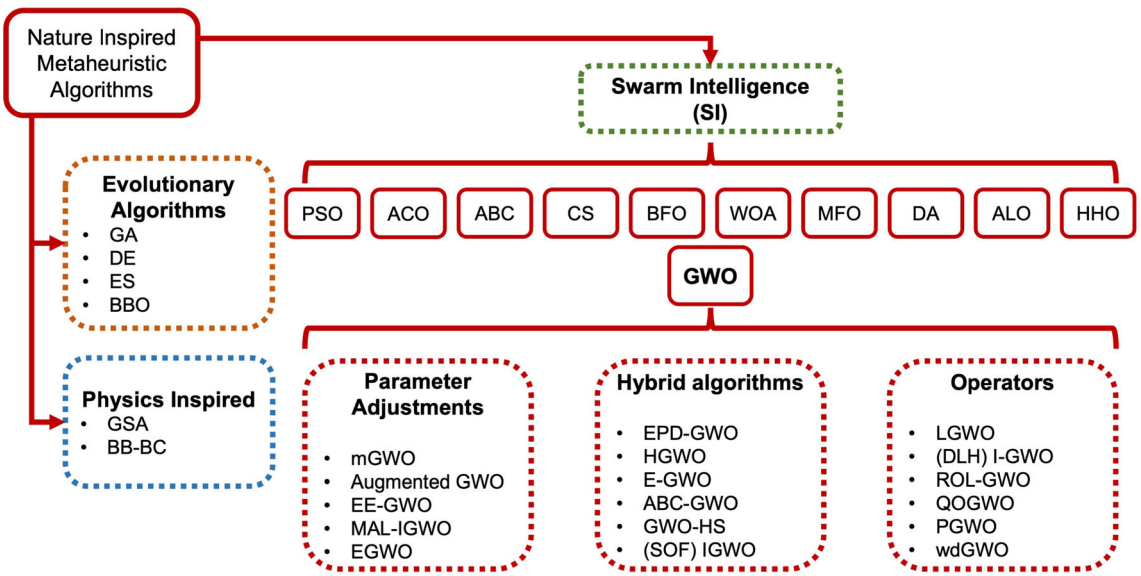

Fig. 1 Classification of nature-inspired metaheuristic algorithms, with the focus on swarm intelligence (SI) algorithms. Variants of grey wolf optimizer (GWO) are categorized based on the method of modification. (Algorithms are mentioned and referenced in the introduction and related works section)

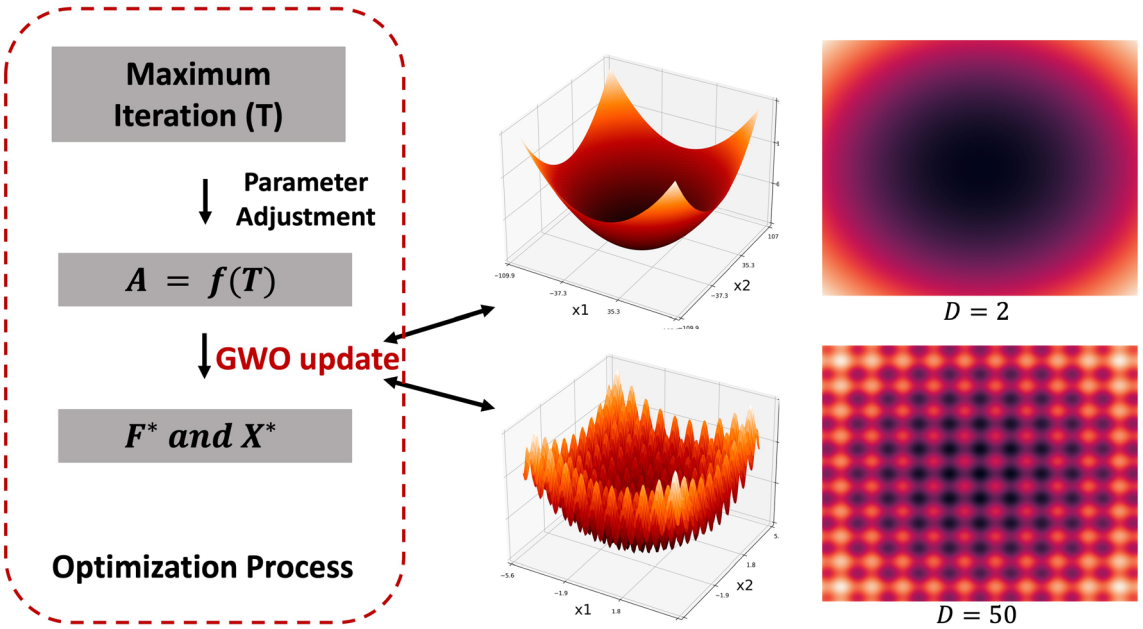

Fig. 2 Optimization process scheme for non-adaptive metaheuristic algorithms in the vanilla GWO and other variants. Parameter adjustment as a function of time is the first step of the optimization prior to algorithm updates on the objective functions. The objective functions can be relatively simple and low-dimensional (top) or more complex and high-dimensional

3 Grey wolf optimizer

Our paper is built upon a swarm intelligence (SI) algorithm named Grey Wolf Optimizer (GWO) that mimics the hunting behavior of wolf packs. GWO method mathematically models how grey wolves search, encircle, and attack the prey. First, it considers the social hierarchy of various types of wolves in a wolf pack. To this end, wolves are categorized into four different kinds that have hierarchical dominance over each other. Three types of Alpha, Beta, and Delta are known as leaders of the pack that are assumed to have superior abilities. The Omega wolves are subordinate ones that follow the navigation of the leaders (Fig. 3).

Grey wolf hunting consists of three main steps of searching for prey, encircling and harassing the prey until it stops moving, and at last attacking it. The encircling process can be modeled mathematically by updating the position of each wolf in relation to the position of the prey. Equation 1a–d can be used to update the position of the wolves for the next iteration.

$$\mathbf{r}_1 \in [0,1], \quad \mathbf{r}_2 \in [0,1], \quad a = 2\left(1 - \frac{t}{T}\right) \tag{1a}$$

$$\mathbf{A} = 2a\mathbf{r}_1 - a, \quad \mathbf{C} = 2\mathbf{r}_2 \tag{1b}$$

$$\mathbf{D} = |\mathbf{C}\mathbf{X}_p(t) - \mathbf{X}(t)| \tag{1c}$$

$$\mathbf{X}(t+1) = \mathbf{X}_p(t) - \mathbf{A}.\mathbf{D} \tag{1d}$$

where r_1, r_2 are uniform random variables that make the algorithm intrinsically stochastic and a is the tuning parameter for exploration and exploitation which is decreased linearly from 2 to zero over iterations. The parameters A, C are the only two adjustable parameters in the GWO algorithm.

The assumption used in the previous equations is that the position of the prey is known. However, this is not true in an abstract search space where we are optimizing an objective function. The GWO algorithm assumes the idea that hopefully the best three agents in the field, i.e., the alpha, beta, and delta wolves, are navigating toward the prey and are closer to the optimal position. Based on this assumption, all the wolves would follow their behavior and update their position with the estimated position of prey defined as the average of the positions of these leader wolves (Fig. 4, Eq. 2a–c).

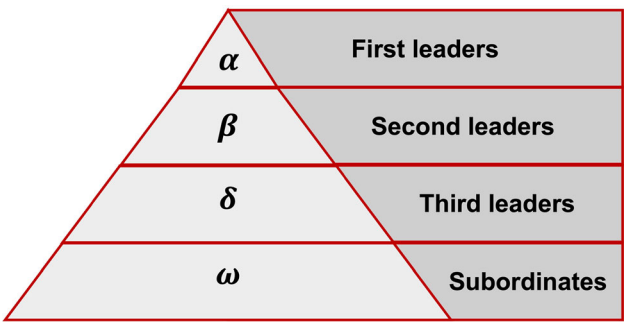

Fig. 3 Social Hierarchy of the grey wolves. The first three types are considered as the leaders and the ω wolves are considered as subordinate agents that follow the navigation of the leaders

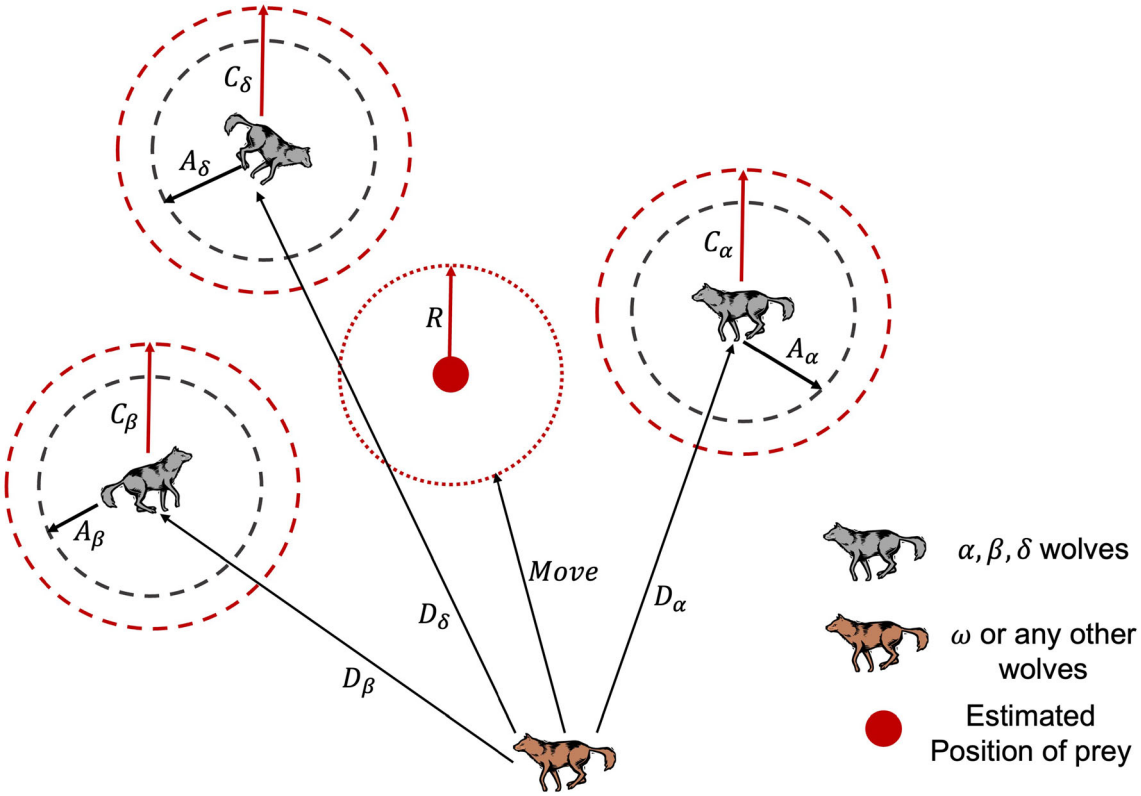

Fig. 4 The position of the wolves is updated based on the estimated position of prey (solid red circle) which is the average of the positions of three leader wolves. The new position depends on parameters A and C which are stochastic. These parameters can be computed by Eqs. 1, 2. The update above is more related to the exploitation phase where the positions are updated close to the best agents

$$\begin{aligned} \mathbf{D}_\alpha &= |\mathbf{C}_\alpha.\mathbf{X}_\alpha(t) - \mathbf{X}(t)| \\ \mathbf{D}_\beta &= |\mathbf{C}_\beta.\mathbf{X}_\beta(t) - \mathbf{X}(t)| \\ \mathbf{D}_\delta &= |\mathbf{C}_\delta.\mathbf{X}_\delta(t) - \mathbf{X}(t)| \end{aligned} \tag{2a}$$

$$\begin{aligned} \mathbf{X}_1 &= \mathbf{X}_\alpha(t) - \mathbf{A}_\alpha.\mathbf{D}_\alpha \\ \mathbf{X}_2 &= \mathbf{X}_\beta(t) - \mathbf{A}_\beta.\mathbf{D}_\beta \\ \mathbf{X}_3 &= \mathbf{X}_\delta(t) - \mathbf{A}_\delta.\mathbf{D}_\delta \end{aligned} \tag{2b}$$

$$\mathbf{X}(t+1) = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3} \tag{2c}$$

4 Adaptive GWO method

One of the main limitations of the GWO algorithm is that it requires us to select some maximum iteration value T which influences the computational time of the optimization. If this parameter is selected too small, we would have premature convergence to a not satisfactory point which is probably far from the real minimizer of the objective function. On the other hand, increasing this parameter to make sure convergence to a presumably good point may result in wasting computational time for non-significant improvements in the quality of the final solution. Therefore, a balance is required for the setting of this parameter. However, in most practical cases, there is no or little information about the complexity of the function and the corresponding enough maximum iteration that balances between time and effectiveness of the optimization algorithm. Finding stopping criteria that are independent of the number of iterations can bring about efficient optimization with the goal of reaching a sufficiently good optimal value in the shortest amount of time.

In addition to the stopping of the optimization, the balance of exploration and exploitation throughout the optimization is a very important aspect in convergence to a good optimal point. In the GWO algorithm, the variable A is the critical parameter for the shift of the algorithm from exploration in the initial iterations toward more exploitation at the end of the optimization process. The essence of adjustable variable A is such that when $|A| > 1$, the wolves diverge from the currently estimated prey and search for better points, i.e., exploration, and when $|A| < 1$, they approach and attack the currently estimated prey, i.e., exploitation (Fig. 5a). Based on Eq. 1b, this parameter is a random variable in the range $[-a, a]$, where a decreases linearly over time. Hence, we can roughly state that the first half of the optimization is spent on exploring the search

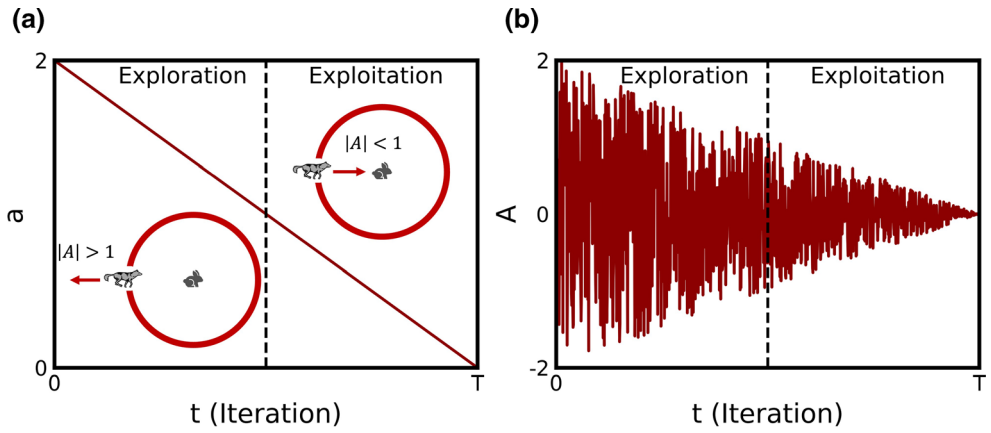

Fig. 5 The evolution of parameter **a)** a, and **b)** the adjustable variable A during the optimization process for T iterations. A is a random value uniformly in the range of $[-a, a]$

space and the agents exploit the so far achieved information in the second half of the optimization (Fig. 5b).

The main goal of our proposed method is to make the optimization algorithm independent of the number of iterations and avoid setting the maximum iteration in advance of the optimization process. Optimization problems cover a wide variety of simple and very complex objective functions which require different computational resources to reach a sufficiently good optimal value that satisfies the requirements of the problem. For example, a simple two-dimensional unimodal objective function may need much less computational time and iterations in comparison with a large-scale, constrained function. However, a common condition is that we are not aware of how complex the function is before optimization. Furthermore, the correlation between the complexity and the maximum required iterations is unknown. In addition, the linearly decreasing behavior of the exploration/exploitation parameter A may not be the optimal behavior for different functions, put aside the fact that the parameter's update is itself dependent on the time (Eq. 1b).

The key idea of adaptive grey wolf optimizer (AGWO) is to leverage the history of the optimization, specifically the average fitness of the agents in the previous iterations in order to adjust the optimization parameters based on the behavior of the function values. In fact, the changes in the fitness values can lead us to select an appropriate exploration parameter and to evaluate the state of the agents in the optimization (Fig. 6a). To this end, we make use of the memory of the system to store a list of average fitness values and to compute their moving average in the recent iterations as in the following equations:

$$\bar{F}^{(t)} = \frac{\sum_{i\in[N]} F_i{}^{(t)}}{N} \tag{3a}$$

$$F^{(t)}_{movavg} = \frac{\sum_{t-w}^{t} \bar{F}^{(t)}}{w} \tag{3b}$$

where N is the population size, w denotes the window length for the moving average which we set as $w = 10$ in the usual case, and i and t indicate the current agent and iteration, respectively. In the first w steps of the optimization before obtaining enough fitness points to compute the moving average, we perform a full exploration period with $a = 2$ to initialize the search space by selecting some random points with the most exploration rate possible.

The evaluation of the current fitness score and its moving average shows that when the current iteration average fitness is improved, i.e., decreased, in comparison with the moving average, it implies that the currently selected exploration/exploitation rate is probably performing well enough. Generally, two cases can happen that may hinder the performance of the optimization. First, search agents can get stuck in a locally optimal solution, i.e., high value in a minimization problem, and then very small improvements can be considered as good performance. The second case is the possibility of the agents wandering around at a high exploration rate where sudden small improvements can keep the exploration parameter high and hinder the algorithm from converging to a minimum value. To alleviate the first problem, similar to many other algorithms, we have to start the optimization with the highest exploration rate to allow the search agents to sufficiently search the space. Then, we can reduce the exploration rate and increase the exploitation over time. To address the second issue, we propose *a damping parameter*, i.e., reduction factor, γ that can reduce the exploration rate whenever a significant improvement is not observed over the recent iterations (Fig. 6b). This parameter reduction, in some sense, is analogous to learning rate decay commonly used in gradient-based methods [50].

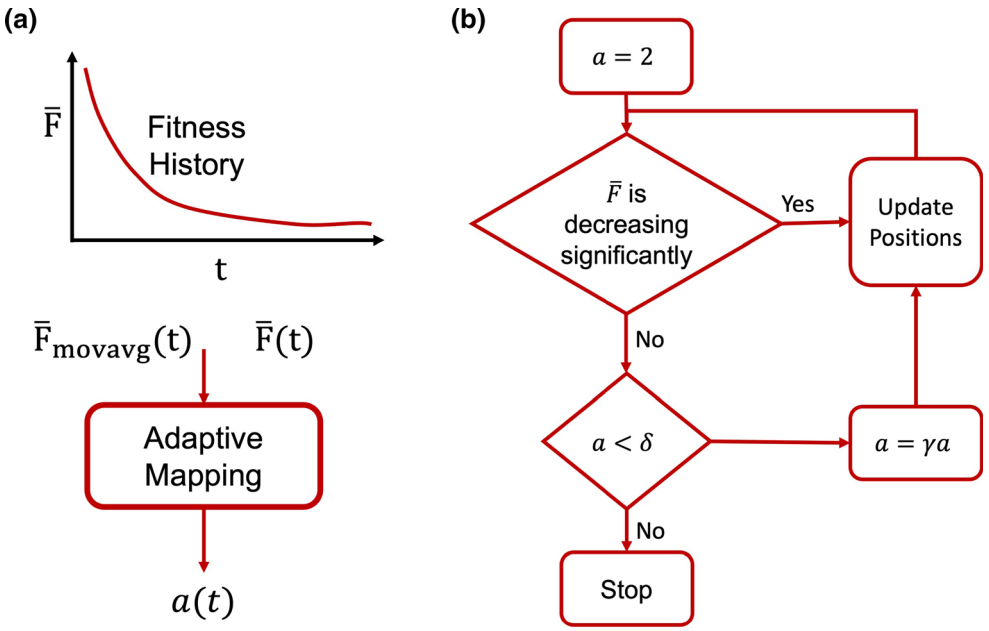

Fig. 6 **a** The schematic of the idea of adaptive parameter setting using the information stored from the history of the optimization. **b** The flowchart of the stopping criteria and adjusting method of parameter a during the optimization from the initial step. δ is set to 10^{-3} and $\gamma = 0.95$

Algorithm 1: Adaptive Grey Wolf Optimizer (AGWO); the base algorithm is taken from GWO [18]

Initialize the grey wolf population X_i $(i = 1, 2, ..., N)$
Set $a = 2$, and calculate A, C using Equation 1
Calculate the fitness of each search agent and store their average value $\bar{F}$
X_α = the best search agent
X_β = the second best search agent
X_δ = the third best search agent
F_{movavg} = Moving average of fitness values over last w iterations
`Stage 1: Fully Explore for a set number of iterations and keep track of required variables`
for $t \leq$ *Full Exploration Time* **do**
  Update the positions of the search agents using Equations 1, 2.
  Update A, and C, X_α, X_β, and X_δ
  Calculate the fitness of all search agents and store $\bar{F}$, and compute the Moving Average of F (Equations 3)
`Stage 2: Adaptive Strategy. Stop when either` a `becomes smaller than threshold AND function values are not changing significantly`
while $(a > \delta \ OR \ \bar{F} < F_{movavg} - \epsilon \ )$ **do**
  Update the positions of the current search agents using Equations 1, 2.
  if $(\bar{F} \geq F_{movavg} - \epsilon)$ **then**
    Update exploration parameter as $a = \gamma a$
  Update A, and C, X_α, X_β, and X_δ
  Calculate the fitness of all search agents and store $\bar{F}$, and compute the Moving Average of F (Equations 3)
return X_α

In many optimization problems, the goal is to reach a good enough solution in the shortest time, i.e., computational cost. In optimization problems, the optimal function value is usually unknown but the degree to which the precision of the answer is acceptable is known. For example, in the engineering design of a tension spring, we are not aware of the optimal weight, yet we can claim that difference in weights of lower than one milligram is negligible. Therefore, we are not willing to allocate more computational time for these small decrements. This is a common approach in different types of optimization problems including gradient-based and gradient-free methods. For example, in gradient descent algorithms, the norm of gradient is compared with a threshold of epsilon $\|\nabla F\|_2^2 < \epsilon$. Here, we define a parameter ϵ which corresponds to the lowest fitness change that can be presumed significant. Hence, we can mathematically model the significant improvement as the following condition:

$$\bar{F}^{(t)} < F_{movavg}^{(t-1)} - \epsilon \tag{4}$$

which means we have good fitness improvement when the function value, on average, is decreased at least by ϵ. The advantage of setting this threshold parameter ϵ over selecting maximum iteration T in the original GWO algorithm is that we have direct control over the goodness of the results. As mentioned before, selected T is highly prone to be too short or too long for the objective function. However, by selecting ϵ we can monitor the function values and make sure the convergence to a satisfactory point.

Here, the stopping criteria are designed to check two conditions. First, if the optimization is not improving the fitness values and second, if exploration/exploitation parameter A is too small to allow the agents to explore the search space and jump to other good points. In this case, we cannot hope for more significant improvements of the function value, and we stop the process. Based on the original algorithm, the value of a defines a hypothetical hypercube in which the agents' positions can be updated. Hence, we define a threshold $\delta = 10^{-3}$ for the smallest a to be considered (Fig. 6b). The complete steps of the Adaptive Grey Wolf Optimizer (AGWO) are elaborated upon in algorithm 1.

Another remark is on the computational complexity of the proposed adaptive algorithm. The additional computation step in the adaptive algorithm consists of keeping the average and then moving average of the fitness values to decide strategy based on them. The initialization, updating of control parameters and positions, and fitness evaluation are $O(Nd)$ where N is number of search agents and d is the dimension of the problem [51]. Also, since the maximum number of iterations for AGWO is the same as GWO, we would have a same upper limit $O(TNd)$ where T is the maximum allowed iterations. However, in the proposed AGWO, we expect the algorithm to stop much earlier than T, with a lower bound that depends on the decaying rate γ, the threshold δ and the problem itself. Starting from $a = a_0$, assuming decaying at every iteration the number of iterations to reach below threshold should satisfy $\gamma^{T_{dec}} a_0 \leq \delta$. Considering w steps of fully exploration (stage 1 in algorithm 1), the lower bound of time will be $w + T_{dec}$ which has to be less than maximum number of iterations to bring about computational efficiency for AGWO. In the results section, we show that this is indeed the case for many problems.

4.1 Fitness curve-dependent adaptive GWO (AGWO^4)

In the proposed AGWO, we explained how fitness values are employed as conditions to have a monotonically decreasing parameter a curve. In this extended version, we suggest another form of adaptive a in the same AGWO platform by only changing the $a = \gamma a$ update formula to a more complicated form that includes information on the fitness values in a more direct way. Therefore, we still have both damping with the damping factor γ, and the condition for the significant decrease since they are necessary for AGWO. In AGWO, the information was used to see the absolute significant reduction in function value in the current iteration t. This form of the update does not contain any information about the fitness curve geometry. However, the behavior of fitness values in the previous iterations can be informative about whether we should keep the same exploration rate or decrease this parameter in favor of more exploitation. In fact, this idea is typically used in other optimization methods like gradient-based algorithms. The history of gradients is used to modify the learning rate in Adagrad [52] and ADAM [53] optimizers that are frequently used in training artificial neural networks (ANNs).

Instead of a paired comparison of fitness values with the average of previous fitness values, we can inspect their behavior in three consecutive iterations. The rate of changes between the last two steps, i.e., ΔF_t, compared to the previous two steps, i.e. ΔF_{t-1}, contains information on the geometry of the curve (Fig. 7a). It also has some analogy to the second derivative of the function (Hessian matrix) which also stochastically informs us about the smoothness of the function. The lower rate implies more smoothness, i.e., going toward convergence, while the higher rate means a favorable reduction, and thus we keep the exploration rate. The flowchart for this extended version is depicted in Fig. 7b. Therefore, the adaptive parameter a is a direct function of fitness values and is computed as follows:

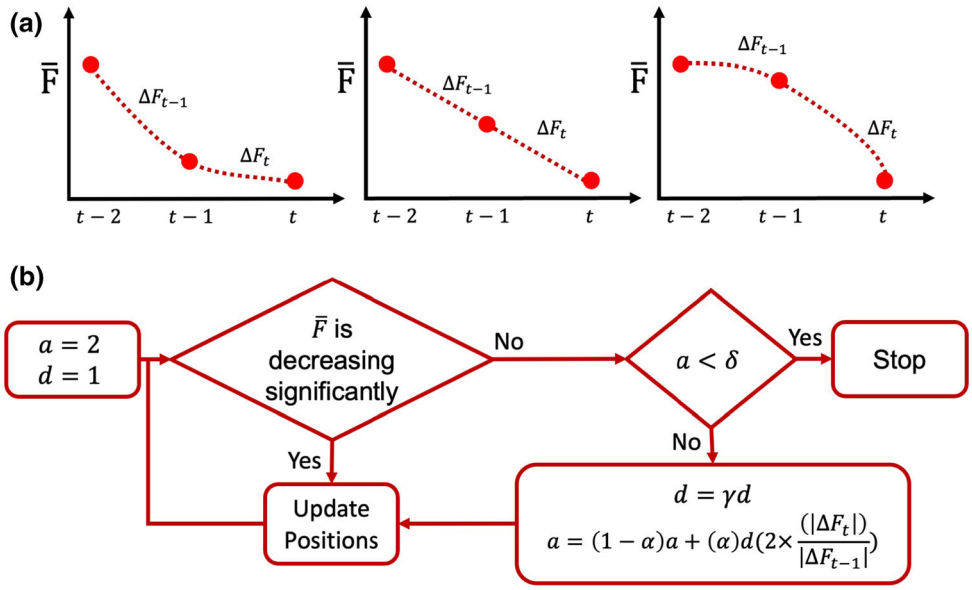

Fig. 7 **a** The plots of different types of average fitness curve behavior, and the effect of the curve on the ratio of ΔF_t and ΔF_{t-1}. **b** The flowchart of the stopping criteria and adjusting method of parameter a during the optimization from the initial step for the extended AGWO$^{\Delta}$. δ is set to 10^{-3} and $\gamma = 0.95$

$$a = (1-\alpha)a + (\alpha) \times d \times \left(2 \times \frac{|\Delta F_t|}{|\Delta F_{t-1}|}\right) \tag{5}$$

where α makes the parameter to consider previous exploration rates and change smoothly. This is analogous in a sense to the concept of momentum for gradient-based optimization [54]. In the current work, the value of α is set to 0.1. d in this equation is an auxiliary parameter, reduced by damping factor γ, which similarly to AGWO reduces the parameter a from the initial maximum rate toward full exploitation $a \to 0$.

5 Experimental evaluation and results

In this section, the performance of the proposed AGWO and its extended version AGWO$^{\Delta}$ are evaluated by comparing them to the vanilla GWO and some of its recent variants via performing various experiments and analyses. Since AGWO introduces various additional forms and parameters, its performance needs to be examined not only in the aspect of convergence to better points but also in the viewpoint of computational time and its ability to balance between these two metrics, i.e., convergence to a good point in the shortest possible time. In this section, we first introduce the benchmark functions and the experimental environment used for the tests. Then, we analyze the proposed methods compared to vanilla GWO for their performance on unimodal and multimodal functions for exploitation and exploration ability, respectively. At last, these algorithms are compared to some other recently proposed variants of GWO with modifications to the convergence parameter a. The advantages and limitations of the methods are discussed and illustrated by convergence curves and performance tables.

5.1 Benchmark functions and experimental environment

To test the performance of the optimization algorithms, we employ 23 benchmark functions containing different unimodal and multimodal search spaces that have been widely used in the literature [18, 55]. The minimization of these functions is considered as the objective function, and the algorithms are evaluated both in terms of computational time and the accuracy or the quality of the final solution. The functions consist of seven unimodal functions with the global optima at the origin, and 14 multimodal functions with fixed or variable dimensions. The two-dimensional versions of these benchmark functions are depicted in Figs. 8 and 9, and the functions are listed in Tables 1 and 2.

Metaheuristic algorithms are stochastic, so we run each algorithm on every benchmark function multiple times and report the average and standard deviation of the best fitness score achieved as well as the total number of iterations in the case of adaptive algorithms.

Some other variants of GWO modified the parameter a which is known as exploration/exploitation or convergence parameter usually to enhance the exploration of the GWO and converge to better points. Some of these variants along

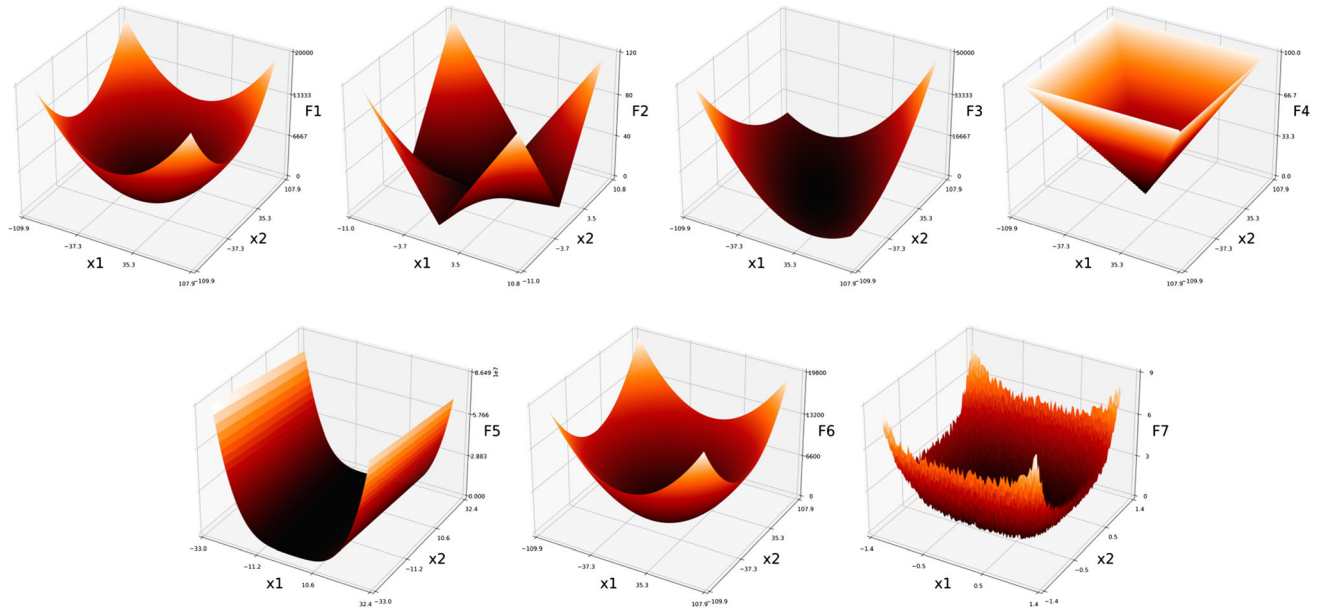

Fig. 8 2D version of unimodal test functions (30 dimensional versions are used for the experiments)

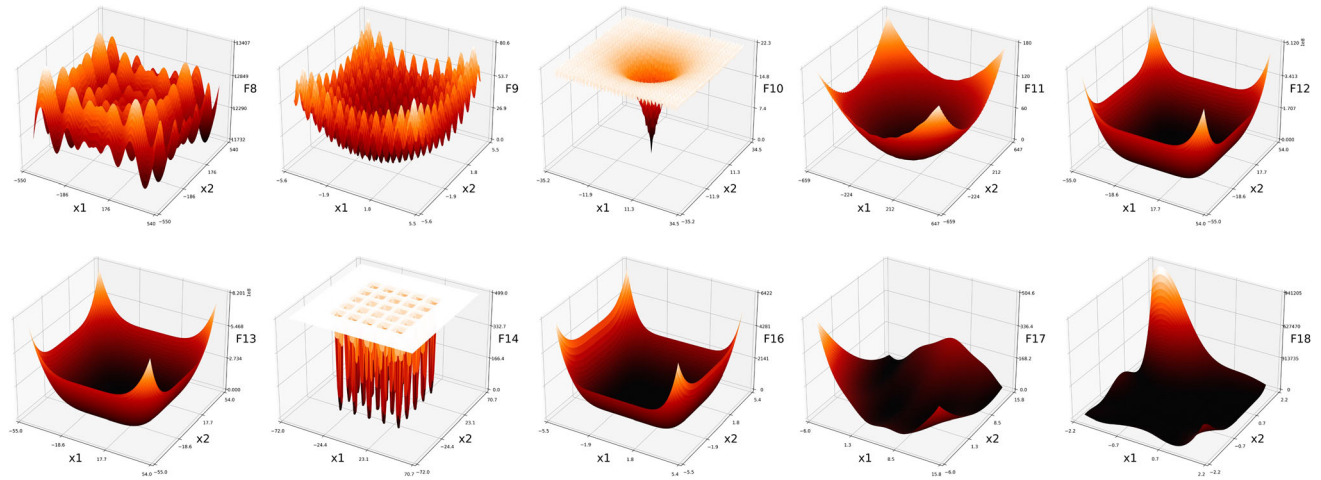

Fig. 9 2D version of some multimodal test functions (F_8–F_{13}) and fixed dimension multimodal functions (F_{14}–F_{18})

Table 1 Unimodal benchmark functions used for the evaluation of AGWO

Function	Range	Dim
$F_1(x) = \sum_{i=1}^{N} x_i^2$	[− 100, 100]	30
$F_2(x) = \sum_{i=1}^{N} \lvert x_i \rvert + \Pi_{i=1}^{N} \lvert x_i \rvert$	[− 10, 10]	30
$F_3(x) = \sum_{i=1}^{N} (\sum_{j-1}^{i} x_j)^2$	[− 100, 100]	30
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le N\}$	[− 100, 100]	30
$F_5(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[− 30, 30]	30
$F_6(x) = \sum_{i=1}^{N} ([x_i + 0.5])^2$	[− 100, 100]	30
$F_7(x) = \sum_{i=1}^{N} i x_i^4 + random[0, 1)$	[− 1.28, 1.28]	30

with their nonlinear parameter adjustments are tabulated in Table 3. We compare our proposed algorithm with some of these methods to analyze the effect of exploration/exploitation parameter a on the optimization performance. Note that only the parameter adjustment of a is following the update equations given in these algorithms and other modifications applied in some of these variants have not been considered in the experiments to have a fair comparison of the exploration/exploitation parameter.

5.2 Convergence analysis in GWO

In the first experiment, we inspect the effect of maximum iteration (T). The importance of the maximum iteration T can be understood by comparing the convergence curves

Table 2 Multimodal benchmark functions used for the evaluation of AGWO

Function	Range	Dim
$F_8(x) = \sum_{i=1}^{N} -x_i \sin\sqrt{\lvert x_i\rvert} + 12569.487$	$[-500, 500]$	30
$F_9(x) = \sum_{i=1}^{N} [x_i^2 - 10\cos 2\pi x_i + 10]$	$[-5.12, 5.12]$	30
$F_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} - exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)) + 20 + e$	$[-32, 32]$	30
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \Pi_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	30
$F_{12}(x) = \frac{\pi}{N}\{10\sin(\pi y_1) + \sum_{i=1}^{N-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+(y_N - 1)^2\} + \sum_{i=1}^{N} u(x_i, 10, 100, 4), \quad y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]$	30
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{N}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+(x_N - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4)$	$[-50, 50]$	30
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1} - 1$	$[-65, 65]$	2
$F_{15}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2 + 0.0027$	$[-5, 5]$	4
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 + 1.03163$	$[-5, 5]$	2
$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 - 0.397$	$[-5, 5]$	2
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] - 3$	$[-2, 2]$	2
$F_{19}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2) + 3.864$	$[1, 3]$	3
$F_{20}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2) + 3.32$	$[0, 1]$	6
$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.1532$	$[0, 10]$	4
$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.4028$	$[0, 10]$	4
$F_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1} + 10.5363$	$[0, 10]$	4

Table 3 Parameter adjustments in some variants of GWO

Algorithm	a	Parameters
GWO [18]	$a = 2(1 - \frac{t}{T})$	–
mGWO [30]	$a = 2(1 - \frac{t^2}{T^2})$	–
Augmented GWO [56]	$a = 2 - \cos(rand)\frac{t}{T}$	–
EE-GWO [35]	$a = 2(1 - \frac{t}{T})^{\mu}$	$\mu = 1.5$
MAL-IGWO [31]	$a = (1 - \frac{t}{T})(1 - \mu\frac{t}{T})^{-1}$	$\mu = 1.1$
AGWO	γ damping when F is not decreasing significantly	

and the curves of average fitness value over time for different choices of T (Fig. 10). It is observed that increasing iterations usually would improve the final fitness; however, the convergence is slower. The underlying reason is the exploration/exploitation parameter a that is different at a given time step for different choices of T, i.e., it is dependent on T. Hence, there is a trade-off between the accuracy and the computational time which strongly depends on the function which we usually do not know beforehand.

In these experiments, we consider two cases of AGWO with a significance threshold $\epsilon = 10^{-5}$ and with $\epsilon = 0$ that means no significance threshold is required for the decrease in the consecutive iterations. In the former case, the ultimate goal is to reach a sufficiently good minimum with the significance defined by ϵ and the most important aspect of AGWO is how quickly it can converge to that point.

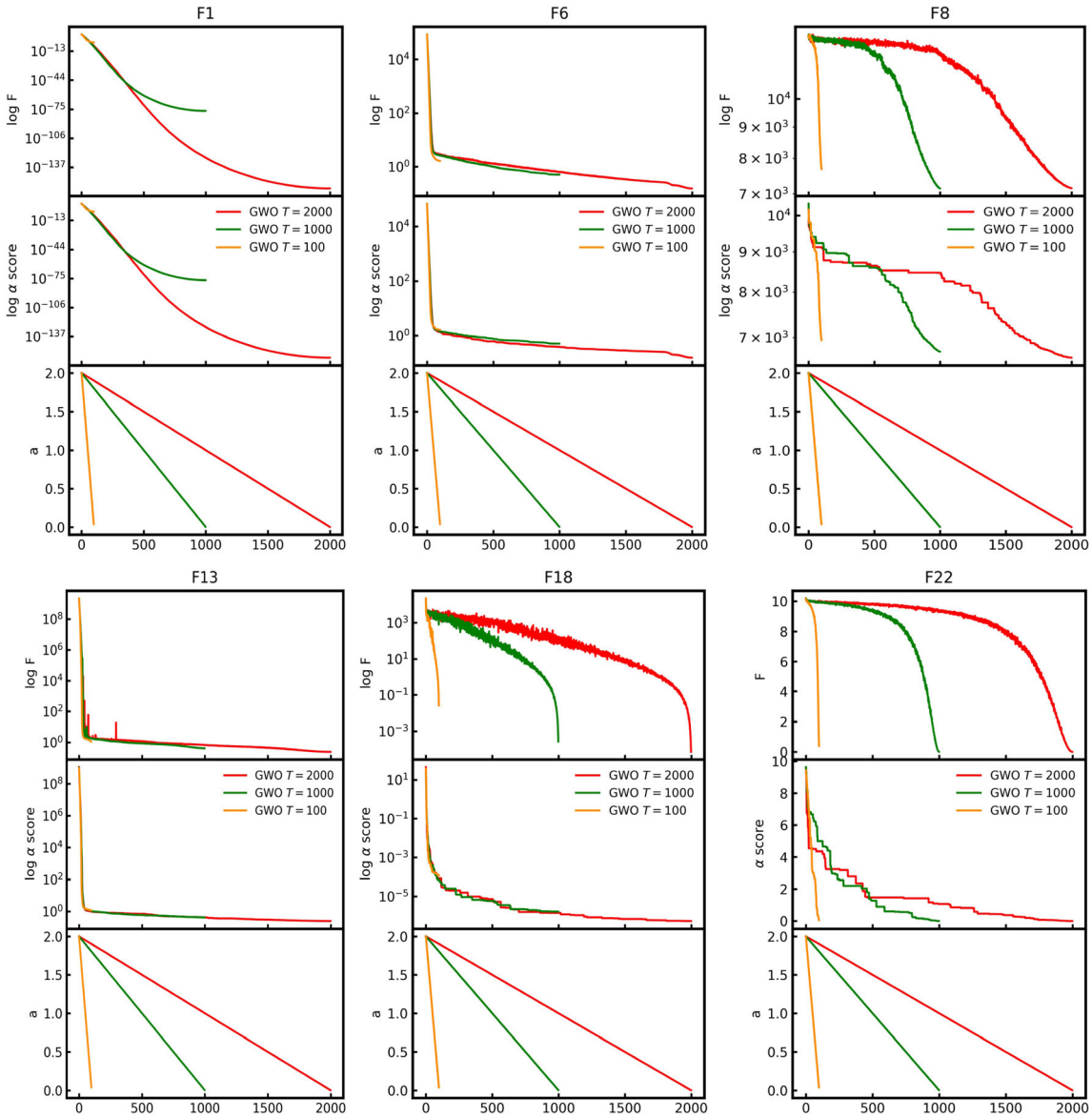

Fig. 10 Comparison of average fitness value (top), convergence curve or alpha score in GWO (middle), and exploration/exploitation parameter a for maximum iteration $T = 100$, $T = 1000$, $T = 2000$ plotted for the experiments on some of the benchmark functions

Therefore, we compare the average of stopping iteration for the vanilla GWO algorithm and AGWO. The results for the iteration and the fitness values obtained by vanilla GWO and AGWO with a significance threshold of $\epsilon = 10^{-5}$ are tabulated in Tables 4 and 5. A pairwise t-test with $\alpha = 0.05$ is used to examine the significance of changes in the iteration numbers between the two algorithms. The results show that AGWO, on average, approximately needs 0.51 and 0.57 of the total iterations required for unimodal and multimodal objective functions, respectively. The last column of the tables inspects the effectiveness of the convergence by performing a significance test with the defined ϵ between the two algorithms. In this test, the win/tie/loss (W/T/L) corresponds to these conditions:

$$
\begin{aligned}
W: &\quad F^*_{AGWO} < F^*_{GWO} - \epsilon \\
L: &\quad F^*_{GWO} < F^*_{AGWO} - \epsilon \\
T: &\quad Otherwise
\end{aligned}
$$

The results of this experiment indicate the efficiency of AGWO and its ability to automatically find the required computational time for a balanced optimization to a good enough solution. We can see that the number of iterations, on average, is shorter for unimodal functions in comparison with more complex multimodal functions. The iteration numbers required for AGWO are almost always meaningfully improved with 22 wins out of 23 functions, while it is able to preserve its ability to obtain a good fitness

Table 4 Comparison of iteration number and statistical test of fitness change based on significance ϵ for unimodal functions

Objective function	GWO				AGWO $\epsilon = 10^{-5}$				Iteration t test	Fitness significant change
	Stop iteration		Fitness		Stop iteration		Fitness			
	Ave	Std	Ave	Std	Ave	Std	Ave	Std		
F1	999	0	9.974E-77	1.836E-76	240.24	2.9296	4.832E-10	3.236E-10	**W**	T
F2	999	0	8.602E-45	8.783E-45	262.04	2.5997	8.468E-08	3.041E-08	**W**	T
F3	999	0	1.586E-17	4.478E-17	435.4	32.6141	7.201E-07	1.354E-06	**W**	T
F4	999	0	1.436E-16	1.956E-16	371.36	13.5348	1.109E-06	6.362E-07	**W**	T
F5	999	0	26.445427	0.545875	994.08	24.1030	26.42245	0.81135	T	T
F6	999	0	0.380873	0.226604	920.8	83.1264	0.26593	0.23056	**W**	T
F7	999	0	0.000612	0.000246	346.76	9.8317	0.00559	0.00246	**W**	L
Sum =	6993			Sum =	3570.68				W/T	W/T/L
				Ratio =	**0.5106**				6/1	0/6/1

Table 5 Comparison of iteration number and statistical test of fitness change based on significance ϵ for unimodal functions

Objective function	GWO				AGWO $\epsilon = 10^{-5}$				Iteration t test	Fitness significant change
	Stop iteration		Fitness		Stop iteration		Fitness			
	Ave	Std	Ave	Std	Ave	Std	Ave	Std		
F8	999	0	6003.424	654.442	712.04	256.7585	6939.184	1780.190	**W**	L
F9	999	0	3.208224	9.68689	710.08	334.0003	17.73550	38.33726	**W**	T
F10	999	0	1.451E-14	2.926E-15	266.04	2.8068	1.275E-07	2.907E-08	**W**	T
F11	999	0	0.00229	6.967E-03	310.6	141.3605	0.00284	0.00602	**W**	T
F12	999	0	0.02298	1.313E-02	680.4	160.9713	0.02357	0.01669	**W**	T
F13	999	0	0.351438	1.847E-01	959.12	93.2306	0.27868	0.14179	**W**	T
F14	999	0	1.877959	3.505E+00	424.52	16.4709	2.63314	3.13365	**W**	T
F15	999	0	0.035660	8.984E-03	390.12	22.9038	0.03408	0.00799	**W**	T
F16	999	0	1.552E-06	5.543E-09	425.04	10.8756	1.547E-06	5.187E-11	**W**	T
F17	999	0	0.000888	8.891E-07	509.64	65.3409	0.001168	0.001335	**W**	T
F18	999	0	3.432E-06	3.332E-06	559.04	13.7999	0.000057	0.000043	**W**	L
F19	999	0	0.001539	0.00154	513.64	15.4347	0.000240	0.002350	**W**	**W**
F20	999	0	0.062995	0.07767	557.8	25.0440	0.06957	0.06726	**W**	T
F21	999	0	0.848178	1.94938	710.48	32.8343	1.11085	2.26265	**W**	T
F22	999	0	0.211490	1.03357	720.52	13.5532	− 0.000137	2.516E-06	**W**	T
F23	999	0	0.314461	1.13944	714.92	26.7670	0.32448	1.59014	**W**	T
Sum:	15984			Sum =	9164				W/T	W/T/L
				Ratio =	**0.5733**				14/0	1/11/2

value as in GWO with only one loss in unimodal functions and two losses in multimodal benchmark functions. Interestingly, it shows better performance and wins for F19. Altogether, AGWO with $\epsilon = 10^{-5}$ can satisfy the convergence criteria more efficiently with around half the iterations needed.

It is worth mentioning that some additional stopping criteria have also been used for some single-objective metaheuristic algorithms [57–59], These criteria, for example, can be based on fitness change threshold or distance-based movements. Although these criteria are also applicable for GWO, they are independent of parameter a, which is responsible for the convergence of agents to a point. Hence, the criterion might be satisfied while we are still exploring the space for better solutions.

5.3 Exploration and exploitation evaluation

To evaluate the performance of the algorithms for their exploitation, unimodal functions are usually studied. The majority of unimodal benchmark functions have symmetric bounds with the minimizer at the origin with the best function value of zero. On the other hand, multimodal functions are suitable for the evaluation of the methods for local minima avoidance and their sufficient exploration. Adaptive grey wolf optimizer with $\epsilon = 0$, AGWO with $\epsilon = 10^{-5}$, the extended AGWO$^{\Delta}$ are compared with the vanilla GWO as well as three other algorithms, mGWO [30], EE-GWO [35], and augmented GWO [56] with the parameter a updated following the equations given in Table 3. AGWO algorithms have external stopping criteria which save computational time when the parameter a becomes smaller than a threshold. However, we can still compare the best fitness value obtained by each algorithm throughout the optimization. The average and standard deviation of f^* is reported for each algorithm in Tables 6 and 7 corresponding to unimodal and multimodal functions, respectively.

For unimodal functions, the extended version AGWO$^{\Delta}$ shows superior performance for some of the benchmark functions, specifically F1, F2, and F4. Two other GWO variants, augmented GWO and mGWO, which enhanced exploration rate also appeared to have better performance than vanilla GWO for unimodal functions. A potential reason is that after some time (around half of the iterations), the rate of a becomes so small that it will not allow searching the space for better solutions. The change of slope in the convergence curves shown in the next section also supports this reason. This observation implies that keeping a high exploration rate would result in finding better points in comparison with spending computational time for exploitation. We will analyze the convergence curve in the next section to support this claim. AGWO with a nonzero significance threshold ϵ is not successful in outperforming other variants in terms of fitness score. However, the results are comparable with the average iteration ratio of 0.51. Setting $\epsilon = 0$, however, brings about better solutions obtained at the cost of higher

Table 6 The comparison of solutions obtained by GWO variants on unimodal functions

Objective function		GWO	mGWO	EE-GWO	Augmented GWO	AGWO $\epsilon = 10^{-5}$	AGWO $\epsilon = 0$	AGWO$^{\Delta}$
F1	Ave	9.973573E-77	3.569268E-99	1.251373E-59	2.550738E-133	4.832403E-10	2.722891E-112	**1.250434E-142**
	Std	1.836394E-76	6.949881E-99	3.010570E-59	1.155946E-132	3.235913E-10	1.004929E-111	3.024885E-142
F2	Ave	8.601920E-45	1.916219E-57	9.317989E-35	1.337620E-78	8.467880E-08	4.215620E-64	**2.379787E-89**
	Std	8.783130E-45	2.297146E-57	6.599263E-35	2.293195E-78	3.040536E-08	5.174775E-64	3.445475E-89
F3	Ave	1.586094E-17	1.537065E-21	1.329863E-13	**5.201524E-25**	7.201460E-07	5.108610E-14	1.883386E-15
	Std	4.478424E-17	5.391938E-21	5.144514E-13	1.615881E-24	1.353512E-06	2.190410E-13	4.924030E-15
F4	Ave	1.435774E-16	4.108489E-23	2.269956E-12	5.234153E-30	1.108541E-06	1.907816E-29	**3.153981E-34**
	Std	1.955821E-16	6.544397E-23	2.885322E-12	1.427172E-29	6.362363E-07	3.598499E-29	1.091501E-33
F5	Ave	2.644543E+01	2.647055E+01	2.641240E+01	2.639697E+01	2.642245E+01	**2.603314E+01**	2.685019E+01
	Std	5.458750E-01	6.611032E-01	8.989719E-01	5.993160E-01	8.113471E-01	8.540912E-01	8.801562E-01
F6	Ave	3.808731E-01	**1.806732E-01**	3.183215E-01	5.466271E-01	2.659270E-01	1.895517E-01	9.027241E-01
	Std	2.266045E-01	1.668525E-01	2.269263E-01	3.241510E-01	2.305588E-01	2.385307E-01	3.742916E-01
F7	Ave	6.120804E-04	**4.453983E-04**	1.093969E-03	4.824504E-04	5.592718E-03	6.298299E-03	5.243435E-03
	Std	2.464215E-04	2.367432E-04	6.339044E-04	2.371176E-04	2.464485E-03	2.639767E-03	3.378296E-03
Average iteration ratio		1.0	1.0	1.0	1.0	0.5106	0.9058	0.9069

Table 7 The comparison of solutions obtained by GWO variants on multimodal functions

Objective function		GWO	mGWO	EE-GWO	Augmented GWO	AGWO $\epsilon = 10^{-5}$	AGWO $\epsilon = 0$	AGWO^4
F8	Ave	**6.003424E+03**	6.861190E+03	6.167859E+03	8.568503E+03	6.939184E+03	7.229337E+03	7.021886E+03
	Std	6.544422E+02	1.196285E+03	1.149780E+03	3.618334E+02	1.780190E+03	1.634803E+03	1.653241E+03
F9	Ave	3.208224E+00	**8.138274E-01**	4.679572E+00	1.356495E+00	1.773550E+01	4.857344E+00	1.749568E+01
	Std	9.686892E+00	2.215883E+00	3.706599E+00	4.772586E+00	3.833726E+01	5.877942E+00	3.038730E+01
F10	Ave	1.451284E-14	**9.112711E-15**	2.090772E-14	**9.112711E-15**	1.275094E-07	1.138645E-14	1.209699E-14
	Std	2.926194E-15	2.029716E-15	4.177121E-15	2.477747E-15	2.906776E-08	3.469312E-15	3.100442E-15
F11	Ave	2.294535E-03	**1.345154E-03**	2.063429E-03	2.723239E-03	2.836616E-03	4.400894E-03	3.692693E-03
	Std	6.966651E-03	4.569892E-03	5.115631E-03	5.834913E-03	6.015639E-03	1.122460E-02	7.387366E-03
F12	Ave	2.297590E-02	1.985799E-02	2.659027E-02	3.742882E-02	2.356580E-02	**1.911451E-02**	4.730619E-02
	Std	1.313084E-02	1.387326E-02	1.557176E-02	1.954764E-02	1.669214E-02	8.544951E-03	3.663199E-02
F13	Ave	3.514380E-01	3.493026E-01	2.896706E-01	4.650744E-01	2.786794E-01	**2.505606E-01**	1.073217E+00
	Std	1.846669E-01	1.931763E-01	1.706938E-01	2.129423E-01	1.417926E-01	1.465826E-01	3.564703E-01
F14	Ave	1.877959E+00	**1.801666E+00**	2.659174E+00	2.662245E+00	2.633143E+00	2.715619E+00	1.851291E+00
	Std	3.504592E+00	3.305700E+00	4.045153E+00	3.856012E+00	3.133647E+00	2.831432E+00	2.612035E+00
F15	Ave	3.565976E-02	**3.172186E-02**	3.485752E-02	3.241719E-02	3.407818E-02	3.407012E-02	3.413328E-02
	Std	8.983974E-03	5.416689E-03	8.546793E-03	6.516292E-03	7.994503E-03	8.000542E-03	7.970350E-03
F16	Ave	1.552214E-06	1.560496E-06	1.546527E-06	3.273902E-06	1.546572E-06	**1.546518E-06**	2.819637E-05
	Std	5.543283E-09	1.068046E-08	1.221951E-11	1.631636E-06	5.187123E-11	7.597073E-12	1.305570E-04
F17	Ave	**8.879718E-04**	8.891014E-04	8.992262E-04	1.105982E-03	1.168076E-03	9.027510E-04	4.640423E-03
	Std	8.890569E-07	3.100683E-06	5.813000E-05	2.440331E-04	1.334938E-03	7.540426E-05	1.070648E-02
F18	Ave	3.432271E-06	1.712564E-06	3.673861E-06	**1.694000E-06**	5.658500E-05	6.872778E-05	5.541708E-05
	Std	3.331875E-06	1.972157E-06	4.288000E-06	2.547543E-06	4.321400E-05	6.725822E-05	6.075841E-05
F19	Ave	1.538612E-03	1.711966E-03	2.317183E-03	2.130338E-03	**2.395817E-04**	1.909137E-03	2.171134E-03
	Std	1.543592E-03	1.642117E-03	2.609824E-03	2.012589E-03	2.350427E-03	2.140155E-03	2.275362E-03
F20	Ave	6.299468E-02	7.517542E-02	8.473599E-02	7.667941E-02	6.957426E-02	**3.678583E-02**	6.515415E-02
	Std	7.767082E-02	7.332215E-02	6.886749E-02	7.456391E-02	6.725735E-02	5.654594E-02	7.470911E-02
F21	Ave	8.481779E-01	1.014452E+00	**4.041968E-01**	1.721867E+00	1.110852E+00	1.488746E+00	1.136739E+00
	Std	1.949376E+00	2.024849E+00	1.370689E+00	1.485449E+00	2.262650E+00	2.692206E+00	2.321960E+00
F22	Ave	2.114896E-01	2.150786E-01	**−1.379508E-04**	1.367369E+00	**−1.367058E-04**	5.168687E-01	3.053445E-01
	Std	1.033566E+00	1.041060E+00	1.262276E-06	5.913019E-01	2.515850E-06	1.785181E+00	1.496548E+00
F23	Ave	3.144613E-01	2.186937E-01	3.244796E-01	1.609419E+00	3.244810E-01	**2.143249E-01**	2.679329E-01
	Std	1.139442E+00	1.059229E+00	1.590145E+00	1.458931E+00	1.590145E+00	1.050491E+00	1.313119E+00
Average iteration ratio		1.0	1.0	1.0	1.0	0.5733	0.7019	0.7754

computational time (iteration ratio of 0.9). These results show the influence of new parameter ϵ that can meaningfully affect the optimization. As we claimed in the methodology section, we can see how this parameter is more meaningful and can be easily adjusted for desired performance in comparison with the unknown maximum iteration T.

Inspection of the results for multimodal functions indicates that instead of the three-point AGWO^4, AGWO with $\epsilon = 0$ shows good convergence, specifically on F12, F13, F16, F20, and F23 test functions with an average iteration ratio of 0.7 of the total iterations of other variants. Also, AGWO with $\epsilon = 10^{-5}$ with a 0.57 average iteration ratio has the best obtained solution for F19 and F22. Unlike unimodal functions, augmented GWO has relatively poor performance on some of the multimodal functions. This observation denotes the importance of keeping the balance between exploration and exploitation. In fact, biasing toward more exploration or more exploitation boosts the performance of the algorithm only for some specific

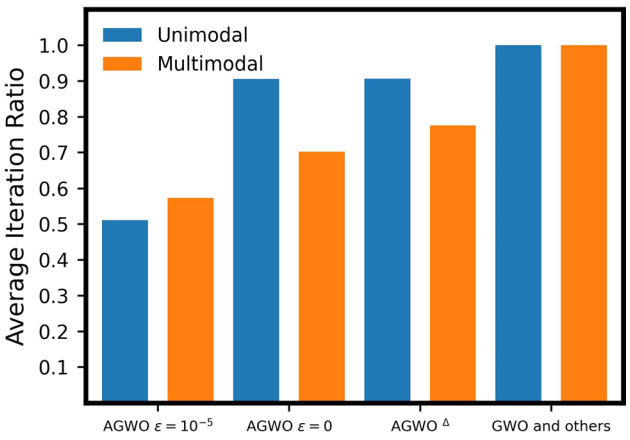

Fig. 11 Bar chart of average iteration ratio of adaptive grey wolf optimizer (simple and extended version) in comparison with the vanilla GWO and other variants for both unimodal and multimodal benchmark functions

benchmark functions. However, in order to perform well for both unimodal and multimodal functions, an algorithm requires to automatically adjust its exploration rate based on the function behavior which is the idea of adaptive GWO. It is worth emphasizing that the reported numbers in Table 7 are the fitness values obtained in the whole duration of optimization for each algorithm. Average iteration ratios are also shown in Fig. 11 to better illustrate the computational efficiency of the adaptive algorithms for both unimodal and multimodal functions. In the next section, we examine the convergence curves to have a fair comparison of these algorithms.

5.4 Convergence evaluation

The performance of different algorithms can be evaluated by observing how the parameter a and the modified stopping criteria can affect the curves of convergence, i.e., best solution so far, and the average of function values. The average curves of every algorithms over 25 replications are plotted in Figs. 12 and 13 for some of the unimodal and multimodal benchmark functions, respectively. In each plot, the top subplot shows how the average fitness value on all the agents in the population changes through time. The second subplot compares the best alpha score which is the same as the convergence curve for GWO algorithms. And finally, the last subplot shows how exploration/exploitation parameter a is set for different algorithms. This setting is constant for non-adaptive algorithms of GWO, mGWO, EE-GWO, and augmented GWO. However, the adaptive grey wolf optimizer and its extended version proposed adaptive behavior of a based on the average fitness curve in the top row of each plot. To better illustrate the curves, a logarithmic scale is used for some functions. Also, some initial steps are removed from some of the plots to be able to distinguish between convergence curves.

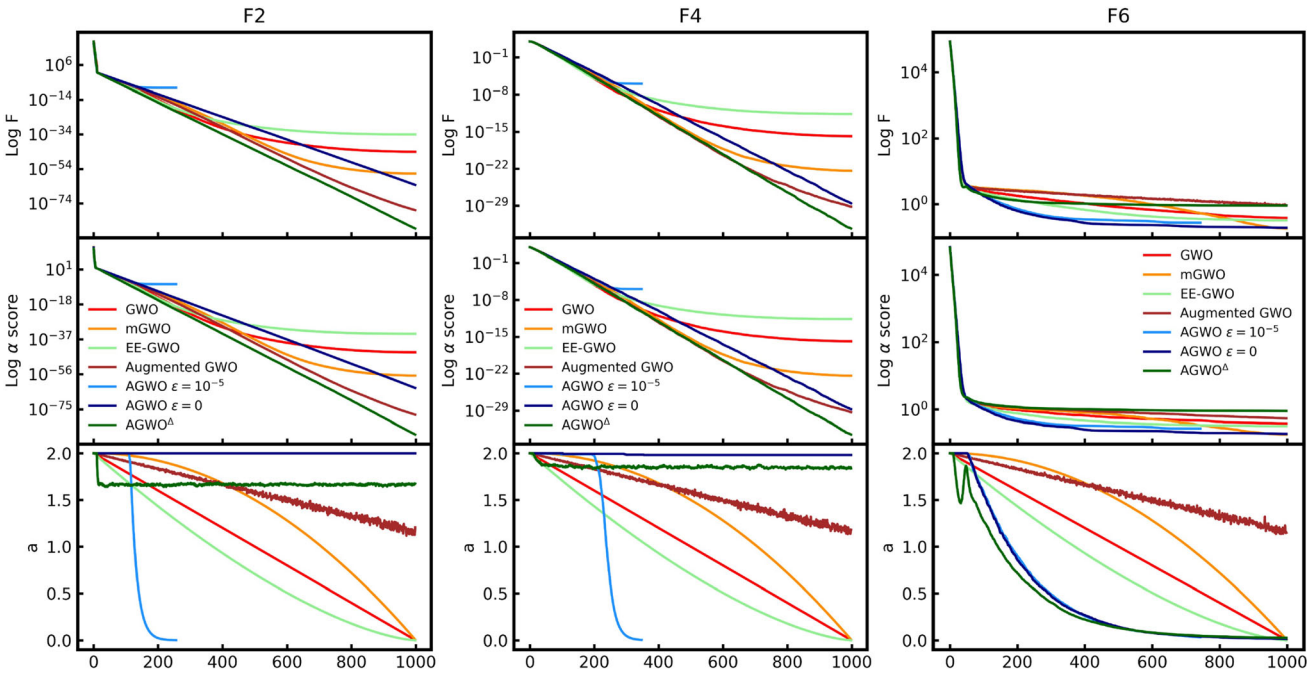

Fig. 12 Convergence curves for some unimodal benchmark functions. The plots consist of three subplots with the horizontal axis being the number of iterations. The average fitness score over all the wolves in the population is plotted in the top row. The middle row shows the convergence curve or the score of the best point found so far. The behavior of parameter a is depicted in the bottom row. For better illustration, a logarithmic scale is used

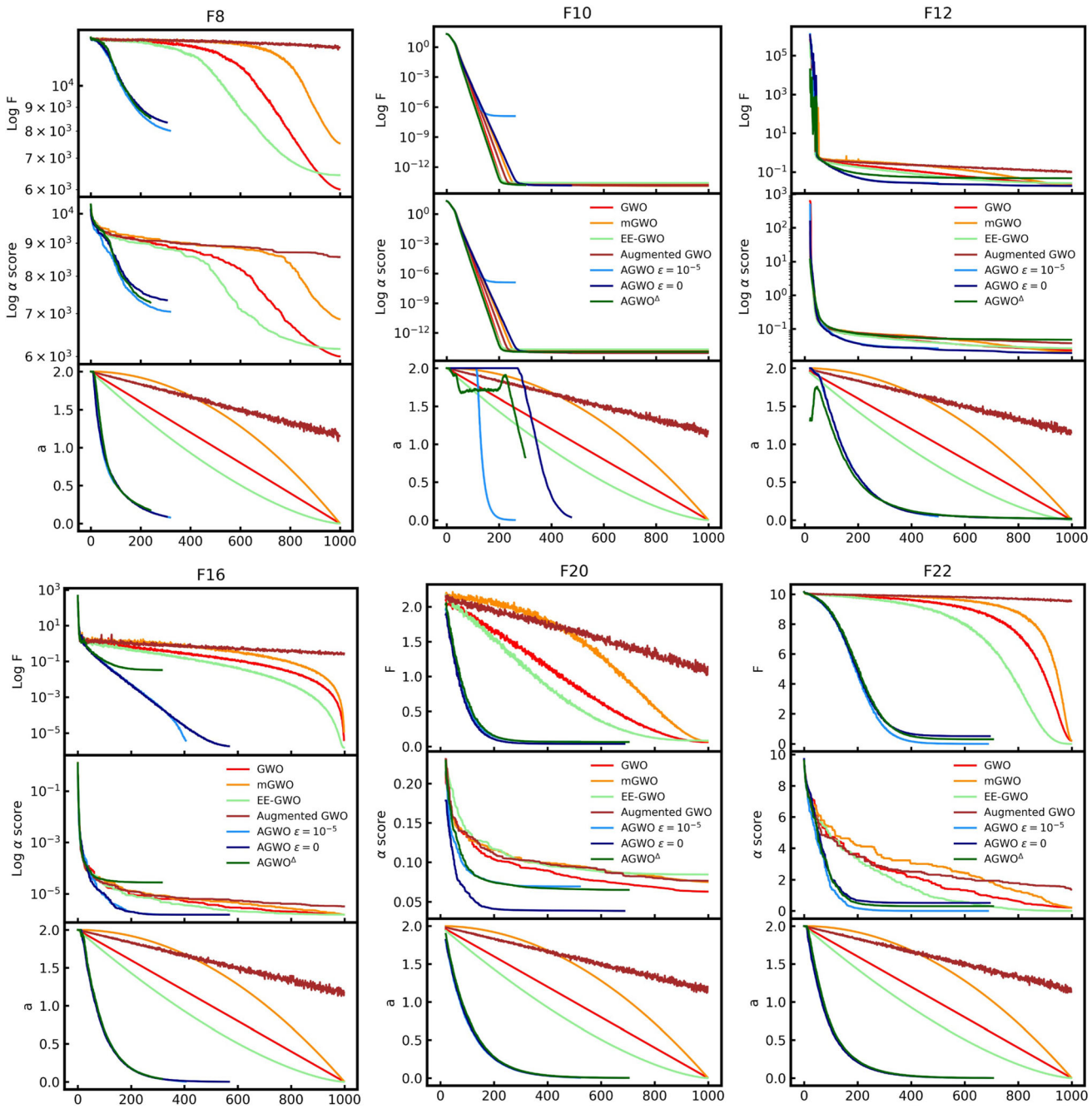

Fig. 13 Convergence curves for some multimodal benchmark functions. The plots consist of three subplots with the horizontal axis being the number of iterations. The average fitness score over all the wolves in the population is plotted in the top row. The middle row shows the convergence curve or the score of the best point found so far. The behavior of parameter a is depicted in the bottom row. For better illustration, a logarithmic scale is used for some functions

From Fig. 12, there are multiple observations from these plots as discussed below:

The effect of ϵ is qualitatively shown in the plots of unimodal functions. Particularly for F1 to F4, we can see early stopping of AGWO $\epsilon = 10^{-5}$ curve after around 200 iterations with the sufficient fitness value achieved. On the other hand, AGWO $\epsilon = 0$ continues optimization with the most exploration rate $a = 2$ for the entire time as shown in Fig. 12 for F2 and F4. As mentioned in the previous section, higher portion of exploration in mGWO, augmented GWO, AGWO $\epsilon = 0$, and AGWO^A enhances the performance for most of the unimodal functions. We can observe the changes in the slope of fitness curves when the algorithms move to the exploitation section in F2 and F4. This suggests that spending time for exploitation with small

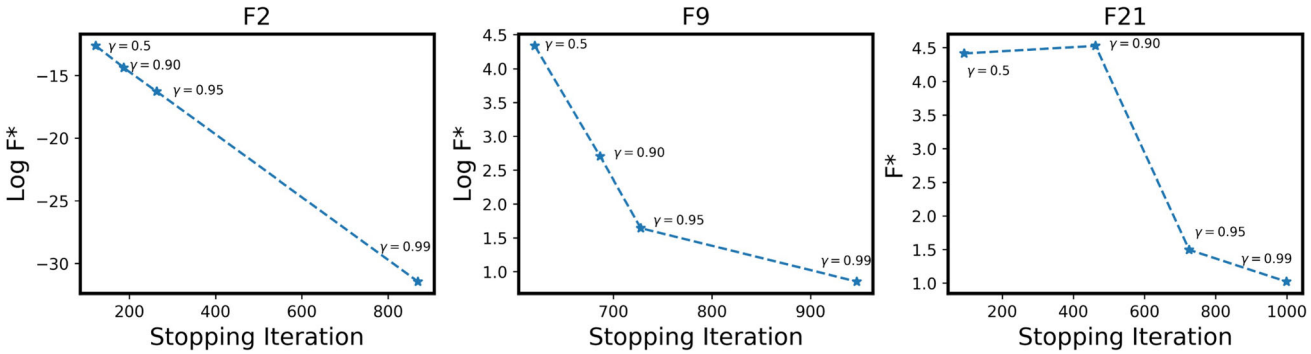

Fig. 14 Final solution with respect to the total number of iterations before stopping for 3 different benchmark functions (F2, F9, F21) which are representative of unimodal, high-dimensional multimodal, and low-dimensional multimodal landscapes, respectively. Each plot shows the corresponding values for 4 different choices of decaying rate γ

parameter a will prevent finding better points further from the so far optimum.

Another interesting observation is that augmented GWO with a slight decrease of a from 2 to 1 through the optimization, shows a better curve than AGWO $\epsilon = 0$. It shows that the desired a is not exactly 2. However, in AGWO we start from the initial value of $a = 2$ and it never decreases to any lower value since the function value is constantly decreasing in average. Surprisingly, the extended AGWO$^{\Delta}$ has superior convergence than augmented GWO for F2 and F4 with a new automatically adjusted a at some $1.5 < a < 2$.

At last, although the overall best fitness score of mGWO is better than AGWO for F6 (as previously shown in Table 6, we can see faster convergence for AGWO algorithms. In fact, if we stop the optimization at the same iteration when AGWO satisfies the stopping criteria, AGWO outperforms mGWO and all other variants in terms of fitness score.

The analysis of multimodal functions plotted in Fig. 13 also indicates some interesting observations:

First, similar to the reported results from the tables, we can see the poor performance of augmented GWO in some of these multimodal functions, while it was a competitive algorithm for unimodal functions. On the other hand, vanilla GWO and EE-GWO that had inferior performance in the unimodal functions showed competitive convergence curves in these plots. This claim shows how a unique form of parameter a can bias the performance toward some specific types of functions. Adaptive tuning of parameter a, however, shows competitive results in both unimodal and multimodal functions.

The plots of F10 show an interesting example of the automatic adjustment happening for AGWO. We can see that the parameter a is held constant at the most exploration rate possible $a = 2$ for around the first 300 iterations when the function value is quickly decreasing. Next, when the agents reach a pseudo-stability in the consecutive iterations without significant improvement, the parameter a starts to be damped by the damping factor and converges to zero in less than 100 iterations. The other blind algorithms, however, continue the optimization with some higher a with spending computational time on non-significant improvements.

For some functions, fast convergence starts to happen at some lower exploration rates which are unknown beforehand. For example, in F22, we can observe that fitness values start to decrease quickly after a reaches values smaller than 1. This observation indicates that a fixed setting of a to spend around half of the optimization time on larger values would waste the computational time. This would result in slower convergence observed for all other variants in comparison with AGWO. This fast convergence of AGWO can be used in many applications where the number of trials is critical and the convergence to a good point in the shortest amount of time is important.

There is a need for a side remark on the effect of another important parameter in the AGWO which is the decaying rate. So far, we have seen the importance of ϵ on the convergence behavior. We have performed several experiments using different choices of decaying rate γ to evaluate its effect on both computational time and final solution. We used four choices of $\gamma = 0.50, 0.90, 0.95, 0.99$ and computed their fitness value and stopping iteration. Figure 14 shows the optimization solution with respect to the number of iterations when the corresponding decaying rate is used. We can see that generally, higher decaying rates cause more computational time while bringing about better solutions. Therefore, a balance has to be kept between these two aspects. Based on our observations, we have selected $\gamma = 0.95$ which results in a good enough time efficiency while obtaining satisfactory solutions.

To recap, the reported fitness values and iterations in the performance tables, as well as convergence curves, show that adaptive exploration/exploitation can improve the GWO performance for both unimodal and multimodal functions. Due to the adaptive nature of AGWO, we expect

this performance boosting for other optimization problems like real-world applications too. The adaptive stopping criterion reduces the computational cost differently for unimodal and multimodal functions (Fig. 11). The number of iterations also depends on the value of ϵ. Higher ϵ results in more efficient optimization with the cost of losing fitness accuracy at some levels.

Also, the comparison of three different versions of AGWO in the tables and plots shows that while the extended version AGWO^4 is more successful in finding very accurate optima for unimodal functions, the AGWO $\epsilon = 0$ has a better balance for both unimodal and multimodal functions, with more illustrated boosted performance in complex multimodal functions.

6 Conclusion

In this work, we proposed AGWO and AGWO^4, which are adaptive versions of the grey wolf optimizer. We showed that the proposed improvements were able to address some inherent limitations of GWO such as the non-adaptive balance between exploration and exploitation. The adaptive algorithms can adjust the control parameters with respect to the complexity of the problem by evaluating the fitness history in an online manner over the course of time. The algorithm has fitness-based convergence criteria that allow efficient optimization by introducing a significance threshold at which we can stop the optimization to avoid the waste of computational time for non-significant reductions in the function values. The ideas used for AGWO and AGWO^4 are novel in terms of automatic adjustments; however, the update equations and adaptive mapping can be improved and optimized for better results which can be an interesting line of research for future works.

Funding This work is supported by the start-up fund provided by CMU Mechanical Engineering, USA, and funding from National Science Foundation (CBET–1953222), United States.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Code availability The code of the algorithm can be accessed from: https://github.com/BaratiLab/Adaptive-Grey-Wolf-Optimization-Algorithm-AGWO.

References

1. Soerensen JS, Johannesen L, Grove U, Lundhus K, Couderc JP, Graff C (2010) A comparison of IIR and wavelet filtering for noise reduction of the ECG. In: 2010 computing in cardiology (IEEE, 2010), pp. 489–492
2. Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Chapter 10 - metaheuristic algorithms: a comprehensive review. In: Sangaiah AK, Sheng M, Zhang Z (eds) Computational intelligence for multimedia big data on the cloud with engineering applications. Intelligent data-centric systems. Academic Press, London, pp 185–231
3. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, London
4. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: proceedings of ICNN'95-international conference on neural networks, vol. 4 (IEEE, 1995), vol. 4, pp. 1942–1948
5. Storn R, Price K (1997) Differential evolution - a simple and efficient Heuristic for global optimization over continuous spaces. J Global Optim 11(4):341. https://doi.org/10.1023/A:1008202821328
6. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67
7. Glover F (1989) Tabu search-part I. ORSA J Comput 1(3):190. https://doi.org/10.1287/ijoc.1.3.190
8. Glover F, Kochenberger G (2002) Iterated local search. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. Kluwer, Netherlands
9. Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford, UK
10. Simon D (2008) Biogeography-based optimization. IEEE Trans Evolut Comput 12(6):702. https://doi.org/10.1109/TEVC.2008.919004
11. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232
12. Erol OK, Eksin I (2006) A new optimization method: big bang-big crunch. Adv Eng Softw 37(2):106. https://doi.org/10.1016/j.advengsoft.2005.04.005
13. Dorigo M, Birattari M, Stützle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28
14. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459
15. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: proceedings of the sixth international symposium on micro machine and human science, pp. 39–43
16. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: 2009 world congress on nature & biologically inspired computing, NaBIC 2009. (IEEE, 2009), pp. 210–214
17. Passino K (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52. https://doi.org/10.1109/MCS.2002.1004010
18. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46
19. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51
20. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228
21. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053. https://doi.org/10.1007/s00521-015-1920-1
22. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30. https://doi.org/10.1016/j.advengsoft.2017.01.004

23. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw. https://doi.org/10.1016/j.advengsoft.2015.01.010
24. Tu Q, Chen X, Liu X (2019) Hierarchy strengthened grey wolf optimizer for numerical optimization and feature selection. IEEE Access 7:78012. https://doi.org/10.1109/ACCESS.2019.2921793
25. Heidari AA, Pahlavani P (2017) An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. Appl Soft Comput 60:115. https://doi.org/10.1016/j.asoc.2017.06.044
26. Lu C, Gao L, Yi J (2018) Grey wolf optimizer with cellular topological structure. Exp Syst Appl 107:89. https://doi.org/10.1016/j.eswa.2018.04.012
27. Negi G, Kumar A, Pant S, Ram M (2021) GWO: a review and applications. Int J Syst Assur Eng Manag 12(1):1. https://doi.org/10.1007/s13198-020-00995-8
28. Faris H, Aljarah I, Al-Betar MA, Mirjalili S (2018) Grey wolf optimizer: a review of recent variants and applications. Neural Comput Appl 30(2):413. https://doi.org/10.1007/s00521-017-3272-5
29. Malik MRS, Mohideen ER, Ali L (2015) Weighted distance grey wolf optimizer for global optimization problems. In: 2015 IEEE international conference on computational intelligence and computing research (ICCIC)
30. Mittal N, Singh U, Sohi BS (2016) Modified grey wolf optimizer for global engineering optimization. Appl Comp Intell Soft Comput. https://doi.org/10.1155/2016/7950348
31. Long W, Liang X, Cai S, Jiao J, Zhang W (2017) A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems. Neural Comput Appl 28(1):421. https://doi.org/10.1007/s00521-016-2357-x
32. Rodríguez L, Castillo O, Soria J (2016) Grey wolf optimizer with dynamic adaptation of parameters using fuzzy logic. In: 2016 IEEE congress on evolutionary computation (CEC), pp. 3116–3123. https://doi.org/10.1109/CEC.2016.7744183
33. RodrÍguez L, Castillo O, Soria J, Melin P, Valdez F, Gonzalez CI, Martinez GE, Soto J (2017) A fuzzy hierarchical operator in the grey wolf optimizer algorithm. Appl Soft Comput 57:315. https://doi.org/10.1016/j.asoc.2017.03.048
34. Dudani K, Chudasama A (2016) Partial discharge detection in transformer using adaptive grey wolf optimizer based acoustic emission technique. Cogent Eng 3(1):1256083. https://doi.org/10.1080/23311916.2016.1256083
35. Long W, Jiao J, Liang X, Tang M (2018) An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. Eng Appl Artif Intell 68:63. https://doi.org/10.1016/j.engappai.2017.10.024
36. Long W, Jiao J, Liang X, Cai S, Xu M (2019) A random opposition-based learning grey wolf optimizer. IEEE Access 7:113810. https://doi.org/10.1109/ACCESS.2019.2934994
37. Sharma S, Salgotra R, Singh U (2017) An enhanced grey wolf optimizer for numerical optimization. In: 2017 international conference on innovations in information, embedded and communication systems (ICIIECS)
38. Nadimi-Shahraki MH, Taghian S, Mirjalili S (2021) An improved grey wolf optimizer for solving engineering problems. Exp Syst Appl 166:113907. https://doi.org/10.1016/j.eswa.2020.113917
39. Zhang S, Zhou Y (2015) Grey wolf optimizer based on powell local optimization method for clustering analysis. Discrete Dyn Nat Soc. https://doi.org/10.1155/2015/481360
40. Mahdad B, Srairi K (2015) Blackout risk prevention in a smart grid based flexible optimal strategy using Grey Wolf-pattern search algorithms. Energy Convers Manag 98:411. https://doi.org/10.1016/j.enconman.2015.04.005
41. Saremi S, Mirjalili SZ, Mirjalili SM (2015) Evolutionary population dynamics and grey wolf optimizer. Neural Comput Appl 26(5):1257. https://doi.org/10.1007/s00521-014-1806-7
42. Jayabarathi T, Raghunathan T, Adarsh B, Suganthan PN (2016) Economic dispatch using hybrid grey wolf optimizer. Energy 111:630. https://doi.org/10.1016/j.energy.2016.05.105
43. Wang JS, Li SX (2019) An improved grey wolf optimizer based on differential evolution and elimination mechanism. Sci Rep 9(1):7181. https://doi.org/10.1038/s41598-019-43546-3
44. Guha D, Roy PK, Banerjee S (2016) Load frequency control of large scale power system using quasi-oppositional grey wolf optimization algorithm. Eng Sci Technol Int J 19(4):1693. https://doi.org/10.1016/j.jestch.2016.07.004
45. Gaidhane PJ, Nigam MJ (2018) A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems. J Comput Sci 27:284. https://doi.org/10.1016/j.jocs.2018.06.008
46. Alomoush AA, Alsewari AA, Alamri HS, Aloufi K, Zamli KZ (2019) Hybrid harmony search algorithm with grey wolf optimizer and modified opposition-based learning. IEEE Access 7:68764. https://doi.org/10.1109/ACCESS.2019.2917803
47. Aleti A, Moser I (2016) A systematic literature review of adaptive parameter control methods for evolutionary algorithms. ACM Comput Surv. https://doi.org/10.1145/2996355
48. Zhan ZH, Zhang J, Li Y, Chung HSH (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern Part B (Cybernetics) 39(6):1362. https://doi.org/10.1109/TSMCB.2009.2015956
49. Naik MK, Panda R (2016) A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition. Appl Soft Comput 38:661. https://doi.org/10.1016/j.asoc.2015.10.039
50. You K, Long M, Wang J, Jordan MI (2019) How does learning rate decay help modern neural networks?
51. Yan F, Xu J, Yun K (2019) Dynamically dimensioned search grey wolf optimizer based on positional interaction information. Complexity. https://doi.org/10.1155/2019/7189653
52. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
53. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization
54. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: proceedings of the 30th international conference on international conference on machine learning - Vol 28 (JMLR.org, 2013), ICML'13, p. III-1139-III-1147
55. Digalakis J, Margaritis K (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77(4):481. https://doi.org/10.1080/00207160108805080
56. Qais MH, Hasanien HM, Alghuwainem S (2018) Augmented grey wolf optimizer for grid-connected PMSG-based wind energy conversion systems. Appl Soft Comput 69:504. https://doi.org/10.1016/j.asoc.2018.05.006
57. Zielinski K, Peters D, Laur R (2005) Stopping criteria for single-objective optimization
58. Zielinski K, Laur R (2007) Stopping criteria for a constrained single-objective particle swarm optimization algorithm. Informatica (Slovenia) 31:51
59. Fernández-Vargas JA, Bonilla-Petriciolet A, Rangaiah GP, Fateen SEK (2016) Performance analysis of stopping criteria of population-based metaheuristics for global optimization in phase equilibrium calculations and modeling. Fluid Phase Equilibria 427:104. https://doi.org/10.1016/j.fluid.2016.06.037