Applied Soft Computing Journal 83 (2019) 105658

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Effects of dominant wolves in grey wolf optimization algorithm

Fehmi Burcin Ozsoydan

Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, İzmir, Turkey

HIGHLIGHTS

- Learning hierarchy curves GWO.
- Prioritizing strategy for GWO.
- Comprehensive experimental study.
- Detailed statistical tests for demonstrations.
- Significant improvements over the published results.

ARTICLE INFO

Article history:
Received 4 September 2018
Received in revised form 17 July 2019
Accepted 21 July 2019
Available online 29 July 2019

Keywords:
Grey wolf optimization algorithm
Unconstrained optimization
Constrained optimization
Binary optimization
Particle swarm optimization

ABSTRACT

Bio-inspired computation is one of the emerging soft computing techniques of the past decade. Although they do not guarantee optimality, the underlying reasons that make such algorithms become popular are indeed simplicity in implementation and being open to various improvements. Grey Wolf Optimizer (GWO), which derives inspiration from the hierarchical order and hunting behaviours of grey wolves in nature, is one of the new generation bio-inspired metaheuristics. GWO is first introduced to solve global optimization and mechanical design problems. Next, it has been applied to a variety of problems. As reported in numerous publications, GWO is shown to be a promising algorithm, however, the effects of characteristic mechanisms of GWO on solution quality has not been sufficiently discussed in the related literature. Accordingly, the present study analyses the effects of dominant wolves, which clearly have crucial effects on search capability of GWO and introduces new extensions, which are based on the variations of dominant wolves. In the first extension, three dominant wolves in GWO are evaluated first. Thus, an implicit local search without an additional computational cost is conducted at the beginning of each iteration. Only after repositioning of wolf council of higher-ranks, the rest of the pack is allowed to reposition. Secondarily, dominant wolves are exposed to learning curves so that the hierarchy amongst the leading wolves is established throughout generations. In the final modification, the procedures of the previous extensions are adopted simultaneously. The performances of all developed algorithms are tested on both constrained and unconstrained optimization problems including combinatorial problems such as uncapacitated facility location problem and 0-1 knapsack problem, which have numerous possible real-life applications. The proposed modifications are compared to the standard GWO, some other metaheuristic algorithms taken from the literature and Particle Swarm Optimization, which can be considered as a fundamental algorithm commonly employed in comparative studies. Finally, proposed algorithms are implemented on real-life cases of which the data are taken from the related publications. Statistically verified results point out significant improvements achieved by proposed modifications. In this regard, the results of the present study demonstrate that the dominant wolves have crucial effects on the performance of GWO.

1. Introduction

Bio-inspired computational techniques bring about new opportunities in problem solving. They can be considered as the emerging soft computing techniques of the past decade. Although they do not guarantee optimality, they have become popular due to two main reasons. First, they are quite simple in implementation, which means that by making use of appropriate solution representation techniques, they can easily be adapted to various problem domains. Secondarily, they can achieve near-optimal results and they are still open to numerous possible improvements. Due to space limitation, only some of the commonly known bio-inspired algorithms are discussed in the following.

E-mail address: burcin.ozsoydan@deu.edu.tr.

https://doi.org/10.1016/j.asoc.2019.105658
1568-4946/

It is clear that Genetic Algorithm (GA) [1] is one of the pioneering algorithms in this domain. Because it adopts the evolutionary mechanisms of the Darwinian Evolutionary Theory, it is a representative example of such algorithms. Particle Swarm Optimization (PSO) [2] is another commonly used algorithm that derives inspiration from the movement characteristics of bird or fish flocks that seek for food. Differential Evolution (DE) Algorithm [3] can be considered as another fundamental algorithm, which uses a differentiation-based evolving strategy along with some evolutionary mechanisms of GA.

There are also several other recently introduced bio-inspired algorithms. In some of them, movement and survival behaviours of the artificial bees [4], fireflies [5], cuckoos [6], glowworms [7], bats [8], wolves [9], flowers [10] and grey wolves [11] are adopted in a simulated environment to solve various optimization problems. Although they adopt different strategies, it can be put forward that nature is the source of inspiration for such algorithms.

One of the relatively more recent bio-inspired algorithms, namely, Grey Wolf Optimization (GWO) Algorithm [11], which is inspired by the grey wolves, is shown to be a promising algorithm. It simulates the leadership hierarchy and hunting mechanisms of grey wolves. In nature, dominant wolves lead wolf packs. Therefore, it can put forward that they have crucial effects on hunting, reproducing and survival mechanisms. In parallel with this, the efficiency of GWO should be related to the behaviours of those artificial dominant wolves in the simulated environment. Although most of the metaheuristic algorithms, including GWO have similarities among each other, this feature of GWO can be considered as the distinctive mechanism. In this respect, analysing the effects of dominant wolves on solution quality, which surprisingly has not been discussed sufficiently in the related literature, deserves further research. This constitutes the main motivation of the present study and hence GWO is chosen as the main algorithm to analyse.

In this regard, three new extensions for GWO, particularly based on some modifications of leading wolves, are introduced to analyse the effects of dominant wolves on the efficiency of GWO. In the former extension, three dominant wolves (α, β and δ wolves) are repositioned first by sharing information with each other. In other words, every generation starts with movement procedures of these wolves according to their hierarchical order. Next, the rest of the wolves are allowed to reposition according to these dominant wolves. In the second extension, dominant wolves undergo some specific learning curves so that the behaviours of these wolves change throughout generations. That is to say, while they have equivalent influence on the wolf pack at earlier iterations, the α wolf, which indeed outperforms β and δ wolves, becomes more dominant to some extent, while β and δ lose their influence over generations. In this strategy, by increasing the influence of the α wolf, population is allowed to intensify around the best-found solution throughout generations. Additionally, it is also clear that, by assigning them equivalent weights at earlier iterations, local optima and premature convergence problems are aimed to be avoided. In the final modification, the procedures of the previous extensions are adopted simultaneously.

Performances of all developed algorithms are tested on various benchmarking problems, comprised of both constrained and unconstrained optimization problems including combinatorial problems such as uncapacitated facility location problem and 0-1 knapsack problem, which have numerous real-life applications. The proposed modifications are compared to the standard GWO, some other metaheuristic algorithms taken from the literature and Particle Swarm Optimization, which can be considered as a fundamental algorithm, commonly employed in comparative studies. Finally, appropriate statistical tests are conducted to determine possible significant improvements over the standard GWO.

The rest of this paper is organized as follows: related literature survey is presented in Section 2, while the proposed GWO modifications are introduced in Section 3. Computational results and concluding remarks are given in Section 4 and Section 5, respectively.

2. Literature review

As reported in this section, there is a wide range of reported publications related to GWO. They are chronologically organized as presented in the following.

A multi population-based GWO is proposed by Pan et al. [12]. According to the proposed method, gathered information is shared amongst the subpopulations by exchanging of the best solutions of subpopulations. Saremi et al. [13] present a modification of GWO, where poor individuals are reinitialized around the dominant wolves. Results reported by the authors point out efficiency of the proposed approach. In another study, GWO is employed by Mahdad and Srairi [14] in order to prevent blackout risk in a smart grid based flexible optimal strategy. Komaki and Kayvanfar [15] present a GWO modification, enhanced by a local search procedure to solve a two-stage assembly flow shop scheduling problem, which has variuous industrial applications. Song et al. [16] compare GWO with some other well-known stochastic search algorithms including GA and PSO in parameter estimation of surface waves. In regard to the obtained results, the authors strongly recommend the use GWO for the mentioned problem. Emary et al. [17] introduce a multi-objective GWO to solve attribute reduction problem, which is closely related to data mining and machine learning. Gholizadeh [18] proposes an extension for GWO by sequentially evaluating the individuals of a swarm, where each evaluation actually implements a standalone GWO. The mentioned approach is used to solve optimal design of double layer grids considering nonlinear behaviour.

Emary et al. [19] propose a binary modification for GWO to solve feature selection problem. The authors employ different types of sigmoid function while converting real values to binary variables in solution representation. Jayabarathi et al. [20] introduce another extension for GWO by including some evolutionary operators such as crossover and mutation. The proposed approach is used to solve various economic dispatch problems, which are known to be nonconvex and nonlinear in their nature. A similar approach excluding those evolutionary operators, is reported by Pradhan et al. [21] for a similar problem. Jayakumar et al. [22] propose a GWO application for combined heat and power dispatch, which is an important task in power system operations. Due to the problem characteristics, the authors additionally employ some constraint handling mechanisms. Unmanned combat aerial vehicle path planning problem, which might become seriously challenging particularly with the consideration of battlefield limitations, is solved by Zhang et al. [23] by using GWO. Guha et al. [24,25] modify GWO by making use of an extension with quasi-oppositional based mechanism to solve load frequency control problem in an inter-connected power system network equipped with classical PI/PID controller. Another multi-objective modification of GWO enhanced by an external archive is proposed by Mirjalili et al. [26]. The authors compare the proposed GWO with two other well-known metaheuristics, including a PSO variant. Results point out the efficiency of GWO also in multi-objective problems. A GWO-based recommender system, which is related to data mining and big data research fields, is proposed by Katarya and Verma [27]. Non-convex economic

load dispatch problem is solved by Kamboj et al. [28]. Shakarami and Davoudkhani [29] introduce another GWO implementation for wide-area power system stabilizer design, which is further extended as a multi-objective problem.

In the following year, Rodríguez et al. [30] propose several modifications on the hierarchical GWO hunting process. Motivation of the authors is indeed changing the weights of α, β and δ wolves while generating new solutions. The proposed approach achieves promising results in function optimization problems. Heidari and Pahlavani [31] report a modified GWO algorithm that employs Lévy flight and greedy selection strategies to solve global optimization problems. Tawhid and Ali [32] hybridize GWO by GA in order to establish a balance between the exploration and exploitation. Furthermore, the authors divide population into subpopulations to monitor diverse parts of the corresponding problem space. Another hybrid multi-objective GWO is introduced by Lu et al. [33] for a real-world industrial application including a dynamic scheduling problem. Long et al. [34] solve the well-known constrained optimization problems by using GWO. Authors employ Lagrangian function as the constraint handling method. Rodríguez et al. [35] propose tuning the search parameters of GWO by employing a fuzzy-based procedure for unconstrained function minimization problems. Precup et al. [36] employ a GWO for fuzzy control systems with a reduced parametric sensitivity. Another multi population-based GWO is employed by Yang et al. [37] to solve the problem of for maximum power point tracking of doubly-fed induction generator based wind turbine. The authors divide whole population into two sub-populations, where the former one includes an additional type of wolf for exploitation, and the latter one is used as a scout group for exploration. Khairuzzaman and Chaudhury [38] report another GWO implementation for multilevel thresholding in image segmentation research field. In another novel study, Farshin and Sharifian [39] employ a chaotic grey wolf controller allocator for dynamic controller allocation for the 5th generation of mobile technology (5G).

Long et al. [40] perform modifications on the exploration procedure of GWO by making use of nonlinear step sizing functions with different attributes. Another binary modification of GWO is presented by Panwar et al. [41] to solve unit commitment problem, which is a large scaled and constrained optimization problem involving operational planning of power system generation assets. Ibrahim et al. [42] propose a chaotic map embedded GWO to solve function optimization problems. The authors employ further enhancements including opposition-based search and disruption operator for diversification. Saxena et al. [43] propose a new extension for GWO referred to as intelligent GWO by employing two mathematical frameworks, including an efficient sinusoidal truncated function and oppositional-based learning strategy. Khandelwal et al. [44] propose another modification for GWO by appending a new term to the hunting behaviour, which indeed takes the positions of other wolves into account. Another hybridized GWO to solve constrained optimization design optimization problems is reported by Panagant and Bureerat [45]. Khalilpourazari and Khalilpourazary [46] introduce a modification for GWO, of which the parameters are tuned by using experimental design techniques. The proposed approach is tested on a real-world application. Fahad et al. [47] introduce a GWO-based clustering algorithm for vehicular ad-hoc networks. Long et al. [48] propose another modification of GWO by adopting two additional mechanisms, namely, a nonlinear adjustment strategy and a modified position-updating based on the individual best position and the global best positions. A selection procedure to determine the dominant wolves in GWO is reported by Al-Betar MA [49]. In a recent work, Gupta and Deep [50] propose a GWO that is enhanced by random walks to exhibit a more diversified search. The proposed algorithm is tested on a comprehensive benchmark suite and is compared to the state-of-the-art methods. Results point out the efficiency of this new modification. Qin et al. [51] report a hybridized and discrete multi-objective GWO to solve the casting production scheduling problem. Saxena and Kumar [52] emphasize the balance between exploration and exploitation, which are two important factors for an optimization algorithm. In this regard, the authors further enhance GWO by adopting β-chaotic sequence for better exploration and exploitation to solve global optimization problems. Finally, Luo [53] proposes a more realistic and innovative GWO modification, where the location of promising regions is dynamically estimated by dominant wolves. The reported results demonstrate the efficiency of this new model in global optimization problems.

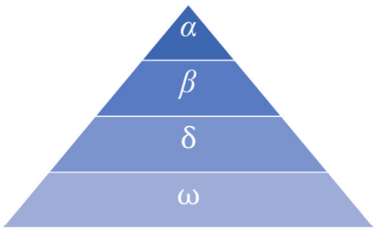

Fig. 1. Hierarchical order of grey wolves in nature [11].

3. Solution approaches

This section is devoted to the used solution approaches in the present study. For a better understanding, the standard version of GWO is presented first.

3.1. The standard GWO

GWO is first introduced by Mirjalili et al. [11] to solve both constrained and unconstrained optimization problems. It derives inspiration from the social hierarchy, tracking, encircling, and attacking prey for grey wolves, which actually belong to Canidae family in nature.

The hierarchy amongst a wolf pack is depicted by Fig. 1. According to this figure, there are four types of wolves with different tasks and attributes. The most dominant wolf is the α wolf that commands all subordinates. All other wolves acknowledge the α wolf. The second rank in Fig. 1 belongs to the β wolf that also acknowledges α wolf but can command the others. The lowest ranking grey wolf is known to be the ω wolves [11]. Although they might seem unimportant individuals, they are required for the sake of the wolf pack in order to prevent violence among the dominant wolves.

Mirjalili et al. [11] model this hierarchical order of grey wolves by considering the best, the second best and the third best solutions of a population as α, β and δ wolves, respectively. The rest of a population can be considered as ω wolves. Thus, the dominant wolves can lead the rest of a population.

Grey wolves first encircle a prey during the hunt. The encircling prey behaviour is modelled by the formulations given by Eqs. (1)–(2), where t, $\vec{X}_p(t)$, $\vec{X}(t)$, $\vec{A}$, $\vec{C}$ and $\vec{a}$ represent the iteration index, coordinates of the prey, coordinates of the related wolf and movement coefficients, respectively. Coefficients $\vec{A}$ and $\vec{C}$, which favour divergence of wolves, are calculated in regard to Eqs. (3)–(4), where $\vec{r}_1$ and $\vec{r}_2$ are uniformly distributed random vectors $\in (0, 1)$. Additionally, in order to allow intensification

towards the end of the search, the elements of $\vec{a}$ are linearly decreased from 2 to 0 throughout iterations.

$$\vec{D} = \left|\vec{C} \;.\; \vec{X}_p\,(t) - \vec{X}\,(t)\right| \tag{1}$$

$$\vec{X}\,(t+1) = \vec{X}_p\,(t) - \vec{A} \;.\; \vec{D} \tag{2}$$

$$\vec{A} = 2\vec{a} \;.\; \vec{r}_1 - \vec{a} \tag{3}$$

$$\vec{C} = 2 \;.\; \vec{r}_2 \tag{4}$$

Finally, the hunting behaviour follows the encircling prey behaviour. As presented above, three dominant wolves (α, β and δ) lead the rest of the population that is comprised of the lowest ranks of wolves denoted by ω. In parallel with this mechanism, Mirjalili et al. [11] propose the following formulations to model this hunting behaviour. Putting things together, a pseudo code for the standard GWO is presented in Algorithm 1.

$$\begin{aligned}\vec{D}_\alpha &= \left|\vec{C}_1 \;.\; \vec{X}_\alpha - \vec{X}\right| \\ \vec{D}_\beta &= \left|\vec{C}_2 \;.\; \vec{X}_\beta - \vec{X}\right| \\ \vec{D}_\delta &= \left|\vec{C}_3 \;.\; \vec{X}_\delta - \vec{X}\right|\end{aligned} \tag{5}$$

$$\begin{aligned}\vec{X}_1 &= \vec{X}_\alpha - \vec{A}_1 \;.\; \vec{D}_\alpha \\ \vec{X}_2 &= \vec{X}_\beta - \vec{A}_2 \;.\; \vec{D}_\beta \\ \vec{X}_3 &= \vec{X}_\delta - \vec{A}_3 \;.\; \vec{D}_\delta\end{aligned} \tag{6}$$

$$\vec{X}\,(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{7}$$

Algorithm 1. A pseudo code for the standard GWO.

initialize GWO population of n solution vectors
define $\vec{A}$, $\vec{C}$ and $\vec{a}$
evaluate fitness values $f(x_i)$, $i = 1, \ldots, popSize$
find $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
while $(t < maxGen)$
    for $i = 1{:}\,popSize$
        update all positions according to Eq. 7
    end for
    update $\vec{A}$, $\vec{C}$and $\vec{a}$
    evaluate fitness values $f(x_i)$, $i = 1, \ldots, n$
    update $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
    $t = t + 1$
end while
print $(\vec{X}_\alpha)$
// $maxGen$ is the maximum number of generations
// $popSize$ is the population size

3.2. Proposed modifications

As mentioned in the previous subsection, the α, β and δ wolves lead the swarm, therefore, their knowledge about promising locations become crucial while hunting. In order to observe the effects of dominant wolves on the efficiency of GWO, some new extensions, only based on some modifications of dominant wolves, are introduced in this section.

3.2.1. Prioritizing dominant wolves

In this modification, the three dominant wolves first share information before leading the ω. That is to say, they first rearrange their own positions according to the locations of each other. According to the hierarchical order, the α is given the first priority. After α tunes its own coordinates, then β relocates according to α and δ. In what follows, α and β command δ for relocation. Finally, the rest of the wolves (ω) are allowed to reposition according to the coordinates of dominant wolves.

More formally, at the beginning of each iteration, the α wolf is moved first. In other words, Eqs. (5)–(7) are used for α first. Thus, the most dominant wolf takes a new position. Next, the same procedure is performed for β and δ in sequence. Finally, these wolves are allowed to lead the rest of the population by using their new locations.

This can be considered as a procedure, where the leading three solutions first perform an implicit local search around each other, then the rest of the population moves according to these new positions. Therefore, this extension indeed provides an implicit local search that does not consume any additional fitness function evaluations. Because leading wolves are prioritized here, this extension is referred to as *prio*GWO throughout the rest of the paper. A pseudo code for *prio*GWO is presented by Algorithm 2.

Algorithm 2. A pseudo code for *prio*GWO.

initialize GWO population of n solution vectors
define $\vec{A}$, $\vec{C}$ and $\vec{a}$
evaluate fitness values $f(x_i)$, $i = 1, \ldots, popSize$
find $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
while $(t < maxGen)$
    move $\vec{X}_\alpha$ *according to* Eq. 7
    move $\vec{X}_\beta$ *according to* Eq. 7
    move $\vec{X}_\delta$ *according to* Eq. 7
    for $i = 1{:}\,(popSize - 3)$
        update positions of $\vec{X}_\omega$ *according to* Eq. 7
    end for
    update $\vec{A}$, $\vec{C}$and $\vec{a}$
    evaluate fitness values $f(x_i)$, $i = 1, \ldots, n$
    update $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
    $t = t + 1$
end while
print $(\vec{X}_\alpha)$
// $maxGen$ is the maximum number of generations
// $popSize$ is the population size

3.2.2. Defining level of dominance amongst the dominant wolves

As the second modification proposed in the present study, dominant wolves undergo some learning curves so that the hierarchical order and level of dominance amongst the dominant wolves are established throughout generations. It is worth stressing that in the standard GWO, all dominant wolves have equal influences on the population (Eq. (7)). Because the wolves learn their hierarchical influence over generations, this modification is referred to as *learn*GWO in the following.

In parallel with the standard GWO, *learn*GWO assigns equal weights to dominant wolves at the initialization stage. In what follows, the α wolf, which indeed outperforms β and δ wolves, becomes more dominant to some extent, while β and δ lose their influences over generations. In order to model this behaviour, the formulations presented by Eqs. (8)–(9) [54] are used. In these equations $w_\alpha\,(t)$, $w_\beta\,(t)$ and $w_\delta\,(t)$ denote the weights of the α, β and δ wolves at the tth iteration, respectively. Additionally, the speed of change for these weights can be tuned by changing value of the parameter θ for the corresponding wolf.

$$w_\alpha\,(t+1) = w_\alpha\,(t) + e^{-t/(t+1)} \times \theta_\alpha \times w_\alpha\,(t)$$

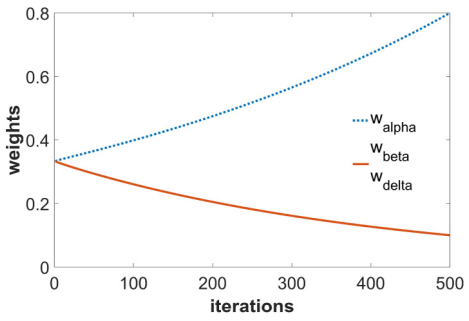

Fig. 2. Changing weights of wolves over iterations.

$$w_\beta(t+1) = w_\beta(t) - e^{-t/(t+1)} \times \theta_\beta \times w_\beta(t) \tag{8}$$

$$w_\delta(t+1) = w_\delta(t) - e^{-t/(t+1)} \times \theta_\delta \times w_\delta(t)$$

$$\vec{X}(t+1) = w\_n_\alpha(t+1) \cdot \vec{X}_1 + w\_n_\beta(t+1) \cdot \vec{X}_2 + w\_n_\delta(t+1) \cdot \vec{X}_3 \tag{9}$$

It is clear that the initial values of all weights are set to $w_\alpha(1) = w_\beta(1) = w_\delta(1) = 1/3$. Next, while the value of w_α is increased throughout generations, w_β and w_δ are decreased (Eq. (8)). Preliminary work shows that promising results can be obtained while $w_\alpha(maxGen) = 0.8$ and $w_\beta(maxGen) = w_\delta(maxGen) = 0.1$. It is clear that because the final and initial values of these weights are defined by the decision maker, one can evaluate the values of θ_α, θ_β and θ_δ for a given total number of iterations (*maxGen*). For example, if *maxGen* is fixed to 500, $\theta_\alpha = 0.004715$ and $\theta_\beta = \theta_\delta = 0.00647$ can be obtained accordingly. A visualization for this parameter combination is depicted in Fig. 2. Moreover, it is clear that sum of the weights might not be equal to unity in all iterations. For this reason, these values are normalized by using Eq. 10 to obtain the normalized weights at the tth iteration, which are denoted by $w\_n(t+1)$. It is worth stressing that the normalized values are not used while evaluating the next values of these weights. Normalized weights are used only in Eq. 9, while the non-normalized values are used to evaluate the upcoming values in Eq. (8).

$$\begin{aligned} w\_n_\alpha(t+1) &= w_\alpha(t+1) \Big/ \left[w_\alpha(t+1) + w_\beta(t+1) + w_\delta(t+1)\right] \\ w\_n_\beta(t+1) &= w_\beta(t+1) \Big/ \left[w_\alpha(t+1) + w_\beta(t+1) + w_\delta(t+1)\right] \\ w\_n_\delta(t+1) &= 1 - \left(w_\alpha(t+1) + w_\beta(t+1)\right) \end{aligned} \tag{10}$$

In this strategy, by increasing the influence of the α wolf, population is allowed to intensify around the best-found solution throughout generations. Moreover, by assigning them equal weights at earlier iterations, local optima and premature convergence are aimed to be avoided. A pseudo code for *learn*GWO is presented by Algorithm 3.

Algorithm 3. A pseudo code for *learn*GWO.

initialize GWO population of n solution vectors
define $\vec{A}$, $\vec{C}$ and $\vec{a}$
evaluate fitness values $f(x_i), i = 1, \ldots, popSize$
find $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
define $w_\alpha(1)$, $w_\beta(1)$ and $w_\delta(1)$
define $w_\alpha(maxGen)$, $w_\beta(maxGen)$ and $w_\delta(maxGen)$
evaluate θ_α, θ_β and θ_δ
while $(t < maxGen)$
    evaluate normalized values of weights (Eq. 10)
    for $i = 1: popSize$
        update positions according to Eq. 8
    end for
    update $\vec{A}$, $\vec{C}$and $\vec{a}$
    evaluate fitness values $f(x_i), i = 1, \ldots, n$
    update $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$
    $t = t + 1$
    evaluate $w_\alpha(t)$, $w_\beta(t)$ and $w_\delta(t)$ (Eq. 8)
end while
print $(\vec{X}_\alpha)$
// *maxGen* is the maximum number of generations
// *popSize* is the population size

It is clear that there are infinite number of parameter combinations for this learning strategy. Moreover, sum of the weights are not necessarily required to be equal to unity, because, normalization procedure (Eq. (10)) can handle this issue. In this regard, numerous patterns for *learn*GWO can be defined here. However, as mentioned above, the presented combination is found as promising in the preliminary work.

3.3. Particle swarm optimization

PSO [2] is one of the most widely used swarm-based algorithms to solve challenging optimization problems. It mimics the behaviours of bird or fish flocks that seek for food. In the movement procedure of PSO, current position (x_{ij}), particle best solution ($pbest$) and global best solution (X_{best}) have effects on particles' speed. This procedure is given by Eq. (11), where x_{ij}, v_{ij}, c_0, c_1 and c_2 represent position and speed of the ith particle for the jth dimension, the inertia coefficient and the accelerating coefficients, respectively. Finally, the next position for the related dimension of a particle is calculated based on the formulation given by Eq. (12). It should be stressed that the velocity of a particle is constrained subject to Eq. (13). If boundaries are violated, velocity of that particle is equalized to the closest boundary.

$$v_{ij} = c_0 v_{ij} + c_1 \times rand \times \left(pbest_{ij} - x_{ij}\right) + c_2 \times rand \times \left(x_{best,j} - x_{ij}\right) \tag{11}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{12}$$

$$v_{ij} \in [v_{min}, v_{max}] \tag{13}$$

As reported by Mirjalili and Lewis [55], a linear decreasing value for the inertia coefficient can yield to more promising results. Therefore, this strategy is also adopted in the present study. The rest of the details are presented in the experimental study. A pseudo code for the employed PSO is given by Algorithm 4.

Algorithm 4. A pseudo code for PSO.

```
initialize PSO population of n solution vectors
evaluate fitness values f(x_i), i = 1, ..., popSize
while (t < maxGen)
    for i = 1: popSize
        for j = 1: dim
            evaluate v_ij (Eq. 11)
            evaluate x_ij (Eq. 12)
        end for
    end for
    evaluate new fitness values
    update the global and particles best solutions
    update c_0
    t = t + 1
end while
print (X_best)
// maxGen is the maximum number of generations
// popSize is the population size
// dim is the dimension of the problem
```

4. Experimental results

4.1. Benchmark suite

In order to test the performances of the developed approaches and to observe the effects of the modified dominant wolves, three different benchmark suites including real-valued, combinatorial, unconstrained and constrained benchmarking problems are employed in the present study.

The first benchmark is comprised of the well-known real-valued unconstrained optimization functions, commonly used in the related literature. The details of all used functions and their landscape figures are presented in Appendix.

The second benchmarking environment is chosen as the uncapacitated facility location problem[1] (UFLP) that has attracted remarkable attention of researchers. UFLP is a well-known type of location problems [56,57], where each customer is served by exactly one facility. Given a set of locations for establishing facilities and customers at demand points, the aim in UFLP is to minimize the total cost that occurs from the facility opening costs and shipment costs. UFLP can be formulated as given in the following (Eqs. (14)–(18)).

$$min \quad \sum_{k\in K}\sum_{l\in L} c_{kl}z_{kl} + \sum_{k\in K} b_k y_k \tag{14}$$

$s.t.$

$$\sum_{k\in K} z_{kl} = 1, \qquad l \in L \tag{15}$$

$$z_{kl} \le y_k, \qquad k \in K \text{ and } l \in L \tag{16}$$

$$z_{kl} \in \{0,1\}, \qquad k \in K \text{ and } l \in L \tag{17}$$

$$y_k \in \{0,1\}, \qquad k \in K \tag{18}$$

where $K = \{1, 2, \ldots, n\}$ is the set of possible facility locations and $L = \{1, 2, \ldots, m\}$ is the set of demand points. In this context, z_{kl} is a binary variable denoting whether the demand of the lth demand point (customer) is fulfilled by the kth facility location ($z_{kl} = 1$ if the lth customer is served by the facility opened at the kth location, $z_{kl} = 0$ otherwise) and y_k is another binary variable representing the status of a facility at the kth location ($y_k = 1$ if a facility is opened at the kth location, $y_k = 0$ otherwise). The parameter b_k is the cost of establishing a facility at the kth location, c_{kl} is the shipment cost occurred between the facility opened at the kth location and the lth customer. Accordingly, Eq. (14) is the objective function that minimizes the total cost. Eq. (15) satisfies that each customer is served by only one facility. Eq. (16) provides that customers can only be served by the opened facilities. Finally, Eqs. (17) and (18) impose restrictions on decision variables.

UFLP, which has various real-life applications such as finding locations of facilities on a network, cluster analysis, machine scheduling, economic lot sizing, portfolio management, computer network design, is shown to be an NP-hard problem [56]. Therefore, solution methods consisting mostly of heuristic approaches are proposed in this domain. Due to space limitation, all related publications cannot be presented here, however, readers might refer to the surveys of Hale and Moberg [58] and Şahin and Süral [59].

The final benchmark suite is chosen as the 0-1 knapsack problem (KP) which has numerous practical real-life applications such as cargo loading, project funding selection, budget management, cutting stock, etc., [60]. Given a set of items with profits and weights, the aim of 0-1 KP is to maximize the knapsack profit without violating the knapsack capacity. This problem can be formulated as given in the following (Eqs. (19)–(21)).

$$max \quad \sum_{r\in R} p_r x_r \tag{19}$$

$s.t.$

$$\sum_{r\in R} w_r x_r \le C \tag{20}$$

$$x_r \in \{0,1\}, \qquad r \in R \tag{21}$$

where $R = \{1, 2, \ldots, z\}$ is the set of items, p_r represents the profit the rth item, w_r is the weight (resource consumption) of the rth item, C is the capacity of the knapsack, x_r is a binary variable denoting whether the rth item is assigned ($x_r = 1$ if the rth item is assigned to knapsack, $x_r = 0$, otherwise). Accordingly, Eq. (19) is the objective function that maximizes the profit of the knapsack and Eq. (20) is the knapsack capacity constraint. Finally, Eq. (21) imposes restrictions on the decision variables.

0-1 KP is also shown to be an NP-hard problem [60]. Therefore, solution approaches consisting mostly of heuristic approaches are reported. Due to space limitation again, readers are asked to refer to the survey presented by Bhattacharjee and Sarmah [61].

4.2. Solution encoding and handling infeasibility

It is clear that continuous variables in real-valued optimization problems can directly be used throughout search. However, this is not possible to use them directly in UFLP. Therefore, in order to convert continuous variables to binary in UFLP, transfer functions formulated by Eqs. (22) and (23) are employed. This is one of the most commonly used methods that is referred to as sigmoid (logistic) transfer function [55]. In these equations $rand$, ϑ, $x_{i,j}$ and $bin_{i,j}$ denote uniform and continuous random number $\in (0, 1)$, coefficient in conversion, continuous value of the ith solution vector at the jth dimension and its corresponding binary value, respectively. To sum up, initial population is randomly generated as continuous variables between the boundaries of the related instance in real-valued global optimization problems (the first benchmark suite). If any boundary is violated throughout search, corresponding dimension is set to the value of the exceeded boundary. On the other hand, in UFLP initial population is randomly generated as uniform and continuous random numbers $\in (0, 1)$ and transfer functions are used while evaluating fitness

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html.

of a solution vector. Real-valued vectors in UFLP and 0-1 KP are not restricted to boundaries.

$$T\left(x_{i,j}\right) = \frac{1}{1 + e^{\vartheta \times x_{i,j}}} \tag{22}$$

$$bin_{i,j} = \begin{cases} 0 & \text{if } rand < T\left(x_{i,j}\right) \\ 1 & \text{if } rand \geq T\left(x_{i,j}\right) \end{cases} \tag{23}$$

It should be noted that according to the Eqs. (22) and (23), it is possible to obtain a zero vector of which all elements (bits) are comprised of zeros. It means that none of the facilities is opened in such solution vectors. In order to avoid this, one facility is opened at a random candidate location in such circumstances. Additionally, it should also be noted that customers are served by the closest opened facility. Therefore, any additional procedures to handle Eq. (15) is not required.

It is further clear that also the 0-1 KP is a binary problem. However, by making use of the *random key representation*, transfer function is not required in 0-1 KP. Moreover, constraints are implicitly handled by this solution representation technique. That is to say, as presented above, 0-1 KP also uses random variables in solution vectors, which means that they can also be considered as *random keys*. It is assumed in the present study that the *random key* with the greatest value (the item with the greatest real value in the solution vector) has the first priority to be assigned to knapsack unless it violates the capacity constraint. If the item is accepted, knapsack capacity is decreased by the weight of that item. Next, the item with the second greatest value is checked. If it violates the capacity, that item is ignored. The same procedure is followed for all items in the solution vector according to the sequence derived from the values in real-valued vector. Thus, transfer function and an explicit constraint handling technique are not required in 0-1 KP.

4.3. Fine-tuning of parameters

All tests are performed on a PC with i7-4790K CPU and 32 GB RAM. Maximum number of iterations *maxIter* is used as the termination criterion. In accordance with the literature, *maxIter* is fixed to 500 in function optimization benchmarks, whereas it used as 1000 in UFLP and 0-1 KP [62–64]. Similarly, population size (*popSize*) is fixed to 30, to the number of candidate facility locations (n) and to 50 in function optimization problems, in UFLP and in 0-1 KP, respectively. The rest of the GWO and PSO related parameters are adopted from Mirjalili et al. [11] and from Mirjalili and Lewis [55]. The parameters θ_α, θ_β, θ_δ are distinctly tuned for $maxIter = 500$ and $maxIter = 1000$ as mentioned in Section 3.2.2. Additionally, it should be noted that as proposed by Mirjalili et al. [11] the parameter $\vec{a}$ is linearly decreased from 2.0 to 0.0. Similarly, the inertia coefficient of PSO (c_0) is also linearly decreased from 0.9 to 0.2 to achieve better results [55]. All algorithm related parameters are brought together in Table 1.

Secondarily, the effect of the parameter ϑ is analysed. As already mentioned, transfer function is used only in UFLP. Therefore, a medium-scaled instance of UFLP (cap102) is used for this analysis. According to the results represented in Table 2, $\vartheta = 50$ gives the most promising results both for GWO and PSO. This value is also used in the other modifications of GWO.

4.4. Computational results for function optimization

Obtained results for function optimization benchmark suite are given in Table 3. Found best results in this table are highlighted. As one can see from the results presented in Table 3, *learn*GWO appears to be more promising than other algorithms in terms of finding the best solution. It means that the dominant wolves, which are exposed to learning curves, seem to be performing well enough to outperform the rest of the algorithms. Moreover, standard deviance values obtained by *learn*GWO is also mostly better than those of found by other algorithms'. It means that *learn*GWO is able to exhibit a more robust performance.

Additionally, it can be concluded that *learn*GWO is followed by *prio*GWO. It is clear that allowing dominant wolves to evolve first just before leading the rest of the population yields to a better performance in comparison to the standard GWO. As mentioned earlier, this is indeed an implicit local search that does not require additional fitness evaluations. Considering the performance of *prio*GWO, it can be concluded that this mechanism is promising for the given benchmark suite.

Although *prLe*GWO performs better than GWO and PSO, surprisingly, it is not found as efficient as *prio*GWO and *learn*GWO. One of the possible reasons might be loss of population diversity, which is preserved better in *prio*GWO and *learn*GWO. It is clear that simultaneous use of prioritizing procedure and learning curves might yield to unnecessarily too much intensification around the best-found solution. In other words, it can be considered that *prLe*GWO might be exposed to premature convergence. In order to observe whether such a circumstance exists, convergence plots of all algorithms for this benchmark suite are presented in Fig. 3.

As one can see from Fig. 3, where each data point represents the mean result over 30 replications, while GWO, PSO and *prio*GWO exhibit standard convergence behaviour, in some of the instances *learn*GWO and *prLe*GWO perform uncommon convergence pattern. It is clear that this pattern arises both from characteristics of the instances and from the procedures used by these algorithms. That is to say, *learn*GWO and *prLe*GWO seem to be being exposed to premature convergence in some of the instances. It is also clear from Fig. 3 that *prio*GWO has a smoother convergence behaviour in comparison to *learn*GWO and *prLe*GWO. Moreover, it performs better than GWO in most of the instances.

4.5. Computational results for UFLP

Obtained results over 30 independent replications for UFLP are presented in Table 4. The second row in this table denotes the size of the corresponding instance. The rows represented by *hit* show the number of runs where optimum solution is found. The maximum *hit* values for each of the instances are highlighted in Table 4.

It is clear from Table 4 that for small-scaled instances, all algorithms achieve similar results. However, in medium and large-scaled instances, the difference between the algorithms becomes more apparent. According to the results in Table 4, *prLe*GWO achieves the best performance regardless of the problem size. Moreover, it also outperforms the previously published results in the literature.

Distinctly from the previous benchmark suite, *prLe*GWO seems to be the most promising algorithm in UFLP. Additionally, it can be concluded that all GWO extensions perform better than the standard GWO, while *prio*GWO and *learn*GWO achieve similar results in most of the instances. Surprisingly, PSO has the worst performance particularly in medium and large-scaled benchmarks. The convergence plots of the mean results for UFLP are depicted in Fig. 4.

It is apparent from these convergence plots (Fig. 4) that GWO and its extensions exhibit a faster convergence in comparison to PSO. Moreover, it is also clear that near-optimal results are achieved by these algorithms in the first 100 iterations in most of the instances. Interesting results are apparent in medium and larger-scaled instances. *pr*LeGWO and *learn*GWO converge to near optimal results later than GWO and *prio*GWO do.

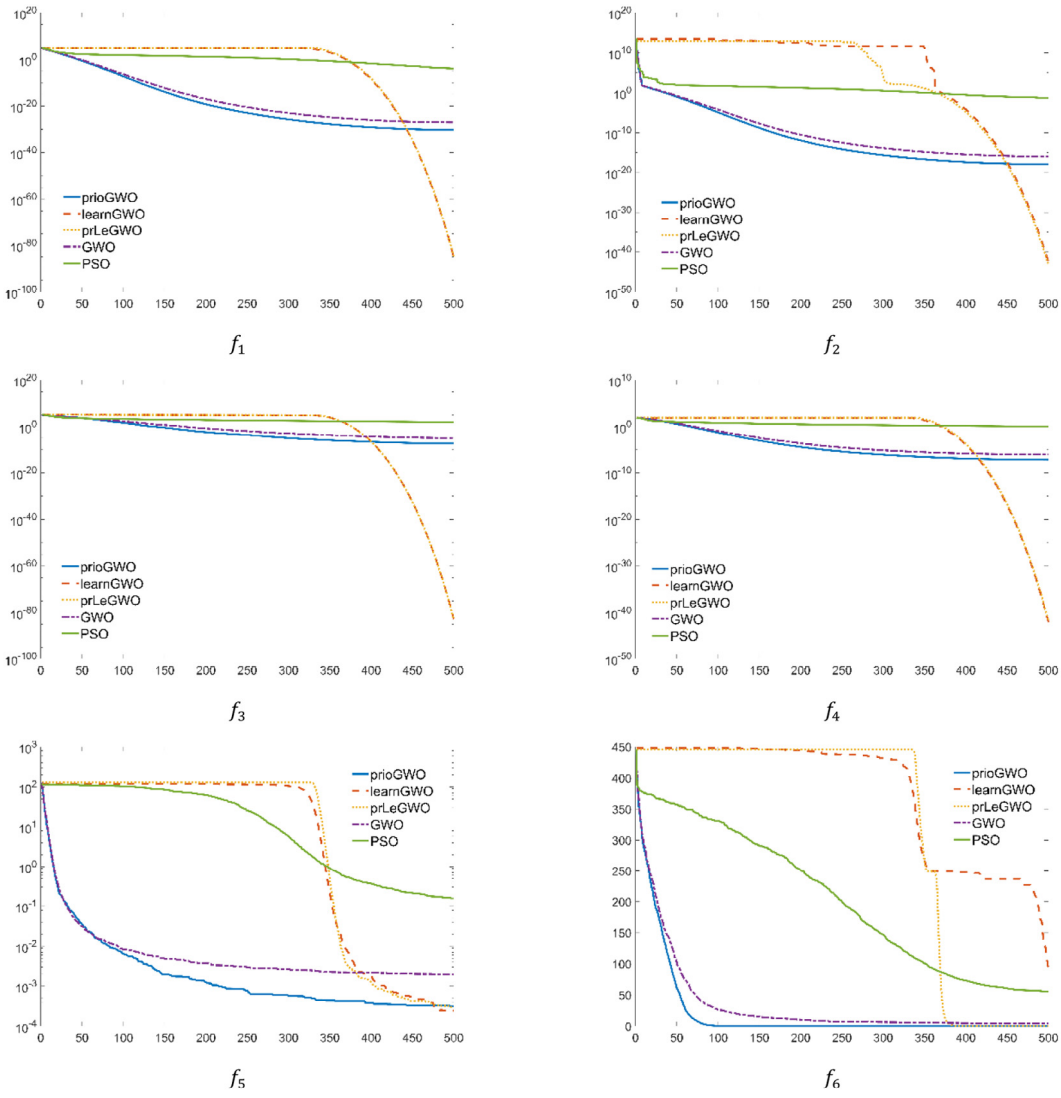

Fig. 3. Convergence graphs for unconstrained function optimization problems.

Table 1
Algorithm related parameters.

Parameters	*Algorithms*				
	GWO	*prio*GWO	*learn*GWO	*prLe*GWO	PSO
$\vec{a}$	[0.0, 2.0]	[0.0, 2.0]	[0.0, 2.0]	[0.0, 2.0]	*Na*
w_α (1)	*Na*	*Na*	1/3	1/3	*Na*
w_β (1)	*Na*	*Na*	1/3	1/3	*Na*
w_δ (1)	*Na*	*Na*	1/3	1/3	*Na*
w_α (*maxIter*)	*Na*	*Na*	0.80	0.80	*Na*
w_β (*maxIter*)	*Na*	*Na*	0.10	0.10	*Na*
w_δ (*maxIter*)	*Na*	*Na*	0.10	0.10	*Na*
θ_α [a]	*Na*	*Na*	0.004715	0.004715	*Na*
θ_β [a]	*Na*	*Na*	0.006470	0.006470	*Na*
θ_δ [a]	*Na*	*Na*	0.006470	0.006470	*Na*
θ_α [b]	*Na*	*Na*	0.002368	0.002368	*Na*
θ_β [b]	*Na*	*Na*	0.003246	0.003246	*Na*
θ_δ [b]	*Na*	*Na*	0.003246	0.003246	*Na*
c_0	*Na*	*Na*	*Na*	*Na*	[0.2, 0.9]
c_1	*Na*	*Na*	*Na*	*Na*	2.0
c_2	*Na*	*Na*	*Na*	*Na*	2.0
$[v_{min}, v_{max}]$	*Na*	*Na*	*Na*	*Na*	[−6.0, 6.0]
popSize	30	30	30	30	30

[a] θ_α, θ_β and θ_δ values for 500 iterations.
[b] θ_α, θ_β and θ_δ values for 1000 iterations.

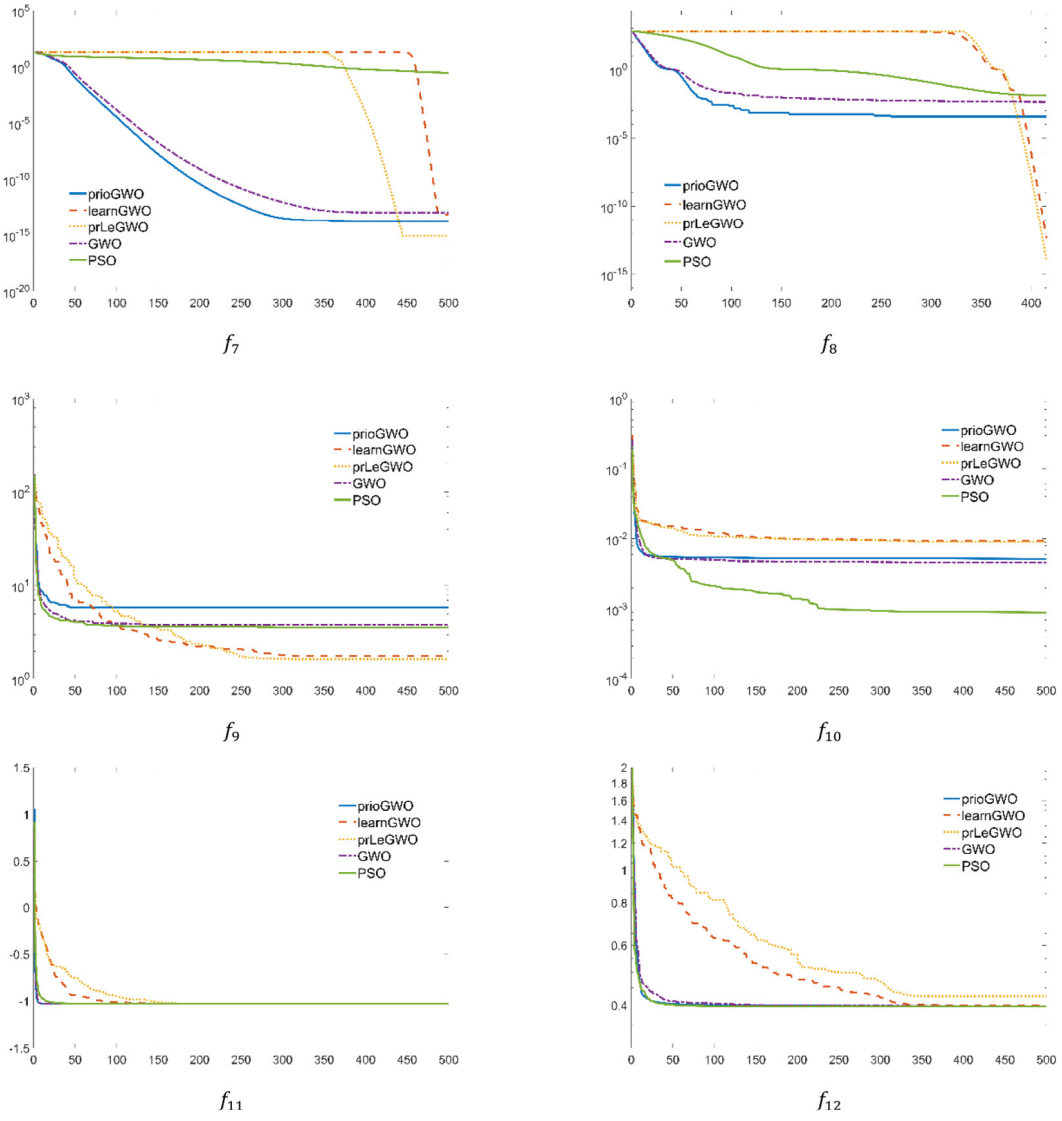

Fig. 3. (*continued*).

Table 2
The effects of ϑ in UFLP.

ϑ	GWO				PSO			
	Best	*Mean*	*Worst*	*Std.*	*Best*	*Mean*	*Worst*	*Std.*
0.50	856802.78	860434.88	865169.12	2624.35	863465.73	865105.62	868295.46	2762.84
1.00	856543.75	860134.87	865845.73	2610.56	857623.52	860568.62	863446.87	2912.25
1.50	854704.20	859594.89	864485.25	2537.17	858801.27	861363.38	863532.65	2390.02
2.00	855466.85	859218.63	867825.47	3058.72	857843.96	858323.32	858782.11	469.41
2.50	854704.20	857833.99	865275.68	2000.67	857286.57	859055.84	861258.45	2021.08
3.00	854704.20	857516.57	864532.16	2460.07	860201.66	860634.88	861299.18	584.11
4.00	854704.20	855972.43	860719.62	1400.74	858894.21	859940.21	861814.92	1627.17
6.00	854704.20	855696.04	860854.08	1558.04	860482.35	861555.25	862757.90	1143.31
8.00	854704.20	855235.94	858070.33	926.69	856113.30	857077.42	857811.30	872.10
10.00	854704.20	855055.72	857048.65	698.75	856287.58	857030.77	858070.33	927.59
15.00	854704.20	854903.45	856879.16	536.89	857481.90	858565.45	859370.55	974.63
25.00	854704.20	854754.90	855971.75	253.51	855971.75	857041.13	857811.30	955.58
35.00	854704.20	854776.63	855971.75	298.51	856113.30	857077.42	857811.30	872.10
45.00	854704.20	854830.95	855971.75	386.76	855781.10	856761.69	857736.91	977.91
50.00	854704.20	854746.45	855971.75	231.42	854704.20	856948.95	859263.50	634.40
60.00	854704.20	854788.70	855971.75	321.58	856113.30	856834.10	857307.68	826.61

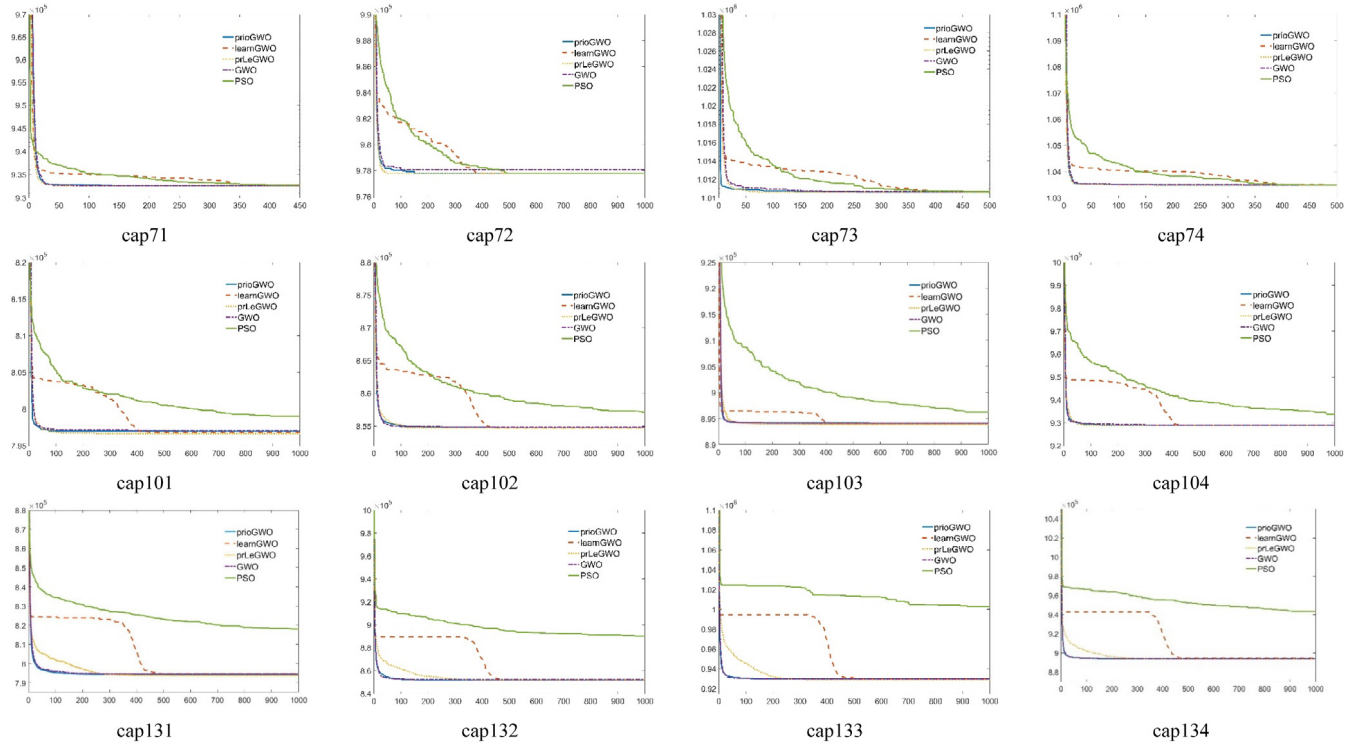

Fig. 4. Convergence plots in UFLP.

4.6. Computational results for 0-1 KP

Results over 30 independent replications for 0-1 KP are presented in Table 5. The first part of this table, which represents the standard benchmarks of 0-1 KP, are adopted from Zou et al. [65]. The second part of Table 5 is related to the larger-scaled and more challenging benchmark problems. These instances are also generated in regard to the procedures reported by the same authors [65]. Since the optimum solution is not known in the second group of benchmarks, *Dantzig upper bound* is used for comparisons.

In this technique, mathematical model is relaxed first. Accordingly, items are sorted by non-increasing order of *profit*/*weight* ratios. Sorted items are assigned to the knapsack until an item s violates knapsack capacity. Thus, the optimum value of the LP-relaxed 0-1 KP can be found via Eq. (24). Finally, the parentage (*gap*%) in Table 5 is obtained by the formulation given in Eq. (25). The best-found *gap*% is highlighted in this table.

$$UB = \sum_{r=1}^{s-1} p_r + \left(C - \sum_{r=1}^{s-1} w_r \right) \times \frac{p_r}{w_r} \tag{24}$$

$$gap\% = \left((UB - best) \big/ UB \right) \times 100 \tag{25}$$

According to the results of Table 5, it is clear that all algorithms perform similarly in the first 10 benchmarks, which include smaller-scaled instances. All tested algorithms obtain optimum values in all replications here. Therefore, it is difficult to come to a conclusion by considering the results of these instances. However, considering instances *kp*11-*kp*18, it is clear that all modifications of GWO are more likely to achieve better results than those of the standard GWO's. Moreover, *prLe*GWO obtains the best results in almost all larger-scaled benchmarks. Additionally, it can be concluded that the GWO and its modifications perform better than the standard PSO.

It is also surprising that *prio*GWO apparently performs better than *learn*GWO in the larger-scaled instances of 0-1 KP. What is more, simultaneous use of the procedures of *prio*GWO and *learn*GWO in *prLe*GWO shows the best performance. This demonstrates that earlier relocation of dominant wolves at the beginning of each iteration as in *prio*GWO, adds to the performance of the standard GWO regardless of the additional procedures used in this modification. Finally, it can be concluded that *prLe*GWO exhibits a more intensified search around the promising regions by making use of learning curves, which also yields to better results.

To sum up, one can conclude that all developed GWO extensions in the present study seem to be performing better than the standard GWO, which is still more efficient than PSO. Although this priori analysis points out efficiency of the developed GWO extensions and shows the importance of the dominant wolves in GWO, in order to demonstrate what intuition suggests, a further statistical analysis is crucially required. In this regard, the next subsection is devoted to a detailed statistical analysis to reveal some possible significant improvements.

4.7. Statistical analysis

Generally, two types of statistical tests, namely parametric and non-parametric tests can be conducted while comparing multiple classifiers. However, because safe use of parametric tests is dependent to some particular conditions (normality, independence and homoscedasticity) that are not fulfilled in the present study, non-parametric tests are employed [66].

In this regard, first, Friedman Test is performed to detect at least a single significantly different pair among a group of algorithms. The obtained p-values of the Friedman Test for each of the benchmarks are presented in Table 6. Since the *mean* values for UFLP are not reported by the previous publications, *hit* values are used here. For the rest of the benchmarks, tests are conducted considering the *best* and the *mean* results. According to the results of Table 6, the null hypothesis, which is based on the assumption of the equivalence of medians, can be rejected for

Table 3
Computational results for function optimization.

Functions		PSO	GWO	*prio*GWO	*learn*GWO	*prLe*GWO
f_1	*Best*	6.3403e−06	8.0408e−29	1.2238e−32	**2.8294e−86**	7.0735e−86
	Mean	1.2372e−04	9.2511e−28	5.9291e−31	6.8563e−86	1.2082e−85
	Std.	2.0263e−04	8.1892e−28	7.8452e−31	2.6225e−86	2.9392e−86
f_2	*Best*	0.0051e+00	1.9238e−17	1.2240e−19	4.3382e−44	**4.1467e−44**
	Mean	0.0502e+00	1.0929e−16	1.1661e−18	2.6605e−43	5.7595e−44
	Std.	0.0854e+00	8.0153e−17	1.2987e−18	9.2247e−43	1.0007e−44
f_3	*Best*	29.6650e+00	4.1206e−08	3.6989e−08	**5.9662e−84**	7.6217e−84
	Mean	82.3008e+00	9.3313e−06	3.6536e−10	1.6235e−83	1.4819e−83
	Std.	34.1431e+00	2.4572e−05	7.8413e−08	6.9151e−84	4.4245e−84
f_4	*Best*	0.7687e+00	1.5499e−07	4.2754e−09	**5.4482e−43**	1.1165e−42
	Mean	1.0902e+00	7.7414e−07	5.7433e−08	8.2990e−43	1.4661e−42
	Std.	0.2144e+00	7.5291e−07	7.6607e−08	1.6503e−43	1.6996e−43
f_5	*Best*	0.0539e+00	5.8221e−04	1.6319e−04	**4.7383e−06**	9.6546e−04
	Mean	0.1541e+00	0.0019e+00	0.0290e+00	2.4110e−04	0.0319e+00
	Std.	0.0560e+00	9.6937e−04	0.0220e+00	2.4623e−04	0.0307e+00
f_6	*Best*	18.0524e+00	5.6843e−14	**0.000e+00**	**0.0000e+00**	**0.0000e+00**
	Mean	55.7322e+00	4.0940e+00	0.000e+00	92.4891e+00	0.0000e+00
	Std.	14.4928e+00	4.3307e+00	0.000e+00	1.1598e+02	0.0000e+00
f_7	*Best*	0.0020e+00	7.5495e−14	7.9936e−15	**8.8817e−16**	1.5099e−14
	Mean	0.2959e+00	1.0865e−13	1.6283e−14	8.8817e−16	6.7916e−14
	Std.	0.5289e+00	2.1056e−14	3.7706e−15	0.0000e+00	3.7837e−14
f_8	*Best*	8.6190e−07	**0.0000e+00**	**0.000e+00**	**0.0000e+00**	**0.0000e+00**
	Mean	0.0118e+00	0.0041e+00	3.9389e−04	0.0000e+00	0.0000e+00
	Std.	0.0118e+00	0.0079e+00	0.0021e+00	0.0000e+00	0.0000e+00
f_9	*Best*	**0.9980e+00**	**0.9980e+00**	2.9821e+00	**0.9980e+00**	12.6705e+00
	Mean	3.5890e+00	3.8099e+00	11.4507e+00	1.7861e+00	12.6705e+00
	Std.	3.0390e+00	3.7121e+00	2.6262e+00	1.9486e+00	3.6134e−15
f_{10}	*Best*	5.1025e−04	3.0749e+00	**3.0858e−04**	4.3815e−04	0.1484e+00
	Mean	8.9683e−04	0.0045e+00	0.0052e+00	0.0092e+00	0.1484e+00
	Std.	1.9605e−04	0.0081e+00	0.0087e+00	0.0098e+00	0.0000e+00
f_{11}	*Best*	**−1.0316e+00**	**−1.0316e+00**	**−1.0316e+00**	**−1.0316e+00**	−4.2334e−71
	Mean	−1.0316e+00	−1.0316e+00	−1.0316e+00	−1.0316e+00	−9.5886e−72
	Std.	6.0458e−16	2.4026e−08	1.8443e−07	1.2658e−05	7.8889e−72
f_{12}	*Best*	**0.3978e+00**	**0.3978e+00**	**0.3978e+00**	**0.3978e+00**	55.6021e+00
	Mean	0.3978e+00	0.3978e+00	0.3978e+00	0.4002e+00	55.6021e+00
	Std.	0.000e+00	3.3911e−04	3.8516e−06	0.0023e+00	2.8907e−14

the significance level of $\alpha = 0.10$ for all benchmarks. In order to detect significantly different pairs while eliminating the family-wise error rate, post-hoc procedures are applied next. Results of these tests are presented in Tables 7–9, where + and $\sim$ denote significant and insignificant differences, respectively.

According to the results of Table 7, where pairwise comparisons are conducted for function optimization benchmarks, Nemenyi Test points out significant differences in comparisons of PSO vs. *learn*GWO and GWO vs. *learn*GWO in terms of finding the best solution for significance level $\alpha = 0.10$. This means that *learn*GWO significantly improves the standard GWO. In parallel, it is also found as significantly better than PSO, which is already outperformed by the standard GWO. Holm Test shows an additional significant difference between PSO and *prio*GWO. However, *prio*GWO is not found as significantly better than the standard GWO also by the Holm Test.

Same post-hoc tests are carried out for the mean results in Table 7. According to the results of this part of Table 7, both Nemenyi and Holm tests do not point out significant differences in terms of finding the most promising mean results except for the comparison of PSO vs. *learn*GWO.

Similar results can be derived from Table 8. The developed GWO extensions, which are based on some modifications on the dominant wolves, significantly improve the most of the previously published results in the literature including the standard GWO. It is also clear that the standard GWO cannot achieve significantly better results than those of the previously reported algorithms'. This statistically demonstrates the importance of the dominant wolves in GWO. Moreover, it can also be put forward that the performance of the developed GWO extensions can be considered as promising algorithms in UFLP, where transfer functions are employed distinctly from function optimization benchmarks.

Final post-hoc tests are conducted for the larger-scaled instances ($kp11$–$kp18$) of 0-1 KP. According to the results presented in Table 9, one can conclude that learning curves combined with prioritized relocation of the dominant wolves significantly add to the performance of the standard GWO in 0-1 KP. It should also be recalled that transfer functions are not used in this benchmark. Therefore, it can additionally be put forward that employing transfer functions does not affect the performance of the developed GWO extensions. Finally, one can conclude that, in parallel with the previous findings, in comparison to GWO and its modifications in the present study, PSO can be considered as a naïve optimizer for the given experimental conditions.

4.8. Complexity analysis

Finally, computational costs for the proposed GWO extensions are presented in this section. From the pseudo code given by Algorithm 1, execution time for the standard GWO depends on the parameters *maxIter*, *popSize* and d, where d is the dimension for any type of problem used in the present study. Thus, execution time complexity of the standard GWO can be represented by $\mathcal{O}(maxIter \times d \times popSize)$ [49]. This is also valid for PSO.

Table 4
Computational results for UFLP.

Algorithms		*Instances* (*facilities* ×*demand points*)											
		16 × 50				25 × 50				50 × 50			
		cap71	cap72	cap73	cap74	cap101	cap102	cap103	cap104	cap131	cap132	cap133	cap134
CPSO [61]	*Best*	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20	893782.11	928941.75	795291.86	851495.33	893076.71	928941.75
	Mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	Worst	934199.14	983713.81	1012643.69	1045342.23	802457.23	857380.85	899424.91	944394.83	804549.64	865667.16	909908.70	951803.25
	Std.	562.23	1324.30	702.13	2124.54	1480.72	1015.64	1695.79	3842.64	2429.54	4297.07	4210.93	6619.05
	Hit	25	25	22	0	0	10	0	18	0	0	0	7
ABC_{bin} [62]	*Best*	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33	893076.71	928941.75
	Mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	Worst	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20	894008.14	928941.75	794910.64	851636.70	895407.93	928941.75
	Std.	0.00	0.00	0.00	0.00	0.00	0.00	85.67	0.00	1065.73	213.28	561.34	0.00
	Hit	**30**	**30**	**30**	**30**	**30**	**30**	25	**30**	6	14	5	**30**
bWSA [63]	*Best*	932615.750	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	793439.56	851495.32	893076.71	928941.75
	Mean	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*	*Na*
	Worst	932615.750	977799.40	1010641.45	1034976.97	797508.72	854704.20	895027.18	928941.75	798449.03	852257.97	894801.16	934586.97
	Std.	0.00	0.000	0.00	0.00	380.43	0.00	470.95	0.00	1025.78	251.65	501.91	1016.14
	Hit	**30**	**30**	**30**	**30**	22	**30**	10	**30**	6	23	7	26
PSO	*Best*	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	803826.01	879785.46	921608.03	944335.68
	Mean	932615.75	977799.40	1010641.45	1034976.97	799006.37	857115.01	896226.99	933746.35	817976.18	889889.93	943216.76	1002804.33
	Worst	932615.75	977799.40	1010641.45	1034976.97	801769.38	860573.81	900406.76	941871.31	824751.43	896935.53	957975.85	1035373.88
	Std.	0.00	0.00	0.00	0.00	1085.48	1268.63	1763.88	3358.73	4545.68	4697.11	9163.94	19102.20
	Hit	**30**	**30**	**30**	**30**	1	2	3	2	0	0	0	0
GWO	*Best*	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	793439.56	851495.32	893076.71	928941.75
	Mean	932615.75	978056.06	1010702.63	1034976.97	797123.96	854830.95	894083.99	928941.75	794606.70	852034.68	894088.39	929716.82
	Worst	932615.75	981649.35	1012476.97	1034976.97	799144.68	855971.75	895027.18	928941.75	797570.30	854704.20	896529.71	935122.70
	Std.	0.00	976.76	335.11	0.00	772.61	386.76	480.63	0.00	1292.51	789.94	1006.16	1783.31
	Hit	**30**	28	29	**30**	19	27	20	**30**	11	13	5	20
*prio*GWO	*Best*	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	793439.56	851495.32	893076.71	928941.75
	Mean	932615.75	977799.40	1010641.45	1034976.97	796963.87	854788.70	894102.76	928941.75	794187.91	851735.81	893925.90	929612.32
	Worst	932615.75	977799.40	1010641.45	1034976.97	797508.72	855971.75	895027.18	928941.75	796662.92	853419.13	895049.66	934586.97
	Std.	0.00	0.00	0.00	0.00	421.65	321.58	459.70	0.00	1000.79	446.02	587.93	1699.92
	Hit	**30**	**30**	**30**	**30**	19	28	15	**30**	15	17	6	21
*learn*GWO	*Best*	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	793439.56	851495.32	893076.71	928941.75
	Mean	932615.75	977799.40	1010641.45	1034976.97	796849.17	854704.20	894016.05	928941.75	794749.63	852229.93	894390.11	929834.29
	Worst	932615.75	977799.40	1010641.45	1034976.97	797508.72	854704.20	894801.16	928941.75	798338.45	855005.57	897548.76	934586.97
	Std.	0.00	0.00	0.00	0.00	370.08	0.00	406.54	0.00	1476.05	1212.19	1214.52	1786.97
	Hit	**30**	**30**	**30**	**30**	23	**30**	20	**30**	10	16	5	20
*prLe*GWO	*Best*	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893782.11	928941.75	793439.56	851495.32	893076.71	928941.75
	Mean	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	893865.11	928941.75	793781.44	851616.90	893717.00	928941.75
	Worst	932615.75	977799.40	1010641.45	1034976.97	796648.43	854704.20	894801.16	928941.75	794373.41	853525.52	894801.16	928941.75
	Std.	0.00	0.00	0.00	0.00	0.00	0.00	260.79	0.00	426.88	429.35	571.45	0.00
	Hit	**30**	**30**	**30**	**30**	**30**	**30**	**26**	**30**	**18**	**26**	**10**	**30**

Table 5
Computational results for 0-1 KP.

			PSO			GWO			*prio*GWO			*learn*GWO			*prLe*GWO		
	Items	*Opt.*	*Best*	*Mean*	*Hit*	*Best*	*Mean*	*Hit*	*Best*	*Mean*	*Hit*	*Best*	*Mean*	*Hit*	*Best*	*Mean*	*Hit*
*kp*1	10	*295*	295	295	30	295	295	30	295	295	30	295	295	30	295	295	30
*kp*2	20	*1024*	1024	1024	30	1024	1024	30	1024	1024	30	1024	1024	30	1024	1024	30
*kp*3	4	*35*	35	35	30	35	35	30	35	35	30	35	35	30	35	35	30
*kp*4	4	*23*	23	23	30	23	23	30	23	23	30	23	23	30	23	23	30
*kp*5	15	*481.0694*	481.0694	481.0694	30	481.0694	481.0694	30	481.0694	481.0694	30	481.0694	481.0694	30	481.0694	481.0694	30
*kp*6	10	*52*	52	52	30	52	52	30	52	52	30	52	52	30	52	52	30
*kp*7	7	*107*	107	107	30	107	107	30	107	107	30	107	107	30	107	107	30
*kp*8	23	*9767*	9767	9767	30	9767	9767	30	9767	9767	30	9767	9767	30	9767	9767	30
*kp*9	5	*130*	130	130	30	130	130	30	130	130	30	130	130	30	130	130	30
*kp*10	20	*1025*	1025	1025	30	1025	1025	30	1025	1025	30	1025	1025	30	1025	1025	30
	items	*UB.*	*Best*	*Mean*	*Gap%*	*Best*	*Mean*	*Gap%*	*Best*	*Mean*	*Gap%*	*Best*	*Mean*	*Gap%*	*Best*	*Mean*	*Gap%*
*kp*11	100	*7016.50*	6999	6985.93	0.24	7003	6996.30	**0.19**	7001	6994.96	0.22	7003	6989.36	**0.19**	7003	6995.26	**0.19**
*kp*12	200	*11639.21*	10878	10763.23	6.54	11574	11509.30	0.56	11614	11545.16	0.21	11552	11405.30	0.74	11624	11571.66	**0.13**
*kp*13	300	*14254.57*	12034	11900.03	15.5	14152	14017.36	0.71	14118	14010.63	0.95	14036	13812.60	1.53	14193	14102.40	**0.43**
*kp*14	500	*18640.57*	14463	14208.83	22.4	17952	17619.43	3.69	18003	17651.80	3.42	17478	17037.53	6.23	18206	18020.10	**2.33**
*kp*15	800	*40607.35*	33485	33111.96	17.5	39779	39263.53	2.03	39971	39645.56	**1.56**	38711	37936.96	4.66	39307	38878.10	3.20
*kp*16	1000	*67256.33*	63069	62900.53	6.20	66154	65990.63	1.63	66238	65942.50	1.51	65389	65130.93	2.77	66528	66315.96	**1.08**
*kp*17	1200	*86994.11*	85370	85207.70	1.86	86611	86482.86	0.44	86690	86593.40	0.34	86225	86026.00	0.88	86698	86516.20	**0.34**
*kp*18	1500	*103745.10*	99033	98715.30	4.54	102453	102132.50	1.24	102529	102166.06	1.17	101534	101080.13	2.13	102797	102521.73	**0.91**

Table 6
Average ranks of the algorithms and obtained *p*-values of Friedman Tests.

Overall avg. ranks						
Algorithms	*UFLP*		*Func. optim.*		*0-1 KP*	
	Hit	*Algorithms*	*Best*	*Mean*	*Best*	*Mean*
CPSO [61]	7.541	PSO	4.167	3.875	5.000	5.000
ABC_{bin} [62]	3.458	GWO	3.458	3.208	2.625	2.500
bWSA [63]	3.958	*prio*GWO	2.542	2.750	2.250	2.000
PSO	6.291	*learn*GWO	**1.708**	**2.167**	3.750	3.875
GWO	4.916	*prLe*GWO	3.125	3.000	**1.375**	**1.500**
*prio*GWO	3.750					
*learn*GWO	3.750					
*prLe*GWO	**2.333**					
p-values (*adj. ties*)	0.000		0.001	0.090	0.000	0.000
Decision on H_0	Reject H_0		Reject H_0	Reject H_0	Reject H_0	Reject H_0

Table 7
Pairwise comparisons and post-hoc test results for function optimization benchmarks.

Best results				*Mean results*			
Comparisons	*Unadjusted*	*Nemenyi*	*Holm*	*Comparisons*	*Unadjusted*	*Nemenyi*	*Holm*
PSO vs *learn*GWO	+	+	+	PSO vs *learn*GWO	+	+	+
GWO vs *learn*GWO	+	+	+	PSO vs *prio*GWO	+	∼	∼
PSO vs *prio*GWO	+	∼	+	GWO vs *learn*GWO	∼	∼	∼
*learn*GWO vs *prLe*GWO	+	∼	∼	PSO vs *prLe*GWO	∼	∼	∼
PSO vs *prLe*GWO	∼	∼	∼	*learn*GWO vs *prLe*GWO	∼	∼	∼
GWO vs *prio*GWO	∼	∼	∼	PSO vs GWO	∼	∼	∼
*prio*GWO vs *learn*GWO	∼	∼	∼	*prio*GWO vs *learn*GWO	∼	∼	∼
PSO vs GWO	∼	∼	∼	GWO vs *prio*GWO	∼	∼	∼
*prio*GWO vs *prLe*GWO	∼	∼	∼	*prio*GWO vs *prLe*GWO	∼	∼	∼
GWO vs *prLe*GWO	∼	∼	∼	GWO vs *prLe*GWO	∼	∼	∼

Table 8
Pairwise comparisons and post-hoc test results for UFLP.

Hit values							
Comparisons	*Unadjusted*	*Nemenyi*	*Holm*	*Comparisons*	*Unadjusted*	*Nemenyi*	*Holm*
*prLe*GWO vs. CPSO	+	+	+	*prio*GWO vs. *prLe*GWO	∼	∼	∼
CPSO vs. ABC_{bin}	+	+	+	*learn*GWO vs. *prLe*GWO	∼	∼	∼
*prLe*GWO vs. PSO	+	+	+	PSO vs. GWO	∼	∼	∼
*prio*GWO vs. CPSO	+	+	+	CPSO vs. PSO	∼	∼	∼
*learn*GWO vs. CPSO	+	+	+	GWO vs. *prio*GWO	∼	∼	∼
CPSO vs. bWSA	+	+	+	GWO vs. *learn*GWO	∼	∼	∼
ABC_{bin} vs. PSO	+	∼	+	ABC_{bin} vs. *prLe*GWO	∼	∼	∼
GWO vs. CPSO	+	∼	∼	bWSA vs. GWO	∼	∼	∼
*prLe*GWO vs. GWO	+	∼	∼	ABC_{bin} vs. bWSA	∼	∼	∼
*prio*GWO vs. PSO	+	∼	∼	ABC_{bin} vs. *prio*GWO	∼	∼	∼
*learn*GWO vs. PSO	+	∼	∼	ABC_{bin} vs. *learn*GWO	∼	∼	∼
bWSA vs. PSO	+	∼	∼	bWSA vs. *prio*GWO	∼	∼	∼
*prLe*GWO vs. bWSA	+	∼	∼	bWSA vs. *learn*GWO	∼	∼	∼
GWO vs. ABC_{bin}	∼	∼	∼	*prio*GWO vs. *learn*GWO	∼	∼	∼

Table 9
Pairwise comparisons and post-hoc test results for 0-1 KP.

Best results				*Mean results*			
Comparisons	*Unadjusted*	*Nemenyi*	*Holm*	*Comparisons*	*Unadjusted*	*Nemenyi*	*Holm*
*prLe*GWO vs. PSO	+	+	+	*prLe*GWO vs. PSO	+	+	+
*prio*GWO vs. PSO	+	+	+	*prio*GWO vs. PSO	+	+	+
GWO vs. PSO	+	+	+	GWO vs. PSO	+	+	+
*learn*GWO vs. *prLe*GWO	+	+	+	*learn*GWO vs. *prLe*GWO	+	+	+
*learn*GWO vs. *prio*GWO	+	+	∼	*learn*GWO vs. *prio*GWO	+	∼	+
PSO vs. *learn*GWO	∼	∼	∼	*learn*GWO vs. GWO	+	∼	∼
GWO vs. *prLe*GWO	∼	∼	∼	PSO vs. *learn*GWO	∼	∼	∼
*learn*GWO vs. GWO	∼	∼	∼	GWO vs. *prLe*GWO	∼	∼	∼
*prio*GWO vs. *prLe*GWO	∼	∼	∼	GWO vs. *prio*GWO	∼	∼	∼
GWO vs. *prio*GWO	∼	∼	∼	*prio*GWO vs. *prLe*GWO	∼	∼	∼

Table 10
Computational results for the case study of UFLP.

	PSO	GWO	*prioGWO*	*learn*GWO	*prLe*GWO
Best	**100734.17**	**100734.17**	**100734.17**	**100734.17**	**100734.17**
Mean	104349.78	100826.06	100822.55	101028.42	**100800.45**
Worst	125552.31	102827.99	**101397.03**	102827.99	**101397.03**
Std.	7436.60	396.98	229.18	564.65	**202.25**
Hit	15	25	26	23	**27**

Table 11
Computational results for the case study of 0-1 KP.

Cases		PSO	GWO	*prioGWO*	*learn*GWO	*prLe*GWO
Off-peak time	*Best*	4090	4090	4090	4090	4090
	Mean	4090	4090	4090	4090	4090
	Hit	30	30	30	30	30
Peak time	*Best*	5760	5760	5760	5760	5760
	Mean	5760	5760	5760	5760	5760
	Hit	30	30	30	30	30
Prime time	*Best*	9045	9045	9045	9045	9045
	Mean	9045	9045	9045	9045	9045
	Hit	30	30	30	30	30
Premium time	*Best*	10750	10750	10750	10750	10750
	Mean	10750	10750	10750	10750	10750
	Hit	30	30	30	30	30

As one can see from the pseudo codes given in Algorithm 2 and Algorithm 3, additional loops or function evaluations are not used in *prio*GWO, *learn*GWO and *prLe*GWO. It is clear that learning coefficients can be generated before the execution of the algorithm or it can simultaneously be evaluated throughout iterations. Similarly, the dominant wolves are already known in *prio*GWO, therefore, computational burden also remains same in *prio*GWO. Thus, it can be concluded that complexity of all used algorithms can be represented by $\mathcal{O}(maxIter \times d \times popSize)$ [49].

Putting things together, these results statistically verify that the strategy of utilizing dominant wolves in GWO can bring about significant differences in overall performance. However, while some of the proposed extensions in the present study significantly add to the existing techniques, some do not. This is an expected result because; it is not possible to develop an algorithm that significantly outperforms all existing algorithms in all benchmarking problems in terms of all performance metrics. Nevertheless, it is clear that the proposed extensions are promising for solving various optimization problems.

4.9. Real-life applications

The proposed algorithms are implemented on some real-life cases of which the data are taken from the related publications [67,68].

The first case study is devoted to implementation of UFLP. In a recent work, Turkoglu et al. [67] solve ATM (automated teller machine) deployment problem as UFLP. This problem is known to be an important strategic problem for banks. For implementation, the authors select Beşiktaş, a municipality of Istanbul with population of 186.570 and 23 districts [67]. According to Turkoglu et al. [67] ATMs are not dispersed uniformly to these districts. The aim here is to deploy those ATMs to possible candidate locations so that the residents can be served with the minimum cost that occurs from ATM deployment cost and cost of the distance between the centres of the districts and the deployed ATM locations. In this regard, distance matrix is calculated first. The authors propose to calculate distance matrix by using great circle distance metric, employing the latitude and longitude data for each of the ATMs and centres of the districts. There are 22 candidate ATM deployment locations and 23 customers (districts as demand points) in the mentioned case study. Therefore, this problem is size of 22×23. Other related data and required coordinates for this case study are presented by Turkoglu et al. [67], however, great circle distance is calculated first in order to make use of the data of Turkoglu et al. [67]. All compared algorithms are run for 30 replications of 100 iterations, and obtained results are presented in Table 10.

As one can see from Table 10, all algorithms are capable of finding the same *best* result. However, *mean* result performance of *prLe*GWO, which is slightly better than those of *prio*GWO's and GWO's, seems to be more promising. Accordingly, the same *worst* result performance is achieved by *prio*GWO and *prLe*GWO. Finally, as one can see from Table 10, the most promising *standard deviance* and *hit* performances are achieved by *prLe*GWO although it is slightly better than other GWO modifications again. It is also clear that all GWO modifications seem to be performing better than the standard PSO.

The second case study is devoted to implementation of 0-1 KP. Ntow [68] proposes to solve an advertisement selection problem as 0-1 KP. The aim in this problem is to choose a set of advertisements from among a set of candidate advertisement items so that the advertisements reach the audience with the most efficient way. In this context, Ntow [68] categorizes the spot advertisements into four categories, such as *off-peak time*, *peak time*, *prime time* and *premium time*, with 21, 18, 23 and 16 items, respectively. Capacity of the knapsack is assumed as 360 secs for all cases. All other related data and parameters are presented by Ntow [68]. Finally, all compared algorithms are run for 30 replications of 100 iterations again, and obtained results are presented in Table 11.

According to the results of Table 11, it is clear that all algorithms achieve same performances. It is an expected result because; this test case is size of the small instances that are presented in Table 5. Therefore, it is not possible to come to a conclusion about the performances of the proposed algorithms for 0-1 KP case study. Nevertheless, it can be concluded that parallel findings are achieved in accordance with the small-scaled instances of Table 5.

To sum up, developed algorithms are capable of finding promising results also by using real data. Thus, one can conclude that the proposed modifications can be considered as promising algorithms also in real-life cases.

5. Concluding remarks

The present study aims to observe the effects of dominant wolves on the efficiency of Grey Wolf Optimization (GWO), which is one of the new generation bio-inspired metaheuristic algorithms. In this context, three new extensions of GWO, only based on some modifications of dominant wolves, are introduced in this study.

In the prior extension, three dominant wolves are repositioned first by sharing information with each other. That is to say, the rest of the population is allowed to reposition according to the dominant wolves, only after those leading wolves are repositioned first. Thus, an implicit local search without an additional computational cost is conducted at the beginning of each iteration. In the latter modification, dominant wolves undergo some role-specific learning curves so that the hierarchy amongst the leading wolves is established throughout iterations. These modifications are simultaneously used in the final extension of GWO proposed in this paper.

Table A.1
Used benchmarking functions in real-valued optimization.

Function	*Dim.*	*Range*	*Opt.*
$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100,\ 100]$	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$[-10,\ 10]$	0
$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100,\ 100]$	0
$f_4(x) = \max_i \{ \lvert x_i \rvert ,\ 1 \le i \le D \}$	30	$[-100,\ 100]$	0
$f_5(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	30	$[-1.28,\ 1.28]$	0
$f_6(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	30	$[-5.12,\ 5.12]$	0
$f_7(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	30	$[-32,\ 32]$	0
$f_8(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600,\ 600]$	0
$f_9(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$	2	$\begin{bmatrix} -65.5360, \\ 65.5360 \end{bmatrix}$	1
$a_{ij} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix}$			
$f_{10}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5,\ 5]$	0.00030
$a_i = [0.1957,\ 0.1947,\ 0.1735,\ 0.1600,\ 0.0844,\ 0.0627,\ 0.0456,\ 0.0342,\ 0.0323,\ 0.0235,\ 0.0246] \quad i = 1, \ldots, 11$ $b_i = [4.000,\ 2.000,\ 1.000,\ 0.5000,\ 0.2500,\ 0.1667,\ 0.1250,\ 0.1000,\ 0.0833,\ 0.0714,\ 0.0625\ ] \quad i = 1, \ldots, 11$			
$f_{11}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,\ 5]$	−1.0316
$f_{12}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi} \right)\cos x_1 + 10$	2	$x_1 \in [-5,\ 10]$ $x_2 \in [0,\ 15]$	0.398

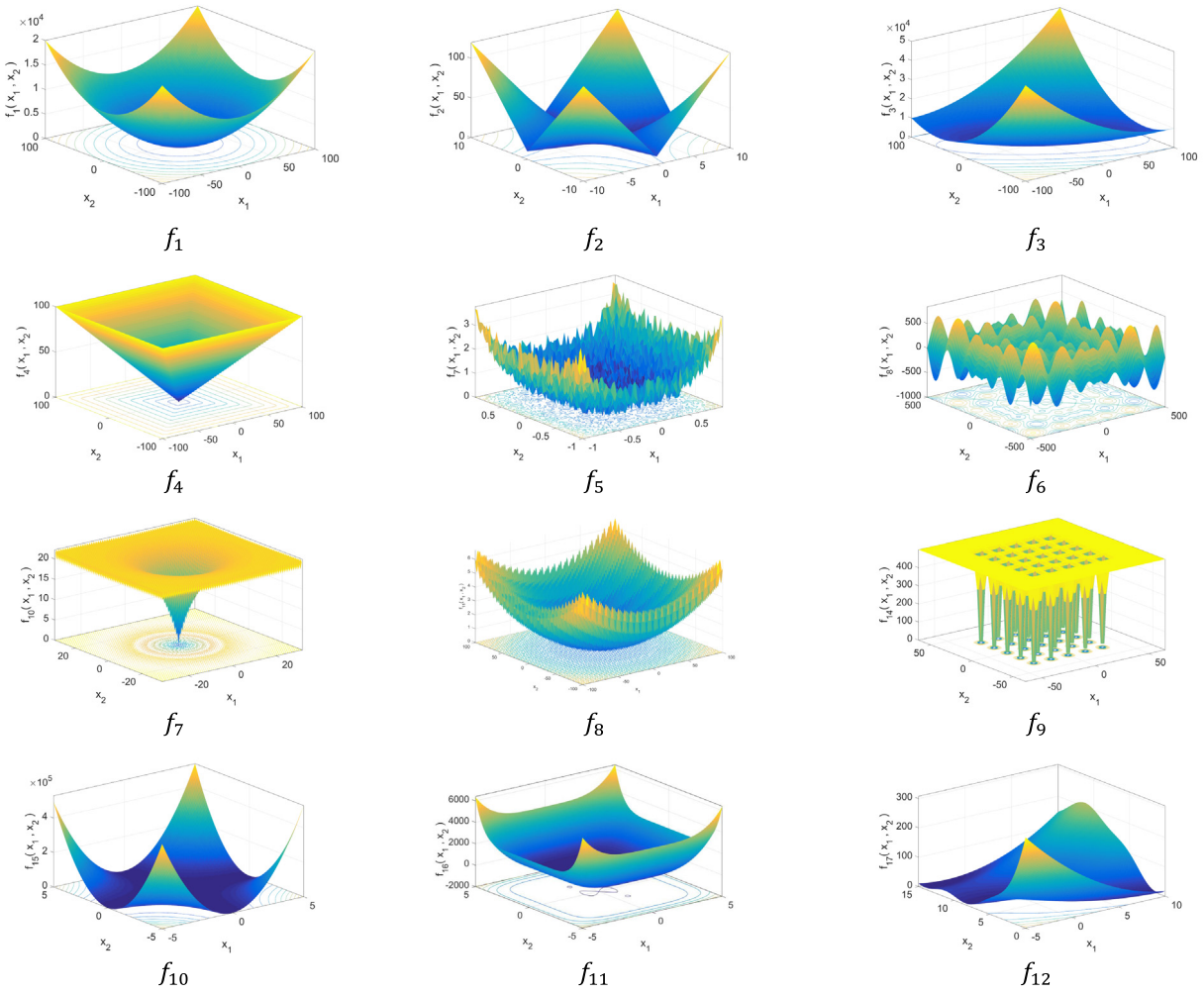

Fig. A.1. 2-Dimensional forms of the used real-valued benchmarking functions.

All developed modifications are compared to the standard GWO and Particle Swarm Optimization, which can be considered as a basis algorithm that is commonly employed in empirical studies. The performances of the proposed algorithms are tested on both constrained and unconstrained optimization problems including real test cases and combinatorial problems such as uncapacitated facility location problem and 0-1 knapsack problem, which have numerous possible real-life applications. Comprehensive experimental study and statically verified results demonstrate that the dominant wolves in GWO have crucial effects on the efficiency of the standard GWO. Moreover, it is shown by the statistical tests that the developed GWO modifications significantly outperform some of the reported algorithms in the related literature. Thus, it can additionally be concluded that the proposed extensions are found as promising and competitive algorithms for the optimization problems used in the present study.

It is clear that learning enhanced GWO modifications in the present study devote positions to dominant wolves of which the weights are changed throughout generations. It is further clear that fitness of solution vectors are used only for defining dominant wolves. Weights are calculated based some user-defined parameters regardless of fitness qualities. Although the proposed GWO modifications archive significant improvements, as an extension of this study, those weights are planned to be modified in accordance with both fitness values and user-supplied parameters to further improve the efficiency of the proposed algorithms. Analysing the effects of different learning techniques while establishing the hierarchy amongst the dominant wolves and making use of different encoding–decoding approaches are also scheduled as future works. Moreover, since GWO is inspired by grey wolves in nature, modifying the proposed GWO extensions as a multi-population based approach, which is indeed in accordance with the cooperation and competition amongst wolf packs, is considered as another future research.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105658.

Appendix

See Table A.1 and Fig. A.1.

References

[1] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989, Reading.

[2] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of IEEE the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39-43.

[3] R. Storn, K. Price, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report TR-95-012, ICSI, 1995, Available via ftp from ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012ps.Z.

[4] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.

[5] X.S. Yang, Firefly algorithms for multimodal optimization, in: O. Watanabe, T. Zeugmann (Eds.), Lecture Notes in Computer Sciences, Springer, Berlin, 2009, pp. 169–178.

[6] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: proceedings of IEEE World Congress on Nature and Biologically Inspired Computing, 2009, pp. 210-214.

[7] K.N. Krishnan, D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, Swarm. Intell. US 3 (2) (2009) 87–124.

[8] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: C. Cruz, J.R. González, N. Krasnogor, D.A. Pelta, G. Terrazas (Eds.), in: Studies in computational intelligence., Springer, Berlin, 2010, pp. 65–74.

[9] R. Tang, S. Fong, X.S. Yang, S. Deb, Wolf search algorithm with ephemeral memory, in: proceedings of IEEE international conference on digital information management (ICDIM), 2012, 165-72.

[10] X.S. Yang, Flower pollination algorithm for global optimization, in: J. Durand-Lose, N. Jonoska (Eds.), Unconventional Computation and Natural Computation. UCNC 2012, in: Lecture notes in computer science, vol. 7445, Springer, Berlin, Heidelberg, 2012.

[11] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.

[12] T.S. Pan, T.K. Dao, T.T. Nguyen, S.C. Chu, A communication strategy for paralleling grey wolf optimizer, in: T. Zin, J.W. Lin, J.S. Pan, P. Tin, M. Yokota (Eds.), Genetic and Evolutionary Computing. GEC 2015, in: Advances in Intelligent Systems and Computing, vol. 388, Springer, Cham, 2016.

[13] S. Saremi, S.Z. Mirjalili, S.M. Mirjalili, Evolutionary population dynamics and grey wolf optimizer, Neural. Comput. Appl. 26 (5) (2015) 1257–1263.

[14] B. Mahdad, K. Srairi, Blackout risk prevention in a smart grid based flexible optimal strategy using Grey Wolf-pattern search algorithms, Energy Convers. Manage. 98 (2015) 411–429.

[15] G.M. Komaki, V. Kayvanfar, Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, J. Comput. Sci-Neth. 8 (2015) 109–120.

[16] X. Song, L. Tang, S. Zhao, X. Zhang, L. Li, J. Huang, W. Cai, Grey wolf optimizer for parameter estimation in surface waves, Soil Dyn. Earthq. Eng. 75 (2015) 147–157.

[17] E. Emary, W. Yamany, A.E. Hassanien, V. Snasel, Multi-objective gray-wolf optimization for attribute reduction, Procedia. Comput. Sci. 65 (2015) 623–632.

[18] S. Gholizadeh, Optimal design of double layer grids considering nonlinear behaviour by sequential grey wolf algorithm, J. Opt. Civil Eng. 5 (4) (2015) 511–523.

[19] E. Emary, H.M. Zawbaa, A.E. Hassanien, Binary grey wolf optimization approaches for feature selection, Neurocomputing 172 (2016) 371–381.

[20] T. Jayabarathi, T. Raghunathan, B.R. Adarsh, P.N. Suganthan, Economic dispatch using hybrid grey wolf optimizer, Energy 111 (2016) 630–641.

[21] M. Pradhan, P.K. Roy, T. Pal, Grey wolf optimization applied to economic load dispatch problems, Int. J. Electr. Power 83 (2016) 325–334.

[22] N. Jayakumar, S. Subramanian, S. Ganesan, E.B. Elanchezhian, Grey wolf optimization for combined heat and power dispatch with cogeneration systems, Int. J. Electr. Power 74 (2016) 252–264.

[23] S. Zhang, Y. Zhou, Z. Li, W. Pan, Grey wolf optimizer for unmanned combat aerial vehicle path planning, Adv. Eng. Softw. 99 (2016) 121–136.

[24] D. Guha, P.K. Roy, S. Banerjee, Load frequency control of interconnected power system using grey wolf optimization, Swarm Evol. Comput. 27 (2016) 97–115.

[25] D. Guha, P.K. Roy, S. Banerjee, Load frequency control of large scale power system using quasi-oppositional grey wolf optimization algorithm, Eng. Sci. Technol. Int. J. 19 (4) (2016) 1693–1713.

[26] S. Mirjalili, S. Saremi, S.M. Mirjalili, L.D.S. Coelho, Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization, Expert Syst. Appl. 47 (2016) 106–119.

[27] R. Katarya, O.P. Verma, Recommender system with grey wolf optimizer and FCM, Neural Comput. Appl. 30 (5) (2018) 1679–1687.

[28] V.K. Kamboj, S.K. Bath, J.S. Dhillon, Solution of non-convex economic load dispatch problem using grey wolf optimizer, Neural Comput. Appl. 27 (5) (2016) 1301–1316.

[29] M.R. Shakarami, I.F. Davoudkhani, Wide-area power system stabilizer design based on grey wolf optimization algorithm considering the time delay, Electr. Power Syst. Res. 133 (2016) 149–159.

[30] L. Rodríguez, O. Castillo, J. Soria, P. Melin, F. Valdez, C.I. Gonzalez, G.E. Martinez, J. Soto, A fuzzy hierarchical operator in the grey wolf optimizer algorithm, Appl. Soft. Comput. 57 (2017) 315–328.

[31] A.A. Heidari, P. Pahlavani, An efficient modified grey wolf optimizer with Lévy flight for optimization tasks, Appl. Soft. Comput. 60 (2017) 115–134.

[32] M.A. Tawhid, A.F. Ali, A hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function, Memet Comput. 9 (4) (2017) 347–359.

[33] C. Lu, L. Gao, X. Li, S. Xiao, A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry, Eng. Appl. Artif. Intell. 57 (2017) 61–79.

[34] W. Long, X. Liang, S. Cai, J. Jiao, W. Zhang, A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems, Neural Comput. Appl. 28 (1) (2017) 421–438.

[35] L. Rodríguez, O. Castillo, J. Soria, A study of parameters of the grey wolf optimizer algorithm for dynamic adaptation with Fuzzy logic, in: P Melin, O. Castillo, J. Kacprzyk (Eds.), Nature-Inspired Design of Hybrid Intelligent Systems, in: Studies in Computational Intelligence, vol. 667, Springer, Cham, 2017.

[36] R.E. Precup, R.C. David, Petriu, Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity, IEEE T Ind. Electr. 64 (1) (2017) 527–534.

[37] B. Yang, X. Zhang, T. Yu, H. Shu, Z. Fang, Grouped grey wolf optimizer for maximum power point tracking of doubly-fed induction generator based wind turbine, Energy Convers Manage. 133 (2017) 427–443.

[38] A.K.M. Khairuzzaman, S. Chaudhury, Multilevel thresholding using grey wolf optimizer for image segmentation, Expert Syst. Appl. 86 (2017) 64–76.

[39] A. Farshin, S. Sharifian, A chaotic grey wolf controller allocator for software defined mobile network (SDMN) for 5th generation of cloud-based cellular systems (5g), Comput. Commun. 108 (2017) 94–109.

[40] W. Long, J. Jiao, X. Liang, M. Tang, An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization, Eng. Appl. Artif. Intell. 68 (2018) 63–80.

[41] L.K. Panwar, S. Reddy, A. Verma, B.K. Panigrahi, R. Kumar, Binary grey wolf optimizer for large scale unit commitment problem, Swarm Evol. Comput. 38 (2018) 251–266.

[42] R.A. Ibrahim, M.A. Elaziz, S. Lu, Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization, Expert Syst. Appl. 108 (2018) 1–27.

[43] A. Saxena, B.P. Soni, R. Kumar, V. Gupta, Intelligent grey wolf optimizer-development and application for strategic bidding in uniform price spot energy market, Appl. Soft. Comput. 69 (2018) 1–13.

[44] A. Khandelwal, A. Bhargava, A. Sharma, H. Sharma, Modified grey wolf optimization algorithm for transmission network expansion planning problem, Arab. J. Sci. Eng. 43 (6) (2018) 2899–2908.

[45] N. Panagant, S. Bureerat, Truss topology, shape and sizing optimization by fully stressed design based on hybrid grey wolf optimization and adaptive differential evolution, Eng. Optim. 50:10 (2018) 1645–166.

[46] S. Khalilpourazari, S. Khalilpourazary, Optimization of production time in the multi-pass milling process via a robust grey wolf optimizer, Neural Comput. Appl. 29 (12) (2018) 1321–1336.

[47] M. Fahad, F. Aadil, S. Khan, P.A. Shah, K. Muhammad, J. Lloret, H. Wang, J.W. Lee, I. Mehmood, Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks, Comput. Electr. Eng. 70 (2018) 853–870.

[48] W. Long, J. Jiao, X. Liang, M. Tang, Inspired grey wolf optimizer for solving large-scale function optimization problems, Appl. Math. Model 60 (2018) 112–126.

[49] M.A. Al-Betar, M.A. Awadallah, H. Faris, I. Aljarah, A.I. Hammouri, Natural selection methods for grey wolf optimizer, Expert Syst. Appl. 113 (2018) 481–498.

[50] S. Gupta, K. Deep, A novel random walk grey wolf optimizer, Swarm. Evol. Comput. 44 (2019) 101–112.

[51] H. Qin, P. Fan, H. Tang, P. Huang, B. Fang, S. Pan, An effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint, Comput. Ind. Eng. 128 (2019) 458–476.

[52] A. Saxena, R. Kumar, S. Das, β-Chaotic map enabled grey wolf optimizer, Appl. Soft. Comput. 75 (2019) 84–105.

[53] K. Luo, Enhanced grey wolf optimizer with a model for dynamically estimating the location of the prey, Appl. Soft. Comput. 77 (2019) 225–235.

[54] F.B. Ozsoydan, A. Baykasoglu, Analysing the effects of various switching probability characteristics in flower pollination algorithm for solving unconstrained function minimization problems, Neural Comput. Appl. (2018) in press.

[55] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput. 9 (2013) 1–14.

[56] G. Laporte, S. Nickel, F.S. da Gama, Location Science, Springer, Berlin, 2015.

[57] J.de. Armas, A.A. Juan, J.M. Marquès, J.P. Pedroso, Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic, J. Oper. Res. Soc. 68 (10) (2017) 1161–1176.

[58] T.S. Hale, C.R. Moberg, Location science research: a review, Ann. Oper. Res. 123 (1) (2003) 21–35.
[59] G. Şahin, H. Süral, A review of hierarchical facility location models, Comput. Oper. Res. 34 (8) (2007) 2310–2331.
[60] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, Berlin, 2004.
[61] K.K. Bhattacharjee, S.P. Sarmah, Modified swarm intelligence based techniques for the knapsack problem, Appl. Intell. 46 (1) (2017) 158–179.
[62] M. Sevkli, A.R. Guner, A continuous particle swarm optimization algorithm for uncapacitated facility location problem, in: M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, T. Stützle (Eds.), in: Lecture notes in computer sciences., vol. 31, Springer, Berlin, 2006, pp. 6–23.
[63] M.S. Kiran, The continuous artificial bee colony algorithm for binary optimization, Appl. Soft. Comput. 33 (2015) 15–23.
[64] A. Baykasoğlu, F.B. Ozsoydan, M.E. Senol, Weighted superposition attraction algorithm for binary optimization problems, Oper. Res. (2018) in press.
[65] D. Zou, L. Gao, S. Li, J. Wu, Solving 0-1 knapsack problem by a novel global harmony search algorithm, Appl. Soft. Comput. 11 (2) (2011) 1556–1564.
[66] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.
[67] D.C. Turkoglu, M.E. Genevois, M. Cedolin, Facility location problems-a case study for ATM site selection, in: proceeding of the 12th Multidisciplinary Academic Conference (MAC), Czech Republic, Prague, 2018.
[68] A.S. Ntow, Knapsack problem a case study of metro t.v. a television station in ghana, MSc Thesis, 2012.