Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 04, No. 03, December 2018.

DISTRIBUTED GREY WOLF OPTIMIZER FOR NUMERICAL OPTIMIZATION PROBLEMS

Bilal H. Abed-alguni and Malek Barhoush

(Received: 29-Jul.-2018, Revised: 10-Sep.-2018, Accepted: 13-Sep.-2018)

ABSTRACT

The Grey Wolf Optimizer (GWO) algorithm is an interesting swarm-based optimization algorithm for global optimization. It was inspired by the hunting strategy and leadership hierarchy of grey wolves. The GWO algorithm has been successfully tailored to solve various continuous and discrete optimization problems. However, the main drawback of GWO is that it may converge to sub-optimal solutions in early stages of its simulation process due to the loss of diversity in its population. This paper introduces a distributed variation of GWO(DGWO) that attempts to enhance the diversity of GWO by organizing its population into small independent groups (islands) based on a well-known distributed model called the island model. DGWO applies the original GWO to each island and then allows selected solutions to be exchanged among the islands based on the random ring topology and the best-worst migration policy. The island model in DGWO provides a better environment for unfit candidate solutions in each island to evolve into better solutions, which increases the likelihood of finding global optimal solutions. Another interesting feature about DGWO is that it can run in parallel devices, which means that its computational complexity can be reduced compared to the computational complexity of existing variations of GWO. DGWO was evaluated and compared to well-known swarm-based optimization algorithms using 30 CEC 2014 functions. In addition, the sensitivity of DGWO to its parameters was evaluated using 15 standard test functions. The comparative study and the sensitivity analysis for DGWO indicate that it provides competitive performance compared to the other tested algorithms. The source code of DGWO is available at: https://www.dropbox.com/s/2d16t46598u03y0/DistributedGreyWolfOptimizer.zip?dl=0.

KEYWORDS

Grey wolf optimizer, Island-based model, Distributed optimization algorithm, Optimization, Swarm-based optimization.

1. INTRODUCTION

Swarm-based optimization algorithms, such as the Bat algorithm [1]-[2], Cuckoo search [3]-[6], Whale optimization [7]-[9], Butterfly optimization [10]-[14], Grasshopper optimization [15], flower pollination [16] and Particle swarm optimization [1], [17], have been successfully used to solve difficult optimization problems in various fields (e.g., image processing [18]-[19], fuzzy logic [20]-[21], control engineering [22]-[25], scheduling problems [26]-[27]). The Grey Wolf Optimizer (GWO) is an interesting swarm-based optimization algorithm that was recently proposed to solve difficult optimization problems based on the hunting strategy and leadership hierarchy of grey wolves [28].

The GWO simulates the leadership hierarchy of grey wolves based on four hierarchical leaderships: alpha wolf, beta wolf, delta wolf and omega wolf. In addition, GWO simulates the hunting strategy of grey wolves based on three sequential steps: searching for prey, encircling prey and attacking prey. GWO has lately attracted much attention from the optimization community due to its attractive advantages. First, GWO is an efficient optimization algorithm that has a simple structure (Figure 1). Second, the simulation process of GWO is controlled by one key parameter (Section 2.1). Finally, GWO has been successfully tailored to solve various continuous optimization problems as well as discrete optimization problems (Section 2.3).

Like most of the swarm-based optimization algorithms, GWO may converge faster than expected to sub-optimal solutions [29]-[30]. This is because the evolutionary operators of GWO may not adequately preserve the diversity of the population over the course of the simulation process of GWO. Swarm-based optimization algorithms can be in general parallelized to run in different machines.

B. Abed-alguni and M. Barhoush are both with the Computer Sciences Department, Yarmouk University, Irbid, Jordan.
Email:Bilal.h@yu.edu.jo

Interestingly, the parallel swarm-based approach offers a possible solution to the problem of premature convergence of most of the swarm-based optimization algorithms [31]-[32]. This might be because the parallel approach allows the population of candidate solutions of a given optimization problem to be divided into several groups, which provides a better environment for unfit candidate solutions to evolve in each group.

The island model, which is a structured population model, can be integrated with the framework of a swarm-based optimization algorithm to facilitate its parallelization [21], [33]. Using the island model, the population of a parallel swarm-based optimization algorithm can be divided into *n* groups (islands), where a swarm-based optimization algorithm is applied independently to the population of each island. These islands periodically exchange selected candidate solutions among each other (i.e., migration process) in an attempt to maintain the diversity of population on each island.

The current paper introduces a Distributed Grey Wolf Optimizer (DGWO) that attempts to enhance the diversity of GWO by organizing its population into small islands based on the island model. DGWO applies the original GWO to the population of each island and then allows selected solutions to be exchanged among the islands based on the random ring topology and the best-worst migration policy.

The rest of the paper is organized as follows: Section 2 provides background information about the Grey Wolf Optimizer algorithm, the distributed island model and related work to the Grey Wolf Optimizer. Section 3 presents and discusses the Distributed Grey Wolf Optimizer algorithm. Section 4 presents the simulation results of the proposed algorithm and finally Section 5 presents the conclusions and future work.

2. PRELIMINARIES

This section briefly summarizes some of the underlying concepts of the grey wolf optimizer (Section 2.1) algorithm and the island model (Section 2.2). This section also provides an overview of recently proposed variations of grey wolf optimizer (Section 2.3).

2.1 Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) algorithm, which was developed by Mirjalili et al. [28], is an interesting nature-inspired optimization algorithm. GWO uses a simulation model based on the hierarchical leadership and hunting strategy of grey wolves. In the wild, grey wolves are top predators, which means that they are at the top of their food chain, with no natural predators. A pack of wolves is normally composed of 6 to 7 wolves, but it might have up to 15 wolves. In a wolf pack, normally an alpha (α) male and an alpha female wolves control the pack. The direct followers to the wolves are the beta (β) and delta (δ) wolves. The β and δ wolves help the α wolves to control and dominate the other wolves in the pack (omega (ω) wolves). The hunting strategy of grey wolves is a three-stage cooperative strategy (tracking and chasing prey, pursuing and encircling prey and attacking prey) [34].

Figure 1 shows the flow of the GWO algorithm. The first step of GWO is to generate a population of wolves (candidate solutions) $\vec{X}_i(i = 1,2, \dots, N)$ for a given optimization problem. Each candidate solution is composed of *M* decision variables $\vec{X}_i = \{x_1, x_2, \dots, x_M\}$. The fitness value of each candidate solution is calculated using the fitness function of the optimization problem to determine the hierarchical structure of the population. The solution with the best calculated fitness value is called the α solution ($\vec{X}_\alpha$), while the solutions with the second and third best fitness values are respectively called the β solution ($\vec{X}_\beta$) and δ solution ($\vec{X}_\delta$). The rest of the solutions in the population are called the ω solutions ($\vec{X}_\omega$) [20].

The improvement loop of GWO is composed of three stages: tracking and chasing prey, pursuing and encircling prey and attacking prey. Encircling prey is mathematically modelled in GWO as follows:

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

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

where *t* is the current iteration number, $\vec{A}$ and $\vec{C}$ are two coefficient vectors, $\vec{X}_p$ *is* the candidate solution

that represents the prey and $\vec{X}(t)$ is the candidate solution that represents the grey wolf.

The two coefficient vectors $\vec{A}$ and $\vec{C}$ can be updated as follows:

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \tag{3}$$

$$\vec{C} = 2.\overrightarrow{r_2} \tag{4}$$

where $\vec{a}$ is a vector with values that linearly decrease from 2 to 0 over the course of the simulation of GWO and $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are two random vectors between 0 to 1.

In GWO, the ω solutions are updated based on the α, B and δ solutions. Equations 5 to 11 represent the update process of the solutions:

$$\overrightarrow{D_\alpha} = \left|\vec{C}_1.\vec{X}_\alpha - \vec{X}\right| \tag{5}$$

$$\overrightarrow{D_\beta} = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}\right| \tag{6}$$

$$\overrightarrow{D_\delta} = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}\right| \tag{7}$$

$$\overrightarrow{X_1} = \vec{X}_\alpha - \overrightarrow{A_1}.(\overrightarrow{D_\alpha}) \tag{8}$$

$$\overrightarrow{X_2} = \vec{X}_\beta - \overrightarrow{A_2}.(\overrightarrow{D_\beta}) \tag{9}$$

$$\overrightarrow{X_3} = \vec{X}_\delta - \overrightarrow{A_3}.(\overrightarrow{D_\delta}) \tag{10}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \tag{11}$$

Approaching and attacking prey (exploitation stage) is simulated in GWO by decreasing the components of $\vec{a}$ from 2 to 0 over the course of simulation of GWO. According to Equation 3, the vector $\vec{a}$ controls the range of values of $\vec{A}$ which are in the range $[-2\vec{a},\ 2\vec{a}]$. It is worth noting that a candidate solution is updated in the direction of the best solutions (α, B and δ solutions) when $|\vec{A} < 1|$. The exploration of the search space is triggered in GWO using two settings. First, the candidate solutions diverge from the best solutions when $|\vec{A} > 1|$. Second, the components of $\vec{C}$, which are in the range [0, 2], provide weights for the best solutions in order to increase the influence of the best solutions $|\vec{C} > 1|$ or decrease their influence $|\vec{C} < 1|$ in Equation 1.

Initialize the population of n candidate solutions $\vec{X}_i(i = 1,2,\dots,n)$
Initialize $\vec{A}$, $\vec{a}$ *and* $\vec{C}$
Calculate the fitness value of each candidate solution
$\vec{X}_\alpha$ *= the best candidate solution*
$\vec{X}_\beta$ *= the second best candidate solution*
$\vec{X}_\delta$ *= = the third best candidate solution*
While *(t < Max number of iterations)*
for *each candidate solution*
    Update the values of the current candidate solution using Equation 11
end for
Update $\vec{A}$, $\vec{a}$ *and* $\vec{C}$
Calculate the fitness value of each candidate solution
Update $\vec{X}_\alpha, \vec{X}_\beta$ *and* $\vec{X}_\delta$
t=t+1
end while
return $\vec{X}_\alpha$

Figure 1. The Grey Wolf Optimizer (GWO) Algorithm.

It is important to note that GWO in its current form can be only applied directly to continuous optimization problems. GWO has been tailored for many real-world discrete optimization problems, such as feature selection [35], economic load dispatch problems [36]-[37] and scheduling problems [38]

-[39].

The computational complexity of GWO (Figure 1) can be calculated as follows:

1. Initializing the population of *n* candidate solutions requires *n* operations.
2. Initializing the parameters of GWO ($\vec{A}$, $\vec{a}$ and $\vec{C}$) requires 3 operations.
3. Calculating the first three best solutions ($\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$) requires *n* operations.
4. The following operations are conducted inside the while loop:
    a. The for loop that is used to update the population requires *n* operations.
    b. Calculating the fitness of each candidate solution in an island requires *n* operations.
    c. Updating $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$ requires *n* operations.
    d. Updating the parameters of GWO ($\vec{A}$, $\vec{a}$ and $\vec{C}$) requires 3 operations.

    Overall, the while loop requires $m.(n + n + 3 + 1)$ operations, where *m* is the maximum number of iterations. The number of operations can be further simplified to *m.n*
5. All the operations of the algorithm can be calculated as $n + 3 + n + m.n$, which can be simplified to *m.n*, because 2*n* is greater than 3 and *m.n* is greater than 2*n*.

In summary, the computational complexity of GWO is O(*m.n*). Note that any basic vector operation has been assumed to cost O(1) in the above analysis.

2.2 Island Model

The island model, which is a distributed population model, can be integrated with the framework of a swarm-based optimization algorithm to facilitate its parallelization [33]. Using the island model, the population of a swarm-based optimization algorithm can be divided into *n* groups (islands). Each island is assigned to a computation device, where a swarm-based optimization algorithm is applied to its population. An interaction process, called migration, is periodically triggered among the islands in the island model. In the migration process, selected candidate solutions are exchanged among islands in an attempt to maintain and amend the diversity of population on each island. The best-worst and random-random policies are the most used migration policies in the literature [31]-[33]. In the best-worst policy, the most fitted solutions in one island (say *m* solutions) are exchanged with the *m* worst fitted solutions in a neighbouring island. In the random- random policy, *m* random solutions in one island are swapped with *m* random solutions in a neighbouring island.

The islands in the island model are normally organized and arranged based on a given migration topology [40]. Star, random-star, mesh, random-mesh, ring and random-ring topologies are some popular migration topologies used with swarm-based algorithms [41]. The prefix "random" in the name of the migration topology, which indicates that the order of the islands in the topology changes each time the migration process is triggered.

Two parameters control the migration process among islands in the island model [42]. First, the migration frequency (M_f), which determines the number of iterations between two consecutive migration waves. Second, the migration rate (M_r), which is a parameter that determines the percentage of solutions to be exchanged between an island and a neighbouring island.

2.3 Variations of Grey Wolf Optimizer

Several hybrid GWO algorithms have been recently proposed in an attempt to control and amend premature convergence of GWO. Jayabarathi et al. [36] proposed a hybrid GWO algorithm that integrates the genetic operators (crossover and mutation) of the genetic algorithm (GA) into GWO to improve its exploration ability. The experimental results in [36] suggested that the hybrid GWO and GA algorithm provides good performance in solving several instances of the economic dispatch problem. However, the proposed hybrid algorithm requires heavy computations compared to the basic GWO. This is expected, because applying the genetic operators at each iteration of GWO for each candidate solution is a time-consuming process. Another disadvantage of the hybrid GWO algorithm is that it has a complex structure compared to the GWO algorithm. Tawhid and Ali [29] integrated the whole GA algorithm into GWO to solve the minimization problem of the energy function of the molecule. Although the hybrid GA and GWO algorithm performs much better than the basic GWO, it requires more computational time than GWO. Jitkongchuen [43] proposed a hybrid swarm-based algorithm,

between the differential evolution (DE) algorithm and GWO, for solving numerical optimization functions. The simulation results in [43] suggested that the hybrid GWO is more reliable and more accurate in solving difficult numerical optimization problems than DE, self-adaptive DE and particle swarm optimization (PSO). Unfortunately, the hybrid GWO and DE algorithm is complex and requires heavy computations compared to GWO. Zawbaa et al. [44] combined Antlion optimization (ALO) and GWO in one algorithm (ALO-GWO). ALO-GWO was particularly designed to solve the feature selection problem in large datasets. The simulation results in [44] indicated that ALO-GWO provides good performance in solving feature extraction problems in large datasets compared to PSO and GA. However, ALO-GWO may require a long processing time as most of the hybrid optimization algorithms.

Various intelligent techniques have been successfully used with GWO to enhance its convergence behaviour. Saremi et al. [45] suggested the EPD-GWO algorithm that uses the evolutionary population dynamics (EPD) technique to improve the diversity of population in GWO. In other words, EPD is used in EPD-GWO to improve the exploration of candidate solutions. EDP repositions the worst candidate solutions in the population into the neighbourhoods of the alpha, beta, delta and omega wolves. The simulation results in [45] indicated that EPD-GWO provides better results than GWO. However, the computational complexity of EPD-GWO is higher than the computational complexity of GWO because EPD has to be repeated at each iteration of EPD-GWO. Rodríguez et al. [46] proposed a dynamic variation of GWO that dynamically tunes the parameters of GWO during the simulation process of GWO in order to obtain the best possible performance out of GWO. However, manipulating the parameters of GWO as in [46] does not provide significant enhancement in the performance than the original GWO algorithm. The enhanced GWO (EGWO) algorithm, that was proposed by Joshi and Arora [47], is an another adaptive variation of GWO. EGWO dynamically changes the values of the key parameter of GWO ($\vec{a}$) over the course of its simulation. Moreover, EGWO enhances the exploitation mechanism of GWO by making the best use of the best solution (alpha solution). The experimental results of EGWO compared to standard optimization algorithms (PSO, Firefly Algorithm (FA) and Flower Pollination Algorithm (FPA)) proves that EGWO is a competitive algorithm for solving constrained optimization problems. Malik et al. [48] introduced the wdGWO algorithm which combines the weighted distance (wd) technique with GWO. In wdGWO, the update strategy of the candidate solutions in GWO is modified and the average function of best solutions is replaced with the weighted sum of best solutions. According to the experimental results in [48], the wdGWO showed superior performance compared to basic optimization algorithms (FA, Artificial bee colony, Cuckoo search and PSO).

Moreover, Emary et al. [49] introduced the experienced Grey Wolf Optimizer (EGWO) algorithm that incorporates reinforcement learning (RL) and artificial neural networks (ANNs) into GWO to improve its performance. EGWO uses RL to update the parameters of each candidate solution in the population of GWO. EGWO uses ANNs to estimate the expertness of each candidate solution. The expertness estimation of a solution is used to control its exploration rate. EGWO was evaluated and compared to three optimization algorithms (GWO, PSO, GA). Joshi and Arora [50] proposed an enhanced GWO (E-GWO) algorithm that uses an improved hunting mechanism to balance between the exploration and exploitation of candidate solutions in GWO. Kohli and Arora [51] proposed the chaotic GWO (CGWO) algorithm that incorporates the chaos theory into GWO in an attempt to improve the convergence behaviour of GWO for constrained optimization problems. In CGWO, several types of chaotic maps are employed to adjust the main parameter of GWO ($\vec{a}$). The simulation results of CGWO compared to well-known algorithms (PSO, Firefly Algorithm (FA) and Flower Pollination Algorithm (FPA)) suggest that CGWO provides good results compared to the other algorithms. Heidari and Pahlavani [52] introduced a modified GWO that uses the greedy selection method and the Lévy flight operator with GWO in an attempt to improve the exploration mechanism of GWO. The simulation results in [52] indicated that the improved GWO is more reliable and more effective in solving both discrete and continuous optimization problems than popular state-of-the-art swarm-based optimization algorithms. However, the greedy selection method is not the best selection mechanism according to Abed-alguni and Alkhateeb [4]. Moreover, using the greedy selection method in an optimization algorithm may cause premature convergence [4], [53]. Gupta and Deep [54]-[55] presented an improved GWO algorithm called RW-GWO that uses random walk to enhance the search ability of grey wolves. According to the simulation results in [54]-[55], RW-GWO shows high efficiency in solving both continuous and discrete optimization algorithms.

In summary, the computational complexity of most of the hybrid GWO algorithms (e.g. hybrid GA and GWO [29], [36], DE-GWO [43], EGWO [49], ALO-GWO [44]) is much higher than the computational complexity of the original GWO. This is because the integrated search method in GWO is normally repeated at each iteration of GWO. Moreover, hybrid GWO algorithms have complex structures compared to the basic GWO algorithm. Other enhanced versions of the GWO algorithm (e.g. EPD-GWO [45], GWO and Lévy flight operator [52], Dynamic GWO [46], wdGWO [48], EGWO [47]) provide insignificant enhancement in the performance compared to GWO or can be applied only to specific applications. In the next Section, a distributed variation of GWO (DGWO) is introduced in an attempt to enhance the diversity of GWO by organizing its population into small islands based on the island model. An interesting feature about DGWO is that it can run in parallel devices, which means that its computational complexity can be reduced compared to the computational complexity of hybrid GWO algorithms. However, DGWO can be directly applied to continuous optimization problems. DGWO should be tailored to be applicable to real-world discrete optimization problems.

3. DISTRIBUTED GREY WOLF OPTIMIZER

The Grey Wolf Optimizer (GWO) is a nature-inspired optimization algorithm that mimics the hunting strategy and leadership hierarchy of grey wolves [28]. GWO has been applied successfully to various continuous optimization problems [45], [48], [52] and discrete optimization problems [29], [36], [48], [52]. A problem with GWO is that its improvement loop may not maintain the diversity of its population due the imperfection of its evolutionary operators. Such a problem is a common problem with all optimization algorithms.

The island model enhances the performance and run-time of optimization algorithms. It also provides better chances for unfit solutions in each island to evolve and improve. The distributed genetic algorithm [56], distributed differential evolution [57]-[58], distributed particle swarm optimization [59] and distributed ant colony [60] algorithms are but few examples of successful island-based optimization algorithms.

The current section introduces a Distributed Grey Wolf Optimizer (DGWO) algorithm in an attempt to improve the diversity of GWO by organizing its population into small islands based on the island model. DGWO applies the original GWO to the population of each island and then allows selected solutions to be exchanged among the islands based on the random-ring topology and the best-worst migration policy.

Figure 2 shows the pseudo code of the DGWO algorithm. The first step of DGWO is to determine the total number of candidate solutions (n) for a given number of islands (s) and the maximum number of iterations (*MaxItr)* of DGWO. The second step is to initialize the parameters of the island model. The next step is to generate k candidate solutions for each island and then to calculate the number of migration waves (M_w) and number of migrant solutions (n_r).

DGWO generates k candidate solutions for each island before the beginning of the evolution process of DGWO. In DGWO, the evolution process of the basic GWO algorithm is synchronously applied to each individual island. After each M_f iterations (migration frequency), a number of candidate solutions are swapped between each two neighbouring islands based on the random-ring topology and the best-worst migration policy. In the random-ring topology, the neighbouring relationships are unidirectional relationships (Figure 3). However, the neighbouring relationships in the random-ring topology change after each migration wave among the islands. The number of candidate solutions to be exchanged among the islands is specified by the migration rate (M_r).

The migration process among islands takes place each time the maximum number of iterations that is specified by M_f is reached. Before the beginning of the migration process, the islands in DGWO are organized to form a unidirectional ring based on the principles of the random ring topology. The most fitted solutions in one island (say m solutions) are exchanged with the m worst fitted solutions in a neighbouring island based on the best-worst migration policy. Let $ppo_i = \{x_1^i, x_2^i, \dots, x_s^i\}$ and $ppo_j = \{x_1^j, x_2^j, \dots, x_s^j\}$ be the neighboring islands I_i and I_j, respectively. If we assume that R_m = 20%, n=150 and s= 5, the number of migrant solutions is $R_m \times (n/s) = 20\% \times (150/5) = 6$. Let ppo_i and ppo_j be two lists ordered in ascending order based on their objective values, where $f(x_1^i) \leq f(x_2^i) \leq \cdots \leq f(x_s^i)$ and $f(x_1^j) \leq f(x_2^j) \leq \cdots \leq f(x_s^j)$. If we assume that there is a unidirectional edge from I_i to I_j, the first

six candidate solutions in I_i $(x_1^i, x_2^i, x_3^i, x_4^i, x_5^i, x_6^i)$ will be exchanged with the last six solutions in I_j $\{x_{s-5}^j, x_{s-4}^j, x_{s-3}^j, x_{s-2}^j, x_{s-1}^j, x_s^j\}$.

Determine the total number of candidate solutions of all islands (n) and the maximum number of iterations (MaxItr).
Initialize the parameters of the island model (*number of islands (s), migration frequency (M_f), migration rate (M_r)*)
Calculate the population size for each island (k=n/s)
Calculate the number of migration waves (M_w= *MaxItr/ M_f*)
Calculate the number of migrant solutions ($n_r = n/s \times M_r$)
Initialize the population of k candidate solutions for each island $\vec{X}_i^j (i = 1,2,\dots,k)$ *and* $(j = 1,2,\dots,s)$
for $i = 1$ *to* M_w .....*(1)*
for $j = 1$ *to* s.....*(2)*
Initialize $\vec{A}^j$, $\vec{a}^j$ *and* $\vec{C}^j$
t=0
Calculate the fitness value of each candidate solution
$\vec{X}_\alpha^j$ *= the best candidate solution in island j*
$\vec{X}_\beta^j$ *= the second best candidate solution in island j*
$\vec{X}_\delta^j$ *= the third best candidate solution in island j*
While (*t< M_f*).....*(3)*
for *each candidate solution in island j .....(4)*
Update the values of the current candidate solution using Equation 11
end for
Update $\vec{A}^j$, $\vec{a}^j$ *and* $\vec{C}^j$
Calculate the fitness value of each candidate solution in island j
Update $\vec{X}_\alpha^j$,$\vec{X}_\beta^j$ *and* $\vec{X}_\delta^j$
Replace the nr best candidate solutions in island j with the n_r worst candidate solutions in island $((j+1) \bmod s)$
t=t+1
end while
end for
end for
Calculate $\vec{X}_\alpha$($\vec{X}_\alpha^j$ *with the best fitness)*
return the $\vec{X}_\alpha$

Figure 2. The Distributed Grey Wolf Optimizer (DGWO) Algorithm.

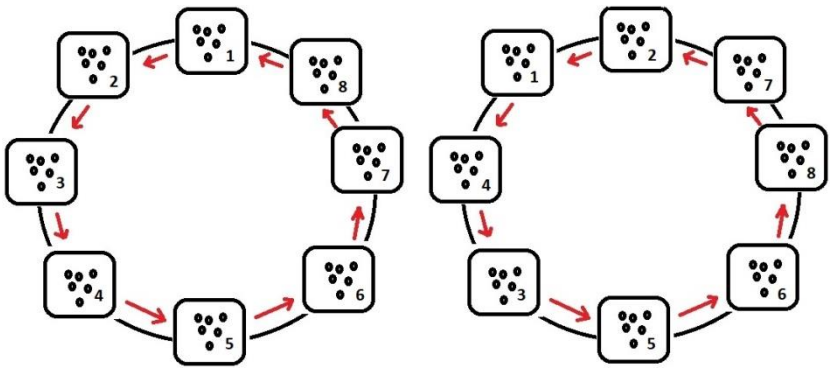


Figure 3. Random-ring Migration Topology.

The computational complexity of DGWO (Figure 2) can be calculated based on its main steps as follows:

1. The first five steps require 5 operations.
2. Initializing the population of n candidate solutions for all islands requires n operations.
3. Calculating the fitness of all candidate solutions requires n operations.
4. Calculating the first three best solutions in an island requires k operations.
5. The most inner for loop (number 4) that is used to update the population of an island requires k operations.
6. Calculating the fitness of each candidate solution in an island requires k operations.
7. Updating the first three best solutions in an island requires k operations.
8. Replacing the n_r best candidate solutions in an island with the n_r worst candidate solutions in another island requires $k+ k+ n_r$ operations, which can be simplified to k because $k > n_r$.
9. Overall, the while loop (number 3) requires $M_f.(k + k + k + n_r)$ operations. This step can be simplified to $M_f. k$, because $k > n_r$.
10. For loop number 2 requires $s.(k + k + M_f. k)$ operations, which can be simplified to $s.M_f.k$, because $s.M_f.k$ is larger than $s.k$.
11. For loop number 1 requires $M_w. s. M_f. k$ operations.
12. The last step that calculates the best candidate solution in all islands requires s operations.
13. All the operations of the algorithm can be calculated as $5+2n+k+M_w.s.M_f.k+s$, which can be simplified to $M_w.s.M_f.k$ operations.

In summary, the computational complexity of DGWO is $O(M_w.s.M_f.k)$. On the other hand, the computational complexity of the basic GWO is $O(m.n)$, where m is the maximum number of iterations and n is the number of candidate solutions. Thus, it is better to run DGWO in parallel devices which will reduce the computational complexity of DGWO to $O((M_w.s.M_f.k)/s)$, where s is the number of parallel devices (number of islands). In this case, the complexity of DGWO can be simplified to $O(M_w . M_f . k)$. Note that any basic vector operation has been assumed to cost O(1) in the above analysis.

4. EXPERIMENTS

In this section, the DGWO is benchmarked on 15 test functions (Table 1 and Table 2). These functions are standardtest functions used widely by the researchers to evaluate and compare the performance of swarm-bsed evolutionary algorithms [3]-[4], [28], [43], [48], [61]. Section 4.1 shows the experimental setup. Section 4.2 provides an analysis of the performance of DGWO based on different experimental

Table 1. Test functions.

Abbreviation	Function name	Range	D	$f(\vec{X}^*)$
f_1	Generalized Schwefel's Problem 2.26	[-500,500]	30	$-418.983 \times D$
f_2	Griewank's Function	[-10,10]	30	0
f_3	Whitley's Function	[-10,10]	30	0
f_4	Ackley's Function 2.9	[32.768, 32.768]	30	0
f_5	Alpine's Function	[-10,10]	30	0
f_6	Schaffer's Function	[-100,100]	2	0
f_7	Rastrigin's Function 2.5	[-5.12, 5.12]	30	0
f_8	Inverted Cosine Wave Function	[-5, 5]	30	-D+1
f_9	Levy Function	[-10, 10]	30	0
f_{10}	Schwefel's 2.22 Function	[-100,100]	30	0
f_{11}	Rotated Hyper-ellipsoid Function	[-65.536, 65.536]	30	0
f_{12}	Shifted Sphere Function	[-100,100]	30	0
f_{13}	Shifted Schwefel Function	[-100,100]	30	0
f_{14}	Shifted Rastrigin's Function	[-5, 5]	30	-330
f_{15}	Shifted Expanded Griewank's Plus	[-5, 5]	30	-130

scenarios. Section 4.3 provides analysis about the performance of DGWO compared to recently proposed optimization algorithms (Grey Wolf Optimizer (GWO) [28], Cuckoo search (CS) [3], adaptive differential evolution with linear population size reduction evolution (L-SHADE) [62], memory-based hybrid Dragonfly (MHDA) [63] and Fireworks algorithm with differential mutation (FWA-DM) [64].

Table 2. Mathematical formulae of test functions.

Formula
$f_1(X) = -\sum_{i=1}^{n}[x_i \sin(\sqrt{\lvert x_i \rvert})]$
$f_2(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
$f_3(x_1 \cdots x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} (\frac{(100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2}{4000} - cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1)$
$f_4(X) = -20\, e^{(-0.2\times\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)]} + 20 + e^{(1)}$
$f_5(X) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$
$f_6(x, y) = 0.5 + \frac{sin^2(x^2 + y^2)^2 - 0.5}{(1 + 0.001(x^2 + y^2))^2}$
$f_7(X) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$
$f_8(X) = -\sum_{i=1}^{n-1}\{ e^{\left[\frac{-(x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1})}{8}\right]} \cos(4 \times \sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}})\}$
$f_9(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + sin^2(2\pi w_d)]$,   where $w_i = 1 + \frac{x_i - 1}{4}$, for all i = 1, 2, ..., d
$f_{10}(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
$f_{11}(X) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
$f_{12}(X) == \sum_{i=1}^{D} z_i^2 + f_{bias} = \pi$,   where ***z=X-o*** and f_{bias}=-450
$f_{13}(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2 + f\_bias$,   where ***z=X-o*** and f_{bias}=-450
$f_{14}(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias$,   where ***z=X-o*** and f_{bias}=-330
The definition of f_{15} is given in [66],   where ***z=X-o*** f_{bias}=-130

4.1 Experimental Setup

In each algorithm, the maximum number of iterations was 10,000 and the size of population was 30. For the GWO and DGWO algorithms, the values of vector $\vec{a}$ were linearly decreased form 2 to 0 over the course of their simulation process. The parameters of the island model in DGWO were tested for different values as shown in Table 3. The parameters of each test function are shown in Tables 1 and 2. The percentage of abandonment in CSwas 25% as suggested in [4], [66]. The parameters of L-SHADE (external archive size, historical memory size, control parameter) were dynamically tuned as in [62]. The parameter setting of MHDA was taken from refernce [63]. The parameters of FWA-DM were set as follows: A_{init} (initial amplitude)= 0.2, A_{final} (final amplitude)= 0.001, F (scaling factor)= 0.5 and CR (Crossover Operator)= 0.9 as in [64].

4.2 Analysis of the Parameters of DGWO

Table 3 shows nine experimental scenarios used to investigate the relationships between the performance of DGWO and the parameters of the island model (s, M_f, M_r). The purpose of the first three scenarios is to investigate the relationship between the island number s and the performance of DGWO. Scenarios 3-6 aim to investigate the influence of the migration frequency M_f on the performance of DGWO. The purpose of the last three scenarios is to study the influence of the migration rate M_r on the performance of DGWO.

Table 3. Scenarios for analyzing the parameters of DGWO.

Scenario	***s***	M_f	M_r%
Scenario 1	2	100	20
Scenario 2	5	100	20
Scenario 3	10	100	20
Scenario 4	10	50	20
Scenario 5	10	100	20
Scenario 6	10	500	20
Scenario 7	10	50	10
Scenario 8	10	50	20
Scenario 9	10	50	30

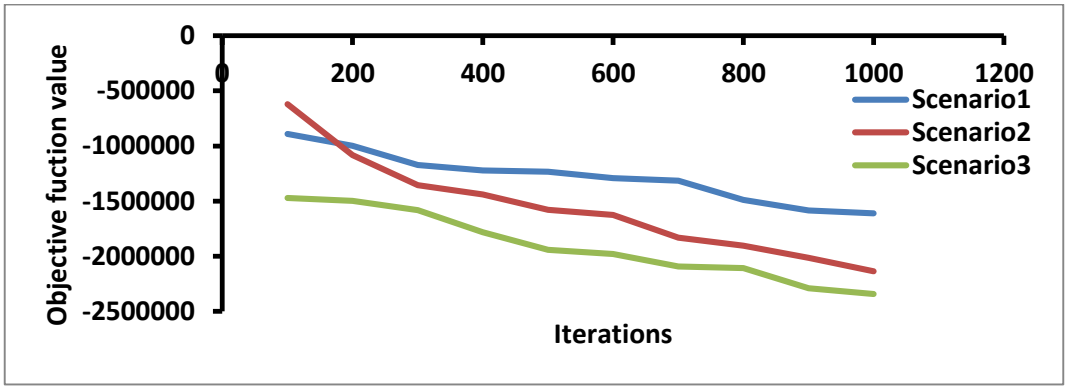
Tables 4-6 show the simulation results of the nine experimental scenarios ilustrated in Table 3. The best results in the tables (lowest objective values) are highlighted with **bold**. The results were averged over 50 independent runs.

Table 4. Simulation results of the algorithms for 15 test functions, D=30, runs=50, iterations=10,000 (Scenario 1, Scenario 2, Scenario 3).

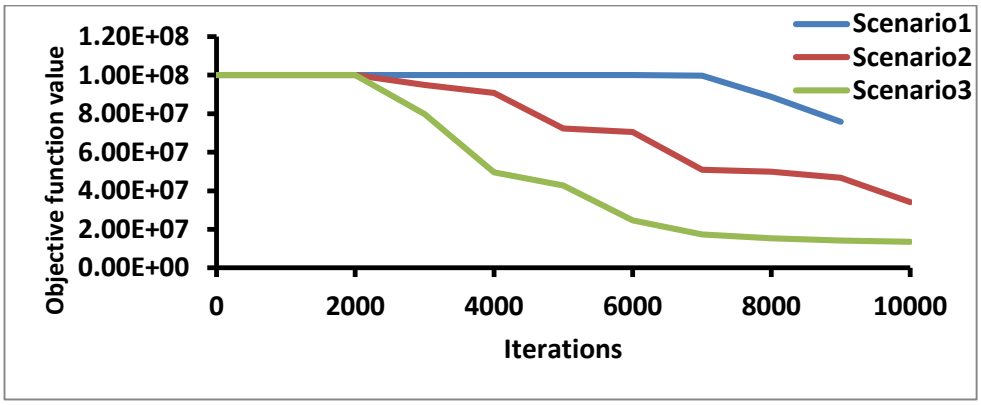
Function	**Scenario 1** ***s* = 2**	**Scenario 2** ***s* = 5**	**Scenario 3** ***s* = 10**
f_1	-1.56E+06	-2.08E+06	**-2.10E+06**
f_2	0.00E+00	0.00E+00	0.00E+00
f_3	4.57E+01	4.52E+01	**4.49E+01**
f_4	1.98E+01	**4.44E-16**	**4.44E-16**
f_5	6.44E-220	**1.97E-222**	1.64E-210
f_6	5.00E-01	5.00E-01	5.00E-01
f_7	0.00E+00	0.00E+00	0.00E+00
f_8	-9.00E+00	-9.00E+00	-9.00E+00
f_9	1.32E+00	1.32E+00	**8.52E-01**
f_{10}	7.27E+03	1.15E+04	**4.21E+03**
f_{11}	8.17E+07	5.27E+07	**1.36E+07**
f_{12}	1.00E+08	1.00E+08	1.00E+08
f_{13}	1.00E+08	1.00E+08	1.00E+08
f_{14}	-2.33E+02	-2.38E+02	**-2.40E+02**
f_{15}	2.49E+04	3.52E+03	**3.15E+03**

Table 4 shows the simulation results of DGWO under different island sizes (s= 2, s=5, s=10). The results in the table show that DGWO in scenario 3 (s=10) performed better than in scenario 1 (s=2) and scenario 2 (s=5). The results clearly indicate that the performance of DGWO improves with an increase in the number of islands. This is expected, because the existence of many islands (large value of s) allows different search regions to be explored simultaneously, while the existence of few islands (low value of s) means that few number of search regions can be explored simultaneously [31]-[32]. In addition, the population in a small island would most likely converge to suboptimal solutions earlier than expected [42]. It is worth pointing out that choosing a large value of s increases the computational complexity of DGWO. For example, if the number of islands is 10, then GWO will be repeated 10 times at each iteration of DGWO. Thus, it is better to run DGWO in parallel devices to reduce its computational complexity. Based on the results in Table 4, s=10 was used in Tables 5 and 6.

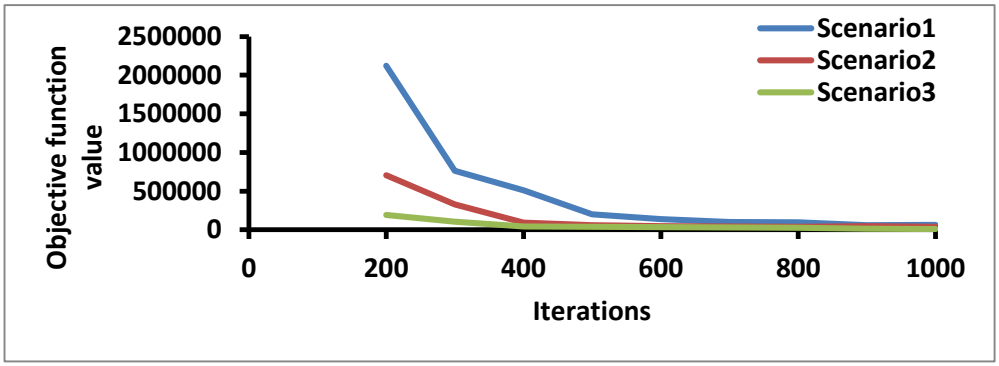
The convergence curves of the first three scenarios of DGWO for three functions (f_1, f_{11}, f_{15}) are shown in Figure 4. It is obvious that convergence increases with the increase in iterations for all functions. Figure 4(a), Figure 4(b) and Figure 4(c) show that DGWO in scenario 3 is the fastest converging algorithm.



(a) f_1



(b) f_{11}



(c) f_{15}

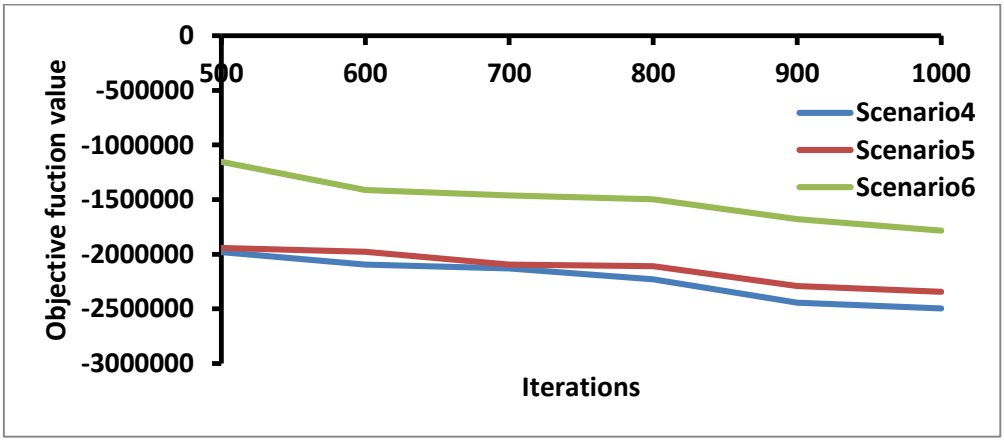
Figure 4. Convergence Plots of DGWO for Scenarios 1-3.

Table 5 shows the simulation results of DGWO under different migration frequencies ($M_f = 50$, $M_f = 100$, $M_f = 500$). It is obvious that DGWO in scenario 4 ($M_f = 50$) performed the best followed by DGWO in scenario 2 ($M_f = 100$) and then DGWO in scenario 3 ($M_f = 500$). These results suggest that low migration frequencies improve the performance of DGWO compared to high migration frequencies. Basically, low migration frequencies provide more chances for a reasonable number of candidate solutions from the source island to move to the destination island with limited effect on the candidate solutions in the destination island. Consequently, the likelihood that convergence will take longer time to occur increases [31]-[32]. Based on the illustrated results in Table 5, $M_f = 50$ was used in Table 6.

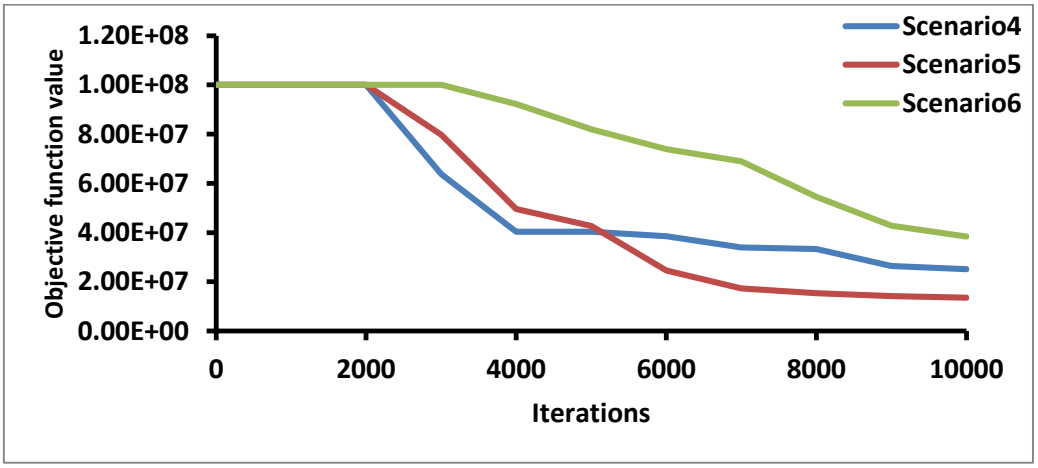
Table 5. Simulation results of the algorithms for 15 test functions, D=30, runs=50, iterations=10,000 (Scenario 4, Scenario 5, Scenario 6).

Function	Scenario 4 (M_f = 50)	Scenario 5 (M_f = 100)	Scenario 6 (M_f = 50)
f_1	**-2.27E+06**	-2.10E+06	-2.13E+06
f_2	0.00E+00	0.00E+00	0.00E+00
f_3	4.49E+01	4.49E+01	**4.47E+01**
f_4	4.44E-16	4.44E-16	4.44E-16
f_5	2.38E-224	**1.64E-221**	2.74E-218
f_6	5.00E-01	5.00E-01	5.00E-01
f_7	0.00E+00	0.00E+00	0.00E+00
f_8	-9.00E+00	-9.00E+00	-9.00E+00
f_9	**7.26E-01**	8.52E-01	1.57E+03
f_{10}	4.90E+03	**4.21E+03**	9.74E+03
f_{11}	1.40E+07	**1.36E+07**	7.76E+07
f_{12}	**9.49E+06**	1.00E+08	1.00E+08
f_{13}	1.00E+08	1.00E+08	1.00E+08
f_{14}	**-2.60E+02**	-2.40E+02	-2.44E+02
f_{15}	**1.10E+03**	3.15E+03	9.01E+03

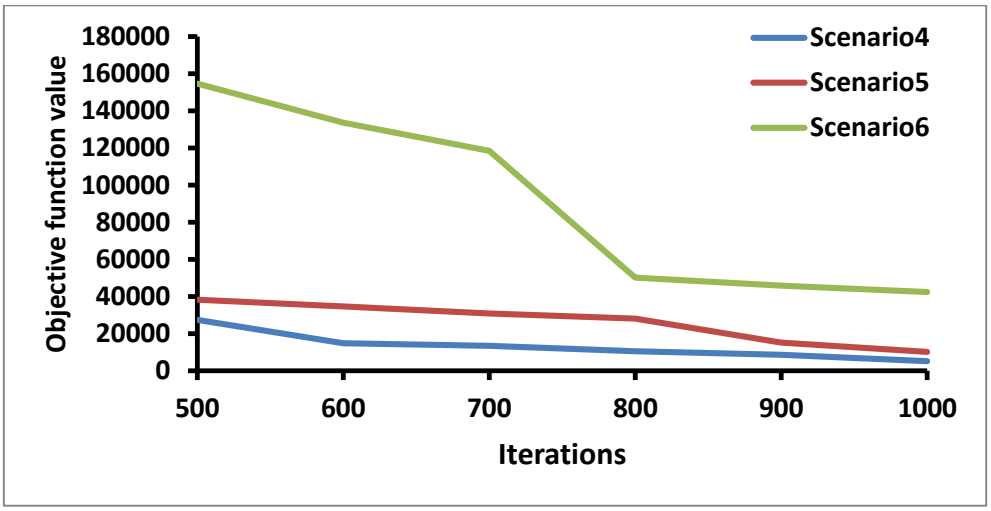
Figure 5 shows the convergence curves of the second three scenarios of DGWO for three functions (Figure 5(a) (f_1,) Figure 5(b) (f_{11}), Figure 5(c) (f_{15})). Figure 5(a) and Figure 5(c) show that DGWO in scenario 4 converges faster than the other algorithms, while Figure 5(b) shows that DGWO in scenario 5 converges faster to a solution compared to the other algorithms.



(a) f_1



(b) f_{11}



(c) f_{15}

Figure 5. Convergence Plots of DGWO for Scenarios 4-6.

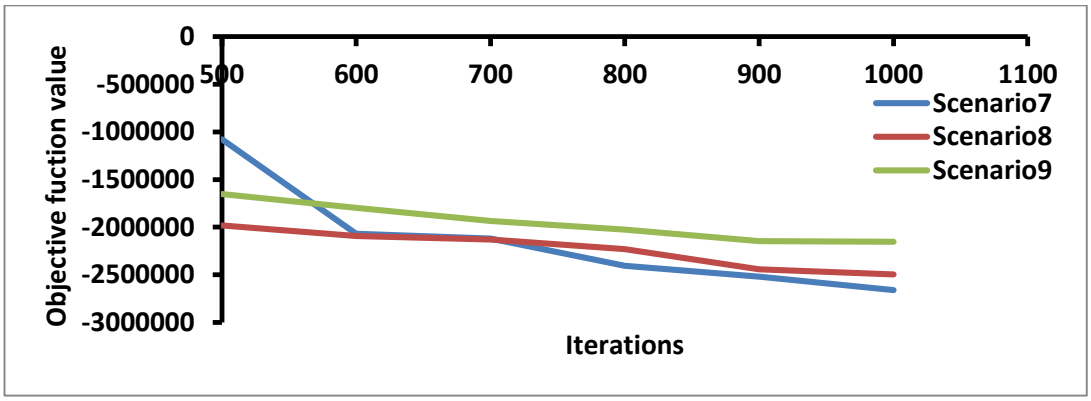
Table 6 shows the simulation results of DGWO under different migration rates (M_r = 10%, M_r = 20%, M_r= 30%). It can be clearly observed that DGWO in scenario 8 (M_r = 20%) performed better than DGWO in scenario 7 (M_r = 10%) and DGWO in scenario 9 (M_r = 30%). These results suggest that there is no clear indication whether high values or low values of M_r improve the performance of DGWO. A possible explanation for the results is that replacing a large number of candidate solutions in an island has an unclear effect on the diversity of its population, while replacing few candidate solutions in an island can act as a seed to enhance its diversity but with a limited effect [31]-[32].

Table 6. Simulation results of the algorithms for 15 test functions, D=30, runs=50, iterations=10,000 (Scenario 7, Scenario 8, Scenario 9).

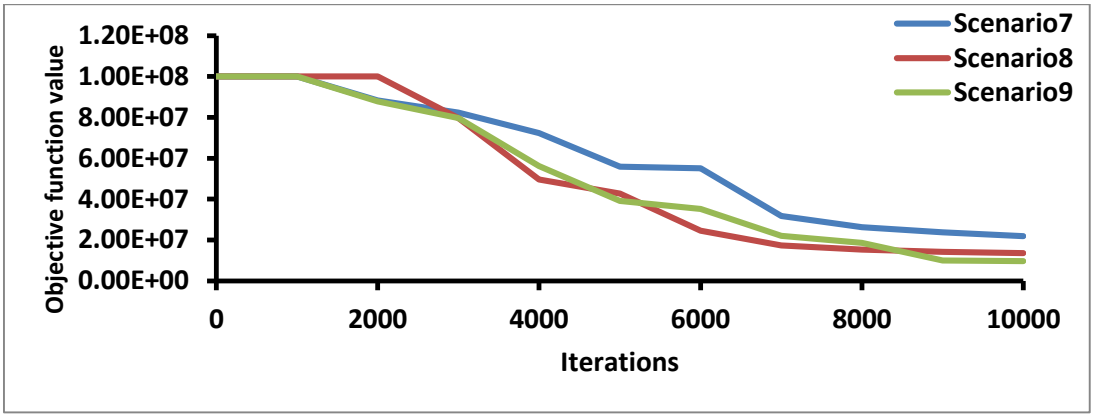
Function	**Scenario 7 M_r = 10%**	**Scenario 8 M_r = 20%**	**Scenario 9 M_r = 30%**
f_1	**-2.75E+06**	-2.27E+06	-2.19E+06
f_2	0.00E+00	0.00E+00	0.00E+00
f_3	4.26E+01	4.49E+01	**4.12E+01**
f_4	4.44E-16	4.44E-16	4.44E-16
f_5	2.24E-219	**2.38E-224**	1.83E-223
f_6	5.00E-01	5.00E-01	5.00E-01
f_7	0.00E+0	0.00E+0	0.00E+0
f_8	-9.0E+00	-9.0E+00	-9.0E+0
f_9	8.52E-01	**7.26E-01**	1.32E+0
f_{10}	1.72E+04	**4.90E+03**	1.65E+04
f_{11}	2.91E+07	1.40E+07	**3.15E+06**
f_{12}	1.00E+08	**9.49E+06**	4.52E+07
f_{13}	1.00E+08	1.00E+08	1.00E+08
f_{14}	-2.54E+02	**-2.6E+02**	-2.46E+02
f_{15}	**1.00E+03**	1.10E+03	1.14E+03

Figure 6 shows the convergence curves of the last three scenarios of DGWO for three functions (Figure 6(a) (f_1,) Figure 6(b) (f_{11}), Figure 6(c) (f_{15})). Figure 6(a) and Figure 6(c) show that DGWO in scenario 7 converges faster than the other algorithms, while Figure 6(b) shows that DGWO in scenario 9 converges faster to a solution compared to the other algorithms. Note that DGWO in Scenario 8 achieved the second best convergence speed for the three functions.

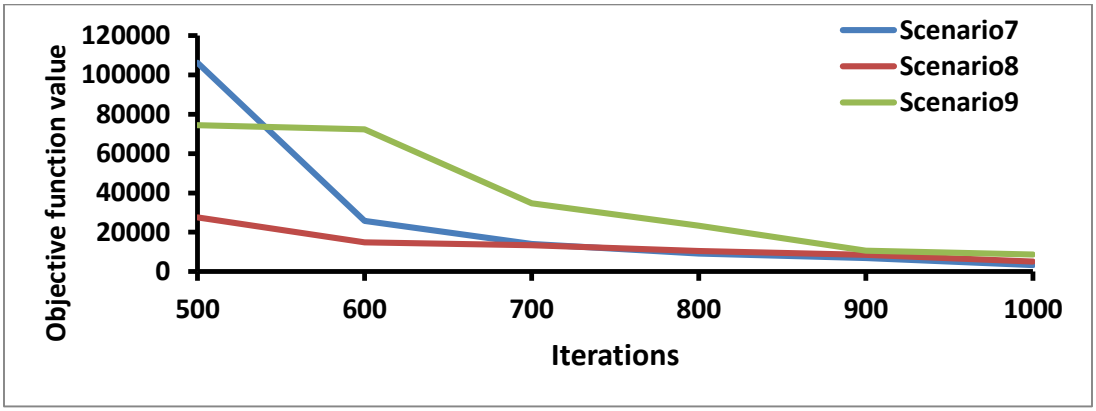
In conclusion, the overall experimental results in this section indicate that high values of s and low values of M_f improve the performance of DGWO. However, there is no clear indication whether high values or low values of M_r improve the performance of DGWO.



(a) f_1



(b) f_{11}



(c) f_{15}

Figure 6. Convergence Plots of DGWO for Scenarios 7-9.

4.3 Comparison among DGWO and Other Algorithms

The single-objective real-parameter optimization-benchmark suit of CEC2014 is composed of 30 functions (Table 7). This suit represents an approximation of real-world optimization problems. The search range of each function in the suit is $[-100.100]^D$. More details about these functions are available in [67].

Using the single-objective real-parameter optimization-benchmark suit of CEC2014 with 30 dimensions (30 decision variables) [67], the performance of DGWO (Scenario 8) was compared with the performance of GWO and recently proposed optimization algorithms: Cuckoo search (CS), adaptive differential evolution with linear population size reduction evolution (L-SHADE), memory-based hybrid Dragonfly algorithm (MHDA) and Fireworks algorithm with differential mutation (FWA-DM).

Table 7. Single-objective real-parameter optimization-benchmark suit of CEC2014.

Function	**Function Type**
f_1- f_3	Unimodal functions
f_4- f_{16}	Multimodal functions
f_{17}- f_{22}	Hybrid functions
f_{23}- f_{30}	Composite functions

Table 8 shows the function error value (FEV) for the 30CEC2014 functions. The FEVis the distance between theaverage of best objective values found in all runs and the true optimal value. Note that the lowest FEV for each function (best result) is marked with **Bold**. The simulation results in Table 8show that L-SHADE ouperforms the other optimization algorithms by providing the lowest FEV for 11 functions of the 30 functions. This is expected, because L-SHADE is a dynamic differential evolution algorithm thatdynamically adjustsitsinternal parameters and population size over the course of its iterations. Interestingly, DGWO is the second best performing optimization algorithm by producing the lowest FEV for 10 functions of the 30 test functions. This is because DGWO synchronously applies GWO to multiple islands, which accelerates its convergence to a good solution.

Table 8. Simulation results of DGWO compared to five optimization algorithms. D= 30, runs=50, number of iterations is 10,000.

Function	**GWO**	**DGWO (Scenario 8)**	**CS**	**L-SHADE**	**MHDA**	**FWA-DM**
f_1	2.00E+03	4.36E+00	3.47E + 07	**9.00E-15**	3.59E+03	4.91E+05
f_2	2.12E+03	2.36E+00	2.50E + 07	8.50E-11	3.82E+03	**2.50E-16**
f_3	2.89E-01	2.54E-04	4.10E + 04	5.83E-10	5.80E-07	**1.88E-16**
f_4	8.75E-03	**1.63E-09**	4.22E+02	2.58E-09	1.42E-08	2.23E+01
f_5	3.96E+02	2.00E+02	5.00E+01	2.00E+01	**2.36E+00**	2.11E+01
f_6	5.19E+01	1.21E+00	3.63E+01	1.25E-06	**8.52E-14**	1.82E+01
f_7	2.89E-03	8.53E-10	1.86E+00	7.25E-09	**2.25E-11**	2.53E-03
f_8	1.33E+00	**1.51E-19**	3.89E+02	1.25E-09	2.20E-19	9.53E-15
f_9	1.82E+01	**1.03E+00**	3.00E+03	8.96E+00	5.30E+00	6.54E+01
f_{10}	9.79E+00	**3.20E-03**	4.37E+03	2.36E-02	1.22E+03	1.13E+01
f_{11}	1.99E+04	2.95E+03	4.00E+03	2.30E+03	**1.52E+02**	2.19E+03
f_{12}	8.50E+00	**6.30E-02**	4.78E-01	9.00E-01	1.42E-01	3.25E-01
f_{13}	2.19E+00	4.59E-01	4.81E-01	6.50E-01	4.78E-01	**3.11E-01**
f_{14}	2.35E-01	**1.99E-01**	4.28E-01	8.60E-01	5.43E-01	2.99E-01
f_{15}	1.01E+02	7.23E+01	9.94E+01	**1.60E+00**	3.25E+00	8.36E+00
f_{16}	1.90E+01	**9.53E+00**	1.53E+01	1.02E+01	1.06E+01	1.10E+01
f_{17}	1.66E+02	4.55E+03	3.47E+06	**2.20E+00**	4.53E+02	6.59E+03
f_{18}	8.77E+00	3.94E+01	3.90E+03	**1.90E+00**	3.69E+00	7.24E+01
f_{19}	4.96E+01	1.22E+02	6.14E+01	**5.30E+00**	3.78E+02	1.04E+01
f_{20}	6.20E+01	4.73E+02	3.97E+04	**4.30E+00**	7.09E+02	4.37E+01
f_{21}	1.04E+03	7.09E+02	3.57E+05	3.69E+02	**2.57E+02**	8.75E+02
f_{22}	2.42E+02	2.73E+02	9.47E+02	**1.32E+02**	2.73E+02	1.62E+02
f_{23}	3.65E+02	**3.69E+01**	3.78E+02	3.26E+02	3.10E+03	3.16E+02
f_{24}	2.24E+02	2.25E+02	2.89E+02	**1.93E+02**	2.26E+02	2.96E+02
f_{25}	2.45E+02	2.11E+02	3.26E+02	**2.00E+02**	2.11E+02	2.09E+02
f_{26}	3.29E+02	2.10E+02	2.22E+02	2.69E+02	1.00E+02	**9.93E+01**
f_{27}	2.95E+02	4.09E+02	5.22E+02	**1.26E+02**	4.05E+02	4.10E+02
f_{28}	5.36E+02	1.65E+03	3.86E+03	**3.62E+02**	1.54E+03	4.22E+02
f_{29}	2.39E+02	**2.29E+02**	2.59E+05	7.33E+02	7.86E+02	2.78E+02
f_{30}	3.32E+02	**2.83E+00**	2.39E+04	6.99E+02	2.63E+03	4.69E+02

This indicates that DGWO performs well compared to powerful optimization algorithms. Note that GWO and CS have the worst performance compared to the other algorithms. A possible explanation is that CS and GWO do not employ any special convergence-enhancement technique compared to the other tested algorithms.

5. CONCLUSIONS

The current paper presented the Distributed Grey Wolf Optimizer (DGWO) algorithm that is based on a distribution model called the island model. The population in DGWO is divided into small populations in an attempt to enhance the diversity of the population and the run-time of the algorithm. DGWO applies the original GWO to the population of each island and then allows selected solutions to be exchanged among the islands based on the random ring topology and the best-worst migration policy.

Different experimental cases were designed and used to study the sensitivity of the performance of DGWO to the parameters of the island model (number of islands s, migration frequency M_f and migration rate M_r). The overall experimental results suggest that high values of s and low values of M_f significantly improve the performance of DGWO. However, there is no clear indication whether high values or low values of M_r improve the performance of DGWO.

In addition, 30 functions of CEC2014 (real-parameter single-objective optimization-benchmark suit) have been used to compare the performance of DGWO to the performance of well-known optimization algorithms (CS, L-SHADE, MHDA, FWA-DM). The results indicate that DGWO produces competitive results compared to those produced by the other compared algorithms. Interestingly, DGWO produced the lowest FEV for 10 functions of the 30 test functions of CEC2014. This is expected, because DGWO synchronously applies GWO to several islands, which accelerates its convergence to good solutions. Moreover, DGWO provides better chances for unfit candidate solutions in each island to evolve to better candidate solutions.

There are four interesting directions for future work. First, it would be interesting to incorporate the island model to multi-objective discrete GWO [38] to explore its efficiency in solving the scheduling problem in welding production. This scheduling problem is one of the most time-consuming processes in modern industry. Second, a binary version of DGWO will be developed and used to solve the problem of feature selection [35], [68]. Feature selection is normally considered as a complex time-consuming problem when it is used with large datasets. Third, hierarchical Q-learning [69]-[70] and cooperative Q-learning [71]-[72] require heavy and complex computations to efficiently solve learning problems with large state space or action space. Based on the fact that the population of Q-values (i.e., values of state-action pairs in Q-learning) in Q-learning can be represented as an optimization problem [1], [17], [66] and [71], the DGWO algorithm will be applied to hierarchical Q-learning [69]-[70] and cooperative Q-learning [71]-[72] as discussed in [17], [66]. Finally, the experimental results in Section 4.3 demonstrated that L-SHADE [62] performs better than many powerful optimization algorithms. Therefore, the incorporation of the island model into the L-SHADE algorithm will be addressed in a future study in an attempt to elevate the performance of L-SHADE.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Oguz Emrah Turgut for the source code of GWO that he made publically, [Online], available at:
https://www.researchgate.net/publication/281939423_Grey_Wolf_Optimizer_java_code.

REFERENCES

[1] B. H. Abed-alguni, D. J. Paul, S. K. Chalup and F. A. Henskens, "A Comparison Study of Cooperative Q-learning Algorithms for Independent Learners," Int. J. Artif. Intell., vol. 14, no. 1, pp. 71-93, 2016.

[2] X.-S. Yang, "A New Metaheuristic Bat-inspired Algorithm," Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, pp. 65-74, 2010.

[3] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy Flights," IEEE World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 210-214, 2009.

[4] B. H. Abed-alguni and F. Alkhateeb, "Novel Selection Schemes for Cuckoo Search," Arabian Journal for Science and Engineering, vol. 42, no. 8, pp. 3635-3654, 2017.

[5] F. Alkhateeb and B. H. Abed-alguni, "A Hybrid Cuckoo Search and Simulated Annealing Algorithm," Journal of Intelligent Systems, 2017, [Online], Available: https://doi.org/10.1515/jisys-2017-0268.

[6] B. H. Abed-alguni and F. Alkhateeb, "Intelligent Hybrid Cuckoo Search and β-hill Climbing Algorithm," Journal of King Saud University - Computer and Information Sciences, pp. 1-44, 2018, [Online], Available: https://doi.org/10.1016/j.jksuci.2018.05.003.

[7] B. H. Abed-alguni and A. F. Klaib, "Hybrid Whale Optimization and β-hill Climbing Algorithm," International Journal of Computing Science and Mathematics, pp. 1-13, 2018.

[8] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51-67, 2016.

[9] G. Kaur and S. Arora, "Chaotic Whale Optimization Algorithm," Journal of Computational Design and Engineering, vol. 5, no. 3, pp. 275-284 , July 2018.

[10] S. Arora and P. Anand, "Learning Automata Based Butterfly Optimization Algorithm for Engineering Design Problems," International Journal of Computational Materials Science and Engineering, July 2018.

[11] S. Arora and S. Singh, "Butterfly Optimization Algorithm: A Novel Approach for Global Optimization," Soft Computing, pp. 1-20, 2018.

[12] S. Arora and S. Singh, "A Hybrid Optimization Algorithm Based on Butterfly Optimization Algorithm and Differential Evolution," International Journal of Swarm Intelligence, vol. 3, no. 2-3, pp. 152-169, 2017.

[13] S. Arora and S. Singh, "An Improved Butterfly Optimization Algorithm for Global Optimization," Advanced Science, Engineering and Medicine, vol. 8, no. 9, pp. 711-717, 2016.

[14] S. Arora, S. Singh and K. Yetilmezsoy, "A Modified Butterfly Optimization Algorithm for Mechanical Design Optimization Problems," Journal of the Brazilian Society of Mechanical Sciences and Engineering, vol. 40, no. 1, p. 21, 2018.

[15] S. Arora and P. Anand, "Chaotic Grasshopper Optimization Algorithm for Global Optimization," Neural Computing and Applications, pp. 1-21, 2018.

[16] S. Arora and P. Anand, "Chaos-enhanced Flower Pollination Algorithms for Global Optimization," Journal of Intelligent & Fuzzy Systems, vol. 33, no. 6, pp. 3853-3869, 2017.

[17] B. H. Abed-alguni, "Bat Q-learning Algorithm," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 3, no. 1, pp. 56-77, 2017.

[18] S. J. Mousavirad and H. Ebrahimpour-Komleh, "Multilevel Image Thresholding Using Entropy of Histogram and Recently Developed Population-based Metaheuristic Algorithms," Evolutionary Intelligence, vol. 10, no. 1-2, pp. 45-75, 2017.

[19] S. Pare, A. Bhandari, A. Kumar and G. Singh, "Rényi's Entropy and Bat Algorithm Based Color Image Multilevel Thresholding," Machine Intelligence and Signal Analysis: Springer, pp. 71-84, 2019.

[20] R.-E. Precup, R.-C. David, A.-I. Szedlak-Stinean, E. M. Petriu and F. Dragan, "An Easily Understandable Grey Wolf Optimizer and Its Application to Fuzzy Controller Tuning," Algorithms, vol. 10, no. 2, p. 68, 2017.

[21] J. Vaščák, "Adaptation of Fuzzy Cognitive Maps by Migration Algorithms," Kybernetes, vol. 41, no. 3/4, pp. 429-443, 2012.

[22] T. Jayabarathi, T. Raghunathan and A. Gandomi, "The Bat Algorithm, Variants and Some Practical Engineering Applications: A Review," Nature-Inspired Algorithms and Applied Optimization: Springer, pp. 313-330, 2018.

[23] S. K. Sarangi, R. Panda, P. K. Das and A. Abraham, "Design of Optimal High Pass and Band Stop FIR Filters Using Adaptive Cuckoo Search Algorithm," Engineering Applications of Artificial Intelligence, vol. 70, pp. 67-80, 2018.

[24] R.-E. Precup, R.-C. David and E. M. Petriu, "Grey Wolf Optimizer Algorithm-based Tuning of Fuzzy Control Systems with Reduced Parametric Sensitivity," IEEE Transactions on Industrial Electronics, vol. 64, no. 1, pp. 527-534, 2017.

[25] N. Jayakumar, S. Subramanian, S. Ganesan and E. Elanchezhian, "Grey Wolf Optimization for Combined Heat and Power Dispatch with Cogeneration Systems," International Journal of Electrical Power & Energy Systems, vol. 74, pp. 252-264, 2016.

[26] M. Nouiri, A. Bekrar, A. Jemai, S. Niar and A. C. Ammari, "An Effective and Distributed Particle Swarm Optimization Algorithm for Flexible Job-Shop Scheduling Problem," Journal of Intelligent Manufacturing, vol. 29, no. 3, pp. 603-615, 2018.

[27] M. K. Marichelvam and M. Geetha, "Cuckoo Search Algorithm for Solving Real Industrial Multi-Objective Scheduling Problems," Encyclopedia of Information Science and Technology, 4th Edition: IGI Global, pp. 4369-4381, 2018.

[28] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46-61, 2014.

[29] M. A. Tawhid and A. F. Ali, "A Hybrid Grey Wolf Optimizer and Genetic Algorithm for Minimizing Potential Energy Function," Memetic Computing, vol. 9, no. 4, pp. 347-359, 2017.

[30] W. Gai, C. Qu, J. Liu and J. Zhang, "An Improved Grey Wolf Algorithm for Global Optimization," 2018Chinese Control and Decision Conference (CCDC), pp. 2494-2498, 2018.

[31] M. A. Al-Betar and M. A. Awadallah, "Island Bat Algorithm for Optimization," Expert Systems with Applications, vol. 107, pp. 126-145, 2018.

[32] M. A. Al-Betar, M. A. Awadallah, A. T. Khader and Z. A. Abdalkareem, "Island-based Harmony Search for Optimization Problems," Expert Systems with Applications, vol. 42, no. 4, pp. 2026-2035, 2015.

[33] A. L. Corcoran and R. L. Wainwright, "A Parallel Island Model Genetic Algorithm for the Multiprocessor Scheduling Problem," Proceedings of the 1994 ACM Symposium on Applied Computing, pp. 483-487, 1994.

[34] E. Emary and H. M. Zawbaa, "Impact of Chaos Functions on Modern Swarm Optimizers," PLOS One, vol. 11, no. 7, p. e0158738, 2016.

[35] E. Emary, H. M. Zawbaa and A. E. Hassanien, "Binary Grey Wolf Optimization Approaches for Feature Selection," Neurocomputing, vol. 172, pp. 371-381, 2016.

[36] T. Jayabarathi, T. Raghunathan, B. R. Adarsh and P. N. Suganthan, "Economic Dispatch Using Hybrid Grey Wolf Optimizer," Energy, vol. 111, pp. 630-641, 2016.

[37] M. Pradhan, P. K. Roy and T. Pal, "Grey Wolf Optimization Applied to Economic Load Dispatch Problems," International Journal of Electrical Power & Energy Systems, vol. 83, pp. 325-334, 2016.

[38] C. Lu, S. Xiao, X. Li and L. Gao, "An Effective Multi-objective Discrete Grey Wolf Optimizer for a Real-world Scheduling Problem in Welding Production," Advances in Engineering Software, vol. 99, pp. 161-176, 2016.

[39] G. Komaki and V. Kayvanfar, "Grey Wolf Optimizer Algorithm for the Two-stage Assembly Flow Shop Scheduling Problem with Release Time," Journal of Computational Science, vol. 8, pp. 109-120, 2015.

[40] M. Ruciński, D. Izzo and F. Biscani, "On the Impact of the Migration Topology on the Island Model," Parallel Computing, vol. 36, no. 10, pp. 555-571, 2010.

[41] M. Tomassini, Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time, Springer, 2006.

[42] M. Tomassini, "Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series), Secaucus," Ed: NJ, USA: Springer-Verlag New York, Inc, 2005.

[43] D. Jitkongchuen, "A Hybrid Differential Evolution with Grey Wolf Optimizer for Continuous Global Optimization," IEEE 7th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 51-54, 2015.

[44] H. M. Zawbaa, E. Emary, C. Grosan and V. Snasel, "Large-dimensionality Small-instance Set Feature Selection: A Hybrid Bio-inspired Heuristic Approach," Swarm and Evolutionary Computation, vol. 42, pp. 29-42, 2018.

[45] S. Saremi, S. Z. Mirjalili and S. M. Mirjalili, "Evolutionary Population Dynamics and Grey Wolf Optimizer," Neural Computing and Applications, vol. 26, no. 5, pp. 1257-1263, 2015.

[46] L. Rodríguez, O. Castillo and J. Soria, "A Study of Parameters of the Grey Wolf Optimizer Algorithm for Dynamic Adaptation with Fuzzy Logic," Nature-Inspired Design of Hybrid Intelligent Systems: Springer, pp. 371-390, 2017.

[47] H. Joshi and S. Arora, "Enhanced Grey Wolf Optimization Algorithm for Constrained Optimization Problems," International Journal of Swarm Intelligence, vol. 3, no. 2-3, pp. 126-151, 2017.

[48] M. R. S. Malik, E. R. Mohideen and L. Ali, "Weighted Distance Grey Wolf Optimizer for Global Optimization Problems," IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1-6, 2015.

[49] E. A. Emary, H. M. A. Zawbaa and C. A. Grosan, "Experienced Grey Wolf Optimizer through Reinforcement Learning and Neural Networks," vol. 29, no. 3, pp. 681-694, 2018.

[50] H. Joshi and S. Arora, "Enhanced Grey Wolf Optimization Algorithm for Global Optimization," Fundamenta Informaticae, vol. 153, no. 3, pp. 235-264, 2017.

[51] M. Kohli and S. Arora, "Chaotic Grey Wolf Optimization Algorithm for Constrained Optimization Problems," Journal of Computational Design and Engineering, vol. 5, no. 4, pp. 458-472,2017.

[52] A. A. Heidari and P. Pahlavani, "An Efficient Modified Grey Wolf Optimizer with Lévy Flight for Optimization Tasks," Applied Soft Computing, vol. 60, pp. 115-134, 2017.

[53] M. A. Al-Betar, I. A. Doush, A. T. Khader and M. A. Awadallah, "Novel Selection Schemes for Harmony Search," Applied Mathematics and Computation, vol. 218, no. 10, pp. 6095-6117, 2012.

[54] S. Gupta and K. Deep, "A Novel Random Walk Grey Wolf Optimizer," Swarm and Evolutionary Computation, 2018.

[55] S. Gupta and K. Deep, "Random Walk Grey Wolf Optimizer for Constrained Engineering Optimization Problems," Computational Intelligence, 2018.

[56] Fr et al., "A Dynamic Island-based Genetic AlgorithmsFramework," Proceedings of the 8th International Conference on Simulated Evolution and Learning, Kanpur, India, 2010.

[57] H. T. T. Thein, "Island Model Based Differential Evolution Algorithm for Neural Network Training," Advances in Computer Science: An International Journal (ACSIJ), vol. 3, no. 1, pp. 67-73, 2014.

[58] Z. A. Mostafa, N. H. Awad and R. M. Duwairi, "Multi-objective Differential Evolution Algorithm with A New Improved Mutation Strategy," International Journal of Artificial Intelligence™, vol. 14, no. 2, pp. 23-41, 2016.

[59] J. F. Romero and C. Cotta, "Optimization by Island-structured Decentralized Particle Swarms," Computational Intelligence, Theory and Applications: Springer, pp. 25-33, 2005.

[60] M. Randall and A. Lewis, "A Parallel Implementation of Ant Colony Optimization," Journal of Parallel and Distributed Computing, vol. 62, no. 9, pp. 1421-1432, 2002.

[61] S. Gupta and K. Deep, "Performance of Grey Wolf Optimizer on Large Scale Problems," AIP Conference Proceedings, vol. 1802, no. 1, p. 020005, 2017, AIP Publishing.

[62] R. Tanabe and A. S. Fukunaga, "Improving the Search Performance of SHADE Using Linear Population Size Reduction," IEEE Congress on Evolutionary Computation (CEC), pp. 1658-1665, 2014.

[63] K. S.SreeRanjini and S. Murugan, "Memory-based Hybrid Dragonfly Algorithm for Numerical Optimization Problems," Expert Systems with Applications, vol. 83, pp. 63-78, 2017.

[64] C. Yu, L. Kelley, S. Zheng and Y. Tan, "Fireworks Algorithm with Differential Mutation for Solving the CEC 2014 Competition Problems," IEEE Congress on Evolutionary Computation (CEC), pp. 3238-3245, 2014.

[65] P. N. Suganthan et al., "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-parameter Optimization," KanGAL Report, vol. 2005005, p. 2005, 2005.

[66] B. H. Abed-alguni, "Action-Selection Method for Reinforcement Learning Based on Cuckoo Search Algorithm," Arabian Journal for Science and Engineering, pp. 1-15, 2017.

[67] J. Liang, B. Qu and P. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-parameter Numerical Optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China; and Technical Report, Nanyang Technological University, Singapore, 2013.

[68] E. Emary, H. M. Zawbaa, C. Grosan and A. E. Hassenian, "Feature Subset Selection Approach by Gray-wolf Optimization," Afro-European Conference for Industrial Advancement, pp. 1-13, Springer, 2015.

[69] B. H. Abed-alguni, S. K. Chalup, F. A. Henskens and D. J. Paul, "A Multi-agent Cooperative Reinforcement Learning Model Using a Hierarchy of Consultants, Tutors and Workers," Vietnam Journal of Computer Science, vol. 2, no. 4, pp. 213-226, 2015.

[70] B. H. Abed-alguni, S. K. Chalup, F. A. Henskens and D. J. Paul, "Erratum to: A Multi-agent Cooperative Reinforcement Learning Model Using a Hierarchy of Consultants, Tutors and Workers," Vietnam Journal of Computer Science, vol. 2, no. 4, pp. 227-227, 2015.

[71] B. H. K. Abed-alguni, "Cooperative Reinforcement Learning for Independent Learners," 2014.

[72] B. H. Abed-alguni and M. A. Ottom, "Double Delayed Q-learning," International Journal of Artificial Intelligence, vol. 16, no. 2, pp. 41-59, 2018.

ملخص البحث:

تعد خوارزمية "الذئاب الرمادية" من الخوارزميات المهمة المستخدمة في الأمثلة والمبنية على تقنية السِّرب. وقد استوحيت من استراتيجية المطاردة وهرمية القيادة التي تتبعها الذئاب الرمادية. وقد استخدمت بنجاح لحل مشكلات متعددة من المشكلات المستمرة والمجرّدة لعملية الأمثَلة. ومع ذلك، فإن السلبية الرئيسية لها تكمن في أنها قد تتجمع في شكل حلول شبه مثالية في مراحل مبكرة من عملية المحاكاة بسبب فقدان التنوع في مجتمع الخوارزمية.

هذه الورقة تقدم تغييراً موزعاً لخوارزمية الذئاب الرمادية في محاولة لتحسين التنوع في الخوارزمية، وذلك عبر تنظيم مجتمع الخوارزمية في هيئة مجموعات مستقلة صغيرة (جُزُر) بناءً على النموذج المعروف بنموذج الجُزُر. الخوارزمية المقترحة تطبق الخوارزمية الأصلية على كل جزيرة من الجُزُر، مما يسمح بتبادل الحلول المختارة بين الجُزر بناءً على تقنية الحلقات العشوائية وسياسة الهجرة بين الأفضل والأسوأ. وتوفر الخوارزمية المعدّلة بيئة أفضل للحلول المرشحة غير المهيأة في كل جزيرة كي تتطور الى حلولٍ أفضل، الأمر الذي يزيد من احتمالية إيجاد الحلول المثالية.

ومن السِّمات المهمة الأخرى للخوارزمية المعدّلة المقترحة أنها يمكن أن تعمل باستخدام أجهزة متوازية، الأمر الذي يعني إمكانية التقليل من تعقيد الحسابات مقارنة بالصيغ القائمة من خوارزمية الذئاب الرمادية. ومن ناحية أخرى، تمّ تقييم الخوارزمية المقترحة في هذه الورقة من خلال مقارنتها بأنواع أخرى من الخوارزميات المبنية على تقنية السِّرب. كذلك جرى تقييم حساسية الخوازمية المقترحة لمتغيراتها باستخدام خمس عشرة دالّة اختبار معيارية. وأشارت المقارنة واختبار الحساسية المشار إليهما الى أن الخوارزمية المقترحة تؤدي أداءً منافساً بالمقارنة مع الخوارزميات الأخرى.