Island-based whale optimization algorithm for continuous optimization problems

Article *in* International Journal of Reasoning-based Intelligent Systems · March 2019

CITATION
1

READS
25

3 authors:

Bilal Abed-alguni
Yarmouk University
19 PUBLICATIONS **136** CITATIONS
SEE PROFILE

Ahmad F Klaib
Yarmouk University
5 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Khalid Nahar
Yarmouk University
33 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Island-based whale optimization algorithm for continuous optimization problems

BILAL H. ABED-ALGUNI
Department of Computer Sciences, Yarmouk University
Irbid, Jordan
Email: bilal.h@yu.edu.jo

AHMAD F. KLAIB*
Department of Computer Information Systems, Yarmouk University
Irbid, Jordan
Email: ahmad.klaib@yu.edu.jo
*Corresponding author

KHALID M. NAHAR
Department of Computer Sciences, Yarmouk University
Irbid, Jordan
Email: khalids@yu.edu.jo

Abstract The whale optimization algorithm (WOA) is a newly proposed evolutionary algorithm that uses a simulation model based on the bubble-net hunting mechanism of humpback whales to find solutions for different classes of optimization problems. WOA may occasionally converge to suboptimal solutions because of the loss of diversity in its population of candidate solutions. The island model is a distributed approach that is commonly used to control the population diversity in evolutionary algorithms. This paper introduces an improved version of WOA namely island-based whale optimization algorithm (iWOA) that incorporates the island model into WOA. In iWOA, the population of candidate solutions is divided into separate sub-populations called islands. The improvement loop of WOA is then applied separately to the candidate solutions in each island. After a predetermined number of generations, a number of candidate solutions are swapped between islands through a process known as migration that is based on the random ring topology. The migration process is conducted to maintain the diversity of population and also to allow each island to exchange candidate solutions with a selected neighbouring island. The iWOA algorithm was tested and compared to well-known optimization algorithms using 18 standard benchmark functions. The simulation results indicate that iWOA improves the accuracy of results compared to WOA and other popular evolutionary algorithms. In addition, the sensitivity analysis of iWOA to its parameters indicates that its convergence behaviour is sensitive to the parameters of the island model.

Keywords: Whale Optimization; Island model; Structured population; Optimization; Evolutionary algorithm.

1 Introduction

The whale optimization algorithm (WOA) is a relatively new evolutionary algorithm that attempts to find solutions for different classes of optimization problems by simulating the bubble-net hunting strategy of humpback whales (Aljarah et al. (2018); Mukherjee et al. (2018); El Aziz et al. (2017)). A major problem with WOA is that it may converge earlier than expected to sub-optimal solutions. Such a phenomenon is called premature convergence which occurs in WOA when the two main internal parameters of WOA ($\overrightarrow{A}$ and $\overrightarrow{C}$, see Section 2.3) are unable to produce better offsprings over consecutive generations, resulting in loss of diversity in the population (Mirjalili and Lewis (2016); Mafarja and Mirjalili (2017)).

Few research studies have been conducted to enhance the performance of WOA (Mafarja and Mirjalili (2017); Kaveh and Ghazaan (2017); Ling et al. (2017); Trivedi et al. (2016)). Mafarja and Mirjalili (2017) introduced two approaches to hybridize the WOA algorithm with the Simulated Annealing (SA) algorithm. In the first algorithm, SA is used after each iteration of the WOA algorithm to improve the best current solution of WOA, while in the second algorithm, SA is part of the WOA algorithm and is used to confirm that the best solution found by WOA so far is the best possible solution in its neighborhood. In general, the goal of using SA with

WOA is to reduce the probability of premature convergence and enhance the accuracy of the best solutions. Kaveh and Ghazaan (2017) proposed an Enhanced WOA algorithm (EWOA) for solving skeletal structures problems. In EWOA, random candidate solutions are chosen to be updated at each iteration of WOA, where only some variables are selected and updated for each selected solution. The numerical results in Kaveh and Ghazaan (2017) suggest that EWOA provides more accurate and reliable results than WOA.

The Lévy flight trajectory-based WOA algorithm (LWOA) is an improved variation of the WOA algorithm (Ling et al. (2017)). The LWOA algorithm uses the Lévy flight trajectory to update the values of the candidate solutions after updating it with the original WOA algorithm. LWOA has been proven empirically that it converges faster than WOA and is capable of jumping out of local minima/maxima regions. The adaptive WOA (AWOA) is a variation of the WOA algorithm that uses a new randomization method called adaptive method (Trivedi et al. (2016)). The AWOA dynamically adjusts the values of the parameters of WOA using a random function based on the fitness of the candidate solutions over the course of iterations. The experimental results in Trivedi et al. (2016) using simple 10 benchmark functions suggest that AWOA performs better than WOA.

In general, the existing hybrid WOA algorithms that integrate other search methods with the WOA algorithm (e.g., Mafarja and Mirjalili (2017); Ling et al. (2017); Wang et al. (2017); Abdel-Basset et al. (2018))require heavy computations, especially when using searching methods at each iteration of WOA. Other enhanced versions of the WOA algorithm (e.g., Mehne and Mirjalili (2018); Hasanien (2018); Alam et al. (2018); Kaveh and Ghazaan (2017); Prakash and Lakshminarayana (2017); Trivedi et al. (2016)) do not significantly enhance the performance of WOA and can be applied only to certain applications.

The structured population model has been recently incorporated with different evolutionary algorithms (Michel and Middendorf (1998); Romero and Cotta (2005); Lardeux and Goëffon (2010); Kushida et al. (2013); Al-Betar et al. (2015); Al-Betar and Awadallah (2018)). Evolutionary algorithms with structured population divide the candidate solutions in the population into sub-populations. The goal here is to improve both the convergence behavior and convergence speed of the evolutionary algorithms using methods that maintain the diversity of the candidate solutions in the search space. The island model is the most used structured population model that partitions the whole population of candidate solutions of a large population into smaller independent populations called (islands) (Kushida et al. (2013)). In the island model, an evolutionary algorithm is usually applied to each island independently. These islands interact periodically via a process known as the migration process. Hence, the population diversity can be preserved and the evolutionary algorithm can be used with parallel machines. Island-based evolutionary algorithms such as island-based Genetic algorithm (Lardeux and Goëffon (2010)), island-based Harmony search (Al-Betar et al. (2015)), island-based Bat algorithm (Al-Betar and Awadallah (2018)), island-based Ant colony optimization (Michel and Middendorf (1998)), island-based Particle swarm optimization (Romero and Cotta (2005)) and island-based Differential evolution (Kushida et al. (2013)) have shown high performance in solving a wide range of optimization problems including discrete and continuous optimization problems.

This paper introduces an enhanced version of WOA namely island-based whale optimization algorithm (iWOA) that incorporates the island model into WOA. In iWOA, the population of candidate solutions is divided and organized into independent populations (islands). The candidate solutions on each independent island are iteratively improved using the evolutionary operators of WOA. After a predefined number of generations, a number of candidate solutions are swapped between islands through a process called migration based on a random ring topology. The migration process plays a significant role in maintaining the diversity of population by allowing the exchange of candidate solutions between neighbouring islands. The proposed algorithm aims to reduce the computational time required for WOA to converge to good solutions.

The rest of the paper is organized as follows: the WOA algorithm and the concepts of island model are overviewed in Section 2. The iWOA algorithm is thoroughly discussed in Section 3. The experimental results are reported and discussed in Section 4. Finally, the conclusion of the paper is presented in Section 5.

2 Related work

This section is divided into three subsections. Section 2.1 describes continuous optimization problems, Section 2.2 provides an overview of the island model and Section 2.3 describes the original whale optimization algorithm.

2.1 Continuous optimization problems

Mathematical optimization problems are commonly categorized into two types based on whether their decision variables are continuous or discrete (Abed-alguni and Alkhateeb (2018); Khaleel and Ahmed (2012); Sabet et al. (2013); Abed-alguni (2018); Huo et al. (2015); Abed-alguni and Barhoush (2018)). A continuous optimization problem, which is the main concern of the current study, is conventionally modeled as a minimization problem $min\{f(\overrightarrow{X_i})|\overrightarrow{X}_i(i=1,2,...,N)\}$, where $f(\overrightarrow{X})$ is the objective function of the feasible solution that consists of M decision variables $\overrightarrow{X}=\{x_1,...,x_M\}$. The value of decision variable x_j is in the range $[LB,UB]$, where LB and UB are the lower and upper bounds of the search range, respectively. The main purpose of solving a minimization problem is to find an optimal solution $\overrightarrow{X}^*$ in the search space S such that $f(\overrightarrow{X}^*) \leq f(\overrightarrow{X}_i),\ \forall \overrightarrow{X}_i \in S$.

2.2 Concepts of Island model

The island model is a distributed population model in which the population of a given optimization problem is separated and organized into small populations (islands) (Corcoran and Wainwright (1994)). This model allows a standard evolutionary algorithm to be applied independently to each island (Lardeux and Goëffon (2010)). The interaction between islands takes place through a migration process. A migration process specifies how candidate solutions can be swapped between islands based on two parameters: migration rate and migration frequency. Migration is an important mechanism in the island model, without it the islands are a set of separate islands. Actually, there are two main advantages for the island model (Skolicki (2005)). First, dividing the population of candidate solutions into several islands may increase the likelihood of finding global optimal solutions by providing a chance to unfit candidate solutions to be improved in separate islands. Second, the island model can be implemented to parallel hardware and thus the computation time of evolutionary algorithms can be reduced.

Five issues have to be considered to include the island model into the organization of any evolutionary method. First, the number of islands. Second, the size of each island where the islands can have the same size or variable sizes. Third, the migration rate which determines the number of candidate solutions that should be exchanged among islands after each predefined number of iterations. Fourth, the migration frequency which determines the number of iterations between migration waves. Finally, the choice between synchronous migration (i.e., the exchange of candidate solutions among islands occur at the same time) and asynchronous migration (i.e., the exchange of candidate solutions among islands does not occur at the same time).

Migration can be implemented through a migration topology which specifies the possible path for swapping candidate solutions between islands depending on a migration policy (i.e., a policy that determines which candidate solutions should be exchanged between islands). The choice of migration topology significantly influences the performance of island-based evolutionary algorithms. The most popular classes of migration topologies are the static topologies (e.g., star, ring and mesh) and the dynamic topologies (e.g., random star, random ring and random mesh) (Corcoran and Wainwright (1994)). The migration paths between the islands in static topologies are fixed, while the migration paths in dynamic topologies are dynamically changed before every migration process. Figure 1 illustrates a unidirectional-connected graph that represents the random ring topology. The nodes represent the independent islands and each unidirectional-edge between each two nodes represents the migration direction. It is important to note that in the ring or random ring topologies, each island has one neighboring island.

Each time a predefined number of generations of evolution is reached (migration frequency), the migration rate specifies the number of migrant candidate solutions to be exchanged between islands based on a migration topology. The migrants solutions are normally selected based on a migration policy (Kushida et al. (2013)), which can be either a random-based migration policy (e.g., random-random policy) or a greedy-based migration policy (e.g., best-worst policy) (Skolicki (2005)). The random-random policy specifies how random candidate solutions form one island can be exchanged with other random solutions in a neighboring island while the best-worst policy specifies how the best candidate solutions in one island can be swapped with the worst candidate solutions in a neighboring island.

2.3 Whale optimization algorithm

The foraging behavior of humpback whales is the inspiration source of a new mathematical optimization algorithm known as the whale optimization algorithm (WOA) (Mirjalili and Lewis (2016); Mafarja and Mirjalili (2017)). Humpback whales are ocean creatures that normally work jointly to catch their food following a distinctive hunting strategy that is widely known as the bubble-net feeding strategy. In detail, a group of whales swim below a shoal of fish and then move in a circular motion towards the surface of ocean while blowing bubbles. These actions create a circular path of bubbles around the shoal of prey and gradually force it to move towards the surface of ocean.

In the WOA algorithm, the solution with the best objective value (best solution) compared to the other candidate solutions represents the target prey and each candidate solution represents a whale. The WOA algorithm attempts to optimize a population of candidate solutions for a given optimization problem by simulating the bubble-net feeding strategy. This strategy is divided into two phases: exploitation and exploration. The exploitation phase models the encircling of prey and spiral bubble-net attacking methods, while the exploration phase simulates the random search for a prey.

In the exploitation phase, each candidate solution is updated using the current best solution as follows:

$$\overrightarrow{D} = |\overrightarrow{C}.\overrightarrow{X}^*(t) - \overrightarrow{X}(t)| \tag{1}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}^*(t) - \overrightarrow{A}.\overrightarrow{D} \tag{2}$$

where t represents the current iteration number, $\overrightarrow{X}(t)$ is a candidate solution and $\overrightarrow{X}^*(t)$ is the best candidate solution found so far. The dot operator is an element-wise multiplication operator (i.e., an operator that calculates a new vector with elements that are the products of the corresponding elements of two vectors). Note that $\overrightarrow{X}^*$ should be updated whenever a better solution is found. $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors that can be calculated respectively as follows:

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

$$\overrightarrow{C} = 2.\overrightarrow{r} \tag{4}$$

where $\overrightarrow{r}$ is a vector of random numbers that are generated from a uniform random distribution in the interval $[0, 1]$ and $\overrightarrow{a}$ is a number that linearly decreases from 2 to 0 over the course of the simulation process of WOA.

Equations 1 and 2 show that the candidate solutions (whales) update their values using the best calculated candidate solution (prey). The Adjustment of the values of vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ using equations 3 and 4 control the regions where a candidate solution can be located in the neighborhood of the current best solution.

The random vector $\overrightarrow{r}$ is used to move the candidate solution to any possible position in the neighborhood of the best solution found so far. Similarly, the same principles of WOA can be applied to n-dimension search space (i.e., search space with n-decision variables) where the candidate solutions move in hyper-cubes around the best current solution.

The WOA algorithm uses two approaches to mathematically model the bubble-net behavior of humpback whales: the shrinking encircling mechanism and the spiral-shaped path. The shrinking encircling mechanism is simulated by linearly decreasing $\overrightarrow{a}$ in equation 3 using the following equation:

$$a = 2 - t\frac{2}{Maxt} \tag{5}$$

where $Maxt$ indicates the maximum number of iterations and t represents the current iteration number.

It is a good practice to set the values of $\overrightarrow{A}$ in $[-1, 1]$ so the new value of a solution can be defined at any position between its original value and the value of the best solution calculated so far (Mirjalili and Lewis (2016)).

The simulation of spiral-shaped path requires the calculation of the distance between a candidate solution (X) and the current best solution (X^*). The spiral equation is then used to calculate the value of the neighbor solution as follows:

$$\overrightarrow{X}(t+1) = \overrightarrow{D'}.e^{bl}.cos(2\pi l) + \overrightarrow{X^*}(t) \tag{6}$$

where $\overrightarrow{D'} = |\overrightarrow{X^*(t)} - \overrightarrow{X}(t)|$ is the distance between solution $\overrightarrow{X}(t)$ and best solution $\overrightarrow{X^*(t)}$ at iteration t and $l \in [-1, 1]$ is a random number. The constant number b is used to define the shape of logarithmic spiral.

It is recommended in Mirjalili and Lewis (2016) to select between the two exploitation mechanisms of WOA (shrinking encircling and spiral-shaped path) with equal probability as follows:

$$\overrightarrow{X}(t+1) = \begin{cases} \text{Shrinking Encircling} & if\ p < 0.5 \\ \text{Spiral-shaped Path} & if\ p \geq 0.5 \end{cases}$$

where $p \in [0, 1]$ is a random number.

The exploration phase in WOA (random search for prey) takes place when $|\overrightarrow{A}| > 1$. In this phase, the candidate solutions are updated in the direction of a randomly selected candidate solution. This provides a chance for the WOA algorithm to explore the entire search space. The random search for a prey is mathematically modeled in WOA as follows:

$$\overrightarrow{D} = |\overrightarrow{C}.\overrightarrow{X}_{rand} - \overrightarrow{X}| \tag{7}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_{rand} - \overrightarrow{A}.\overrightarrow{D} \tag{8}$$

where $\overrightarrow{X}_{rand}$ is a randomly selected solution form the population of WOA at iteration t.

Figure 2 provides the pseudo-code of the WOA algorithm. Initially, the population of WOA is a set of N random solutions $\overrightarrow{X}_i (i = 1, 2, ..., N)$ for a given optimization problem. Each random solution is a vector that comprises M decision variables $\left(\overrightarrow{X}_i = (x_1, ..., x_M)\right)$. The value of decision variable x_j is in the range $[LB_j, UB_j]$, where LB_j and UB_j are the lower and upper bounds of x_j, respectively. The WOA algorithm uses the objective function $f(\overrightarrow{X})$ of a given optimization problem to calculate the objective value for each candidate solution in the

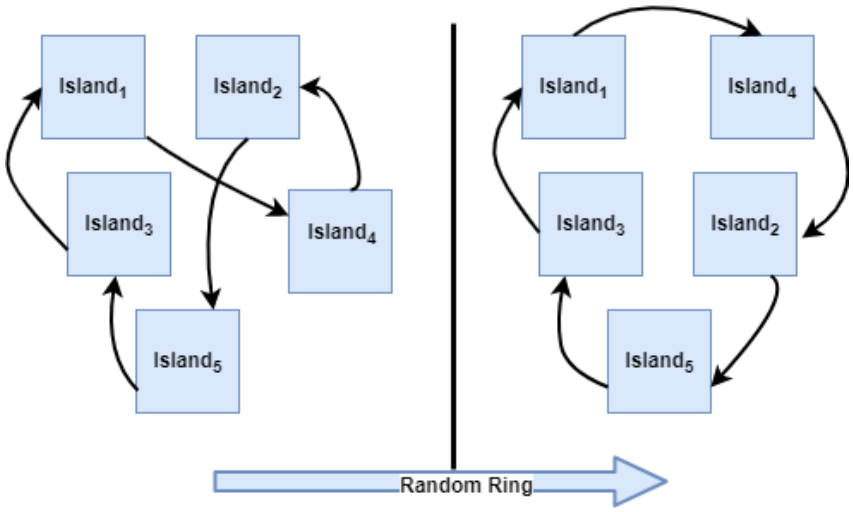

Figure 1 Random ring topology.

population to find the best solution. The improvement loop of WOA is composed of four main steps. First, update all the parameters and coefficient vectors. Second, generate a random number $p \in [0, 1]$ and then use it to update the candidate solutions using equation 2, equation 6 or equation 8. Third, discard the solutions that go outside of the search space. Finally, return the best solution in the population.

```
1: Begin
2: Generate initial population X_i(i = 1, 2, ..., N)
3: Calculate the fitness of each solution
4: X* = the best solution
5: t = 0
6: while (t < Maxt) or (stop criterion) do
7:    for each solution do
8:       Update a, A, C, l, and p
9:       if P < 0.5 then
10:         if |A| < +1 then
11:            Update the current solution by equation 2
12:         else if |A| > +1 then
13:            Select a random solution (X_rand)
14:            Update the current solution by 8
15:         end if
16:      else if P ≥ 0.5 then
17:         Update the current solution by equation 6
18:      end if
19:   end for
20:   Check if any solution goes beyond the search space and amend it
21:   Calculate the fitness of each solution
22:   Update X*, if there is a better solution
23:   t = t + 1
24: end while
25: return X*
26: End
```

Figure 2 The original WOA algorithm (Mirjalili and Lewis (2016)).

3 Island-based whale optimization algorithm.

The whale optimization algorithm (WOA) is an efficient evolutionary algorithm based on the foraging behavior of humpback whales (Zhang et al. (2018); Prakash et al. (2018); Srivastava et al. (2018); Prakash and Lakshminarayana (2017)). As any other optimization algorithm, WOA may converge to suboptimal solutions because of the loss of diversity in its population of candidate solutions (Abed-alguni and Klaib (2018)).

This section provides a detailed description of the island-based WOA algorithm (iWOA) that integrates the island model into the structure of the WOA algorithm. The iWOA algorithm increases the likelihood of finding global optimal solutions by providing a chance to unfit candidate solutions to be improved in separate islands. In addition, iWOA can be implemented to parallel machines and thus its computation time can be reduced. In iWOA, the candidate solutions in the population are organized into k independent islands of size s. The equation $\sum_{i=1}^{k} s$ represents the total population of all of the islands. The basic WOA algorithm is then performed independently at the same time (i.e., synchronous execution) for each island. After a predetermined number of generations (iterations) as specified by migration frequency (F_m), the migration process in iWOA takes place by exchanging a number of candidate solutions determined by migration rate (R_m) between the islands. The iWOA algorithm transfers the best candidate solutions from one island to replace the worst solutions in the neighboring island based on the best-worst migration policy. The iWOA uses the random ring migration topology to determine possible migration paths. In detail, the iWOA algorithm is composed of the following steps:

Step 1 **Initialize the problem and iWOA parameters**. The optimization problem is modeled as a minimization problem as described in Section 2.1. Several categories of parameters should be initialized in this step. First, the problem parameters including the size of the candidate solution M, the possible range of each decision variable $[LB_i, UB_i]$, and the objective function $f(\overrightarrow{X}_i)$. Second, the parameters of WOA $(a, \overrightarrow{A}, \overrightarrow{C}, l, p)$. Third, the parameters of the island model: k which is the number of islands, s which is the size of the island $s = population/k$, migration frequency (F_m) which determines the number of generations before exchanging the candidate solutions among islands, and migration rate (R_m) which calculates the number of candidate solutions to be sent and received between the islands.

Step 2 **Initialize the population of N whales (feasible solutions)**. The population of N feasible solutions $\overrightarrow{X}_i(i = 1, 2, ..., N)$ are generated. The variables of a feasible solution $\overrightarrow{X} = \{x_1, ..., x_M\}$ are initialized with random values from the solutions' range of a given optimization problem $[LB, UB]$.

Step 3 **Divide the candidate solutions in the population into k islands of size s.** One vector is used in the partitioning process $I =< I_1, I_2, ..., I_k >$, where the value range of $I_j \in (1, 2, ..., k)$ and $j \in (1, 2, ..., n)$. The following algorithm shows how the population can be divided into k islands. At the end of this step, k islands are produced where $population =$

$(SubPop_1, SubPop_2, ...SubPop_k)$, where $SubPop_i$ is the sub-population of island I_i.

```
Divide-POP()
Begin
for j = 1...N do
  repeat
    h = U(1...k)
    if SubPop_h ≤ s then
      I_j = h
    end if
  until I_j is assigned to isand h
end for
End
```

Step 4 **Find the best solution in each island.** This step of iWOA is applied individually to each island. In each island, the fitness value of each candidate solution is calculated and the initial best solution $\overrightarrow{X^*}$ is obtained.

Step 5 **Select either the shrinking encircling mechanism or the spiral-shaped path mechanism with equal probabilities in each island.** Generate a random number $P \in [0, 1]$ in each island. For each island, if $P < 0.5$, go to **Step 6**. Otherwise go to **Step 7**.

Step 6 **Apply the shrinking encircling mechanism to the island.** If $|A| < +1$, update the current solution by equation 2. Otherwise select a random candidate solution $\overrightarrow{X}_{rand}$ and then update the current solution by equations 7 and 8. Go to **Step 8**.

Step 7 **Apply the spiral-shaped path mechanism to the island.** If $P \geq 0.5$, update the current solution by equation 6.

Step 8 **Update the population of each island**. In this step, all the feasible solutions in each island that go outside the search landscape are removed.

Step 9 **Find the best solution in each island.** In each island, the fitness of each candidate solution is calculated using the fitness function $f(\overrightarrow{X})$ and the best solution $\overrightarrow{X^*}$ is obtained.

Step 10 **Migration process.** The migration process is triggered each time a predefined number of iterations is reached as specified by F_m to send and receive solutions between islands. The islands are organized in a random ring topology, where the islands are randomly arranged to form a unidirectional ring. The iWOA swaps some candidate solutions among islands based on the best-worst migration policy. Let $SubPop_i = \{\overrightarrow{X}^i_1, \overrightarrow{X}^i_2, ..., \overrightarrow{X}^i_s\}$ and $SubPop_j = \{\overrightarrow{X}^j_1, \overrightarrow{X}^j_2, ..., \overrightarrow{X}^j_s\}$ be two islands I_i and I_j. Provided that $R_m = 15\%$ and $s = 20$, the number of migrant solutions is $R_m \times s = 15\% \times 20 = 3$. Let $SubPop_i$ and $SubPop_j$ be two ascendingly ordered lists based on their fitness values where $f(\overrightarrow{X}^i_1) \leq f(\overrightarrow{X}^i_2) \leq ... \leq f(\overrightarrow{X}^i_s)$ and $f(\overrightarrow{X}^j_1) \leq f(\overrightarrow{X}^j_2) \leq ... \leq f(\overrightarrow{X}^j_s)$. Let us assume that we have a unidirectional edge from I_i to I_j. Hence, the three best candidate solutions in I_i $(\overrightarrow{X}^i_1, \overrightarrow{X}^i_2, \overrightarrow{X}^i_3)$ replace the three worst solutions in I_j $(\overrightarrow{X}^j_{s-2}, \overrightarrow{X}^j_{s-1}, \overrightarrow{X}^j_s)$. The same process apply to all of the islands in the ring.

Step 11 **Rank the solutions in each island and then return the best solution.** The solutions in each island are ranked based on their fitness values and then the best solution in all islands is calculated.

Step 12 **Check the stopping condition**. Repeat **Steps 4-11**, if the stopping condition is not satisfied. Otherwise, return the best obtained solution in all islands.

4 Experiments

4.1 Benchmark Functions

All of the algorithms in the experiments were compared and evaluated using 18 benchmark functions. Table 1 shows a summary of these test functions reporting the abbreviation, function name, search range, number of decision variables (D) and optimum value $f(\overrightarrow{X^*})$ of each function. This set of benchmark functions has been used in the literature to evaluate the performance of various evolutionary algorithms (Abed-alguni and Klaib (2018); Abed-alguni and Alkhateeb (2018, 2017); Alkhateeb and Abed-alguniet (2017)). Note that all of the test functions in Table 1 are continuous minimization functions.

4.2 Setup

In all of the experiments, the size of population in each algorithm was 300 candidate solutions. For the WOA and iWOA algorithms, the parameter $\overrightarrow{a}$ periodically decreases from 2 to 0 over the course of the simulation process, the vector $\overrightarrow{A}$ is in $[-1, 1]$, the random vector $\overrightarrow{r}$ is in $[0, 1]$ and the probability of mechanism selection $p = 0.5$ as suggested in Mirjalili and Lewis (2016); Mafarja and Mirjalili (2017). The value of p was set to 0.5 to balance between the exploration and exploitation of candidate solutions in the population.

In Section 4.3, the main parameters of iWOA (number of islands k, frequency of migration F_m, and rate of migration R_m) were experimentally studied to evaluate their influence on the performance of iWOA. In Section 4.4, the iWOA algorithm was compared with the original WOA algorithm and other state-of-the-art evolutionary algorithms (Island-based genetic algorithm

Table 1 Selected benchmark functions.

Abbreviation	Function name	Search range	D	$f(\overrightarrow{X^*})$
f_1	Wood's function	$[-10, 10]$	10	0
f_2	Himmelblau's function	$[-6, 6]$	2	0
f_3	Zakharov's function	$[-5, 10]$	10	0
f_4	Dixon's price function	$[-10, 10]$	10	0
f_5	Salomon's function	$[-10, 10]$	10	0
f_6	Quartic function with noise	$[-10, 10]$	10	0
f_7	"Drop Wave" function	$[-5.12, 5.12]$	10	-1
f_8	Xin She Yang function1	$[-20, 20]$	10	-1
f_9	De Jong's first function	$[-100, 100]$	10	0
f_{10}	Rosenbrock's valley function	$[-2.048, 2.048]$	10	0
f_{11}	Rotated hyper-ellipsoid function	$[-100, 100]$	10	0
f_{12}	Griewank's function	$[-600, 600]$	10	0
f_{13}	Schwefel 2.22 function	$[-100, 100]$	10	0
f_{14}	Ackley's function 2.9	$[-32.768, 32.768]$	10	0
f_{15}	Shifted sphere function	$[-100, 100]$	10	-450
f_{16}	Shifted Schwefel's 2.22 function	$[-100, 100]$	10	-450
f_{17}	Shifted Rastrigin's function	$[-5, 5]$	10	-330
f_{18}	Shifted Expanded Griewank's plus	$[-5, 5]$	10	-130

(iGA) (Lardeux and Goëffon (2010)), island-based Harmony search (iHS) (Al-Betar et al. (2015)), Particle Swarm optimization algorithm (PSO) (Abed-alguni et al. (2016); Yalcin et al. (2015)) and Bat algorithm (BA) (Yang and Gandomi (2012); Abed-alguni (2017))). The weight parameters of PSO were W=0, C1=C2=1 as in Abed-alguni et al. (2016); Abed-alguni (2017). In the island-based genetic algorithm (iGA), the mutation rate was 0.05 and the crossover rate was 0.7 as in Klempka and Filipowicz (2017). In the BA algorithm, the discount parameter of the frequencies (β) was set to 0.5, the loudness A was set to 1 and the pulse rate r was set to 0 for each possible solution in the search space.

4.3 Experimental analysis of the parameters of iWOA

In this section, the sensitivity of iWOA to its parameters is analyzed. This aims to discover the settings of the island model that increase the convergence speed of iWOA. The main parameters of iWOA (number of islands k, frequency of migration F_m, and rate of migration R_m) were empirically studied based on different experimental scenarios (see Table 2) to analyze their effect on the convergence of iWOA. The first three experimental scenarios in the table were used to evaluate the effect of s on the performance of iWOA. The second three experimental scenarios were used to evaluate the effect of F_m on the performance of iWOA. Finally, the last three experimental scenarios were used to evaluate the effect of R_m on the performance of iWOA. Similar experimental scenarios have been used to evaluate the performance of island-based Harmony search in (Al-Betar et al. (2015)) and island-based Bat algorithm in (Al-Betar and Awadallah (2018)).

Table 3 reports the average of best obtained objective values for 50 independent runs for each of the 18 test functions in Table 1 based on the nine scenarios in Table 2. The best results in Table 3 are colored with red (lowest is best).

The results of first three experimental scenarios in Table 3 indicate that the performance of iWOA enhances with an increase in the size of island. This is because large values of s may indicate that the population is more diverse than the population with small values of s. In detail, the results in Table 3 show that iWOA with $s = 10$ (Scenario 3) achieved the best objective values for 10 functions of the 18 functions. iWOA with $s = 5$ (Scenario 2) is the second best performing algorithm with 3 functions out of 18 followed by iWOA with $s = 2$ (Scenario 1) which achieved the best objective values for 2 functions out of 18 functions.

The value of $s = 10$ which produced the best objective values in the first three Scenarios was chosen to be used in the remaining six experimental scenarios. The results of Scenario 4 ($F_m = 50$), Scenario 5 ($F_m = 100$) and Scenario 6 ($F_m = 500$) in Table 3 suggest that large-scale migration does not necessarily enhance the performance of iWOA because the diversity in islands can be lost so fast with large-scale migration. As shown in Table 3, iWOA achieved in Scenario 5 the best objectives values for 7 functions out of the 18 functions, while iWOA in Scenario 4 achieved the best objective values for 3 of the 18 functions and iWOA in Scenario 6 achieved only one best result.

The value of $F_m = 100$ which produced the best objective values in the second experimental set (Scenario 4, Scenario 5 and Scenario 6) was chosen to be used in last three experimental scenarios. The results of Scenario 7 ($R_m = 10\%$), Scenario 8 ($R_m = 20\%$) and Scenario 9

($R_m = 30\%$) in Table 3 indicate that large values of R_m tend to negatively influence the diversity in the islands and the performance of iWOA. This is because a large value of R_m means that a large number of candidate solutions will be replaced each time migration occurs which makes the population in the islands similar to each other. Interestingly, iWOA with $R_m = 20\%$ produced the best results for 9 out of while iWOA with $R_m = 30\%$ achieved the best objective value for only two functions.

4.4 Comparison between iWOA and other algorithms

Table 4 reports the average of best obtained objective value for 50 independent runs for each of the 18 test functions in Table 1. The best results are colored with red. Interestingly, the results in Table 4 show that iHS outperforms the other algorithms for 11 out of 18 functions. The results also show that iWOA is the second best performing algorithm (best results for 10 out of 18 functions) and iGA is the third best preforming algorithm (best results for 6 out of 18 functions). The overall results indicate the iWOA provides competitive results compared to the other algorithms. The notable performance of the island-based optimization algorithms (iHs, iWOA, iGA) compared to the standard optimization algorithms (WOA, BA, PSO) is expected because the island model provides suitable environment for unfit candidate solutions to evolve in independent islands before the migration process.

5 Conclusion

The current paper presented a new variation of the WOA algorithm called iWOA that incorporates the island model into the framework of WOA. The population in iWOA is organized into several independent islands. The WOA operators are used in each independent island to improve its population. After a predetermined number of generations, the migration process takes place where a number of candidate solutions are swapped between islands based on the random ring topology and the bets-worst migration policy. The migration process is accomplished to maintain the diversity of population and also to allow the islands to exchange candidate solutions with each other. The proposed algorithm aims to reduce the computational time required for WOA to converge to good solutions.

The main parameters of iWOA (number of islands k, migration frequency F_m, and migration rate R_m) were empirically studied based on different experimental scenarios to analyze their effect on the convergence of iWOA. The experimental results indicate that high values of s (high number of islands) provide good results due to better population diffusion compared to low values of s (low number of islands). This also emphasizes that the size of island (k) has a significant effect on the convergence speed of iWOA. This may be because very low populated islands are most likely less diverse than high populated islands, which may cause iWOA to converge prematurely. However, very large islands normally require very large computations.

An interesting observation from the experiments is that large-scale migrations (i.e., high values of F_m) have negative influence of the convergence behavior of iWOA. On the other hand, low-scale migrations (i.e., low values of F_m) provide a better chance for the migrant solution to migrate without affecting the target candidate solutions of any island which means that iWOA will take longer time to converge to solutions. Additionally, a very small value of R_m enhances the diversity of islands, which indicates that a single candidate solution can act as a seed for improvement. On the contrary, a large value of R_m has a negative impact on diversity, simply because it replaces a larger number of candidate solutions.

In addition, iWOA was compared with the original WOA algorithm and other well-known evolutionary algorithms (Island-based genetic algorithm (iGA) (Lardeux and Goëffon (2010)), island-based Harmony search (iHS) (Al-Betar et al. (2015)), Particle Swarm optimization algorithm (PSO) (Abed-alguni et al. (2016)) and Bat algorithm (BA) (Yang and Gandomi (2012); Abed-alguni (2017))). The obtained results indicate that iWOA provides competitive performance compared to the other algorithms for most of the test functions.

Future work includes incorporating the island model into the framework of cooperative Q-learning (Abed-alguni et al. (2015); Abed-alguni et al. (b) (2015); Abed-alguni(b) (2017)). Future work also includes incorporat-

Table 2 Nine experimental scenarios for evaluating the sensitivity of the parameters of iWOA.

Experimental Scenario	s	F_m	R_m (%)
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	100	10
Scenario 8	10	100	20
Scenario 9	10	100	30

Table 3 The effect of various values of the parameters of *i*WOA on its convergence behavior.

Abb	Scenario 1 $s = 2$	Scenario 2 $s = 5$	Scenario 3 $s = 10$	Scenario 4 $F_m = 50$	Scenario 5 $F_m = 100$	Scenario 6 $F_m = 500$	Scenario 7 $R_m =$ 10%	Scenario 8 $R_m =$ 20%	Scenario 9 $R_m =$ 30%
f_1	1.32E-02	7.21E-05	9.24E-05	4.07E-04	9.24E-05	3.55E-04	3.18E-04	9.24E-05	3.48E-04
f_2	1.28E-13	2.17E-16	9.76E-17	6.27E-16	9.76E-17	1.45E-16	4.08E-16	9.76E-17	7.14E-16
f_3	2.95E-168	1.99E-122	1.37E-105	7.18E-112	1.37E-105	3.27E-102	1.19E-99	1.37E-105	8.01E-102
f_4	6.67E-01	6.67E-01	5.70E-01	6.67E-01	5.70E-01	6.44E-01	6.27E-01	5.70E-01	6.67E-01
f_5	9.99E-02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.07E-02	0.00E+00	2.32E-02
f_6	1.04E-05	9.41E-06	1.94E-06	4.82E-06	9.94E-07	6.44E-06	6.13E-06	9.94E-07	5.96E-06
f_7	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0	-1.0E+0
f_8	3.99E-03	3.46E-03	2.69E-03	2.73E-03	9.69E-02	2.86E-03	3.02E-03	9.69E-02	8.68E-02
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{10}	6.86E-04	6.45E-05	5.46E-05	4.15E-05	5.46E-05	7.45E-05	6.62E-05	5.46E-05	5.43E-06
f_{11}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{13}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{14}	1.81E-15	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16	5.98E-16
f_{15}	1.12E+03	1.27E+03	1.11E+03	1.17E+03	1.11E+03	1.12E+03	1.02E+03	1.11E+03	1.07E+03
f_{16}	1.79E+04	2.14E+04	1.72E+04	1.61E+04	1.72E+04	1.73E+04	1.66E+04	1.72E+04	1.74E+04
f_{17}	-2.89E+2	-2.86E+2	-2.88E+2	-2.89E+2	-2.88E+2	-2.89E+2	-2.87E+2	-2.88E+2	-2.86E+2
f_{18}	-1.02E+2	-9.23E+1	-1.03E+2	-9.86E+1	-1.03E+2	-9.89E+1	-9.66E+1	-1.03E+2	-9.88E+1

ing the island model into the frameworks of different hybrid Cuckoo search algorithms (Cuckoo search with simulated annealing (Alkhateeb and Abed-alguniet (2017)), Cuckoo search with Boltzmann selection (Abed-alguni and Alkhateeb (2017)).

References

Khaleel, T. A. and Ahmed, M. Y. (2012). 'Using intelligent water drops algorithm for optimisation routing protocol in mobile ad–hoc networks', *International Journal of Reasoning-based Intelligent Systems*, Vol. 4, No. 14, pp.227–234.

Sabet, S., Shokouhifar, M. and Farokhi, F. (2013). 'A discrete artificial bee colony for multiple knapsack problem', *International Journal of Reasoning-based Intelligent Systems*, Vol. 5, No. 2, pp.88–95.

Huo, J., Zhang, Y., Zhao, H. (2015). 'An improved artificial bee colony algorithm for numerical functions',

Table 4 Simulation results of the algorithms for 18 standard test functions, D=10, runs=100, iterations=10,000.

Abb	WOA	*i*WOA	*i*GA	PSO	BA	*i*HS
f_1	3.88E-01	9.24E-05	2.54E-03	9.80E-02	4.29E-01	3.56E-04
f_2	4.34E-14	9.76E-17	3.30E-15	2.78E-05	2.71E-06	1.56E-18
f_3	2.15E-167	1.37E-105	3.37E-155	2.40E-44	3.39E-51	2.26E-122
f_4	6.67E-01	5.70E-01	5.80E-02	8.69E-01	7.88E-01	2.30E-02
f_5	9.43E-02	0.00E+00	0.00E+00	1.88E-01	6.32E-01	0.00E+00
f_6	9.48E-06	1.94E-06	7.12E-06	9.39E-02	8.52E-04	2.24E-08
f_7	-9.89E-01	-1.00E+0	-1.00E+0	1.93E-02	3.93E-02	-1.00E+0
f_8	3.82E-03	2.69E-03	2.71E-03	1.30E-04	8.32E-07	2.46E-06
f_9	0.00E+00	0.00E+00	0.00E+00	8.33E-10	0.00E+00	4.14E-11
f_{10}	1.24E-01	5.46E-05	6.89E-03	7.80E-02	1.20E+01	18.85-02
f_{11}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.90E-19	0.00E+00
f_{12}	1.95E-02	0.00E+00	0.00E+00	5.63E-02	2.59E-04	0.00E+00
f_{13}	0.00E+00	0.00E+00	0.00E+00	1.52E-04	3.02E-03	0.00E+00
f_{14}	1.83E-15	4.44E-16	1.02E-15	4.33E+00	3.25E+00	1.27+E-05
f15	1.14E+03	1.11E+03	1.15E+02	5.19E+02	8.52E+02	-450
f16	1.66E+04	1.72E+04	2.13E+05	5.89E+04	1.45E+04	-449.94
f17	-2.87E+2	-2.88E+2	-3.01E+2	-2.66E+2	-2.81E+2	-329.7
f18	-9.71E+1	-1.03E+2	-8.78E+1	1.05E+0	1.03E+0	-5.03E+1

International Journal of Reasoning-based Intelligent Systems, Vol. 7, No. 3-4, pp.200–208.

Aljarah, I., Faris, H. and Mirjalili, S. (2018). 'Optimizing connection weights in neural networks using the whale optimization algorithm', *Soft Computing*, Vol. 22, No. 1, pp.1–15.

Mukherjee, V., Mukherjee, A. and Prasad, D. (2018). 'Whale Optimization Algorithm With Wavelet Mutation for the Solution of Optimal Power Flow Problem', *IGI Global*, pp.500–553.

El Aziz, M. A., Ewees, A. A. and Hassanien, A. E. (2017). 'Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation', *Expert Systems with Applications*, Vol. 83, pp.242–256.

Wang, J., Du, P., Niu, T. and Yang, W. (2017). 'A novel hybrid system based on a new proposed algorithm—Multi-objective Whale Optimization Algorithm for wind speed forecasting', *Applied Energy*, Vol. 208, pp.344–360.

Abdel-Basset, M., Manogaran, G., El-Shahat, D. and Mirjalili, S. (2018). 'A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem', *Future Generation Computer Systems*, Vol. 85, pp.129–145.

Mehne, H. H.and Mirjalili, S. (2018). 'A parallel numerical method for solving optimal control problems based on whale optimization algorithm', *Knowledge-Based Systems*, Vol. 151, pp.114–123.

Hasanien, H. M. (2018). 'Performance improvement of photovoltaic power systems using an optimal control strategy based on whale optimization algorithm'., *Electric Power Systems Research*, Vol. 157, pp.168–176.

Prakash, D. B. and Lakshminarayana, C. (2017). 'Optimal sitting of capacitors in radial distribution network using whale optimization algorithm'., *Alexandria Engineering Journal*, Vol. 56, No. 4, pp.499–509.

Srivastava, A., Das, D. K., Rai, A. and Raj, R. (2018). 'Parameter Estimation of a Permanent Magnet Synchronous Motor using Whale Optimization Algorithm'., *In 2018 Recent Advances on Engineering, Technology and Computational Sciences (RAETCS, IEEE*, pp.1–6.

Alam, A. Z., Faris, H. and Hassonah, M. A. (2018). 'Evolving Support Vector Machines using Whale Optimization Algorithm for spam profiles detection on online social networks in different lingual contexts'., *AlexandriaKnowledge-Based Systems*, Vol. 153, pp.91–104.

Zhang, C., Fu, X., Peng, S. and Wang, Y. (2018). 'Linear unequally spaced array synthesis for sidelobe suppression with different aperture constraints using whale optimization algorithm'., *In 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE*, pp.69–73.

Prakash, T., Singh, V. P. and Mohanty, S. R. (2018). 'A Novel Binary Whale Optimization Algorithm-Based Optimal Placement of Phasor Measurement Units'., *IGI Global*, pp.115–138.

Mirjalili, S. and Lewis A. (2016) 'The whale optimization algorithm', *Advances in Engineering Software*, Vol. 95, pp. 51–67.

Mafarja, M. M. and Mirjalili, S. (2017) 'Hybrid whale optimization algorithm with simulated annealing for feature selection', *Neurocomputing* Vol. 260, pp. 302 – 312.

Kaveh, A. and Ghazaan, M. I. (2017) 'Enhanced whale optimization algorithm for sizing optimization of skeletal structures', *Mechanics Based Design of Structures and Machines*, Vol. 45, No. 3, pp. 345–362.

Ling, Y., Zhou, Y. and Luo, Q. (2017) 'Lévy flight trajectory-based whale optimization algorithm for global optimization', *IEEE Access* Vol. 5, pp. 6168–6186.

Trivedi, I. N., Pradeep, J., Narottam, J., Arvind, K. and Dilip, L. (2016) 'Novel adaptive whale optimization algorithm for global optimization', *Indian Journal of Science and Technology*, Vol. 9, No. 38.

Abed-alguni, B. H. and Alkhateeb, F. (2018) 'Intelligent Hybrid Cuckoo Search and β-hill Climbing Algorithm', *Journal of King Saud University - Computer and Information Sciences*, Vol. 0, No. 0, pp.1–43.

Al-Betar, M. A. (2016) 'β-hill climbing: an exploratory local search', *Neural Computing and Applications*, pp.1–16.

Al-Betar, M. A., Awadallah, M. A., Khader, A. T. and Abdalkareem, Z. A. (2015) 'Island-based harmony search for optimization problems'. *Expert Systems with Applications*, Vol. 42, No.4, pp.2026–2035.

Al-Betar, M. A. and Awadallah, M. A. (2018), 'Island Bat Algorithm for Optimization', *Expert Systems with Applications*.

Lardeux, F. and Goëffon, A. (2010) 'A dynamic island-based genetic algorithms framework', *In Asia-Pacific Conference on Simulated Evolution and Learning, Springer, Berlin, Heidelberg.* pp.156–165.

Michel, R. and Middendorf, M. (1998) 'An island model based ant system with lookahead for the shortest supersequence problem', *In Parallel problem solving from nature PPSN, Springer*, pp.692—701.

Romero, J. F. and Cotta, C. (2005) 'Optimization by island-structured decentralized particle swarms', *In Computational intelligence: Theory and applications, Springer*, pp.25-–33.

Kushida, J. I., Hara, A., Takahama, T. and Kido, A. (2013) 'Island-based differential evolution with varying subpopulation size', *IEEE Sixth International Workshop on Computational Intelligence & Applications (IWCIA)*', pp.119–124.

Corcoran, A. L. and Wainwright, R. L. (1994) 'A parallel island model genetic algorithm for the multiprocessor scheduling problem', *In Proceedings of the ACM symposium on applied computing*, pp.483–487.

Abed-alguni, B. H. and Barhoush, M. (2018) 'Distributed Grey Wolf Optimizer For Numerical Optimization Problems', *Jordanian Journal of Computers and Information Technology* Vo. 4, No. 3, pp.130–149.

Skolicki, Z. (2005). 'An analysis of island models in evolutionary computation', *In Proceedings of genetic and evolutionary computation workshop, ACM*, pp.386-–389.

Tomassini, M. (2005) 'Spatially structured evolutionary algorithms: Artificial evolution in space and time', *Natural Computing Series, Springer.*

Abed-alguni, B. H., Chalup, S. K., Henskens, F. A. and Paul. D. J. (2015) '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.

Abed-alguni, B. H., Chalup, S. K., Henskens, F. A. and Paul. D. J. (2015) '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.

Kushida, J., Hara, A., Takaham, T. and Kido, A. (2013) 'Island-based differential evolution with varying subpopulation size'. *IEEE sixth international workshop on computational intelligence & applications (IWCIA)*, pp.119-–124.

Abed-alguni, B. H. and Klaib A. F. (2018) 'Hybrid whale optimisation and β-hill climbing algorithm for continuous optimisation problems', *International Journal of Computing Science and Mathematics*, Vol. 0, No.0, pp.1–14.

Abed-alguni, B. H. (2017) 'Bat Q-learning Algorithm', *Jordanian Journal of Computers and Information Technology* Vo. 3, No. 1, pp.56–77.

Alkhateeb, F. and Abed-alguni, B. H. (2017) 'A hybrid cuckoo search and simulated annealing algorithm', *Journal of Intelligent Systems* Vol. 0, No. 0, from doi:10.1515/jisys-2017-0268.

Abed-alguni, B. H. and Alkhateeb, F. (2017) 'Novel selection schemes for cuckoo search', *Arabian Journal for Science and Engineering*, Vol. 42, No. 8, pp.3635–3654.

Yang, X. S. and Hossein Gandomi, A. (2012) 'Bat algorithm: a novel approach for global engineering optimization', *Engineering Computations*, Vol. 29, No. 5, pp.464–483.

Abed-alguni, B. H., Paul, D. J., Chalup, S. K. and Henskens, F. A. (2016) 'A comparison study of cooperative q-learning algorithms for independent learners', *International Journal of Artificial Intelligence™* , Vol. 14, No.1, pp.71–93.

Abed-alguni, B. H. (2018) 'Island-based Cuckoo Search with Highly Disruptive Polynomial Mutation', *International Journal of Artificial Intelligence™* , Vol. 0, No.0, pp.1–30.

Yalcin, N., Tezel, G. and Karakuzu, C. (2015) 'Epilepsy diagnosis using artificial neural network learned by PSO', *Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 23, No. 2, pp. 421-432.

Klempka, R. and Filipowicz, B. (2017) 'Comparison of using the genetic algorithm and cuckoo search for multicriteria optimisation with limitation', *Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 25, No. 2, pp. 1300-1310.

Abed-alguni, B. H. (2017), 'Action-selection method for reinforcement learning based on cuckoo search algorithm', *Arabian Journal for Science and Engineering*, pp.1–15.