Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems

Amir Seyyedabbasi [a,*], Royal Aliyev [b], Farzad Kiani [c,e], Murat Ugur Gulle [b], Hasan Basyildiz [b], Mohammed Ahmed Shah [d]

[a] Computer Engineering Department, Faculty of Engineering and Architecture, Beykent University, Istanbul, Turkey
[b] Computer Engineering Department, Faculty of Engineering, Istanbul Aydin University, Istanbul, Turkey
[c] Computer Engineering Department, Faculty of Engineering and Architecture, Istanbul Arel University, Istanbul, Turkey
[d] Computer Engineering Department, Faculty of Engineering, Istanbul Ayvansaray University, Istanbul, Turkey
[e] Software Engineering Department, Faculty of Engineering, Istinye University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 30 October 2020
Received in revised form 11 April 2021
Accepted 12 April 2021
Available online 22 April 2021

Keywords:
Metaheuristic algorithm
Reinforcement learning algorithm
Grey wolf optimization algorithm
Whale optimization algorithm
Q-learning

ABSTRACT

This paper introduces three hybrid algorithms that help in solving global optimization problems using reinforcement learning along with metaheuristic methods. Using the algorithms presented, the search agents try to find a global optimum avoiding the local optima trap. Compared to the classical metaheuristic approaches, the proposed algorithms display higher success in finding new areas as well as exhibiting a more balanced performance while in the exploration and exploitation phases. The algorithms employ reinforcement agents to select an environment based on predefined actions and tasks. A reward and penalty system is used by the agents to discover the environment, done dynamically without following a predetermined model or method. The study makes use of Q-Learning method in all three metaheuristic algorithms, so-called $\mathrm{RL}_{\mathrm{I-GWO}}$, $\mathrm{RL}_{\mathrm{Ex-GWO}}$, and $\mathrm{RL}_{\mathrm{WOA}}$ algorithms, so as to check and control exploration and exploitation with Q-Table. The Q-Table values guide the search agents of the metaheuristic algorithms to select between the exploration and exploitation phases. A control mechanism is used to get the reward and penalty values for each action. The algorithms presented in this paper are simulated over 30 benchmark functions from CEC 2014, 2015 and the results obtained are compared with well-known metaheuristic and hybrid algorithms (GWO, $\mathrm{RL}_{\mathrm{GWO}}$, I-GWO, Ex-GWO, and WOA). The proposed methods have also been applied to the inverse kinematics of the robot arms problem. The results of the used algorithms demonstrate that $\mathrm{RL}_{\mathrm{WOA}}$ provides better solutions for relevant problems.

1. Introduction

Metaheuristic algorithms are a group of nature-inspired optimization algorithms that mimic the behavior exhibited in nature. They try to find the best solution in the global environment of a search space. Metaheuristic algorithms contain two exploration and exploration phases [1–3] and their main goal is to conduct these phases in a balanced manner to avoid early local optima traps [4]. This aim is accomplished in a relatively acceptable time and with relatively less processing [5]. In the exploration phase, the algorithms try to find the best area that offers suitable solutions. The purpose of the exploitation phase is to find the best solution in the explored area. Among well-known optimization methods, Genetic Algorithm (GA) [6], Particle Swarm Optimization (PSO) [7], Harmony Search (HS) [8], Gravitational Search Algorithm (GSA) [9], Grey Wolf Optimization (GWO) [5], Battle Royal Optimization (BRO) [4], Black Widow Optimization (BWO) [10], Whale Optimization Algorithm (WOA) [11], Tunicate Swarm Algorithm (TSA) [12], Lion Optimization Algorithm (LOA) [13], Political Optimizer (PO) [14], Firefly Algorithm (FA) [15], Social Team Building Optimization (STBO) [16], Barnacles Mating Optimizer (BMO) [17], Poor and Rich Optimization (PRO) [18] can be listed. In most of the Non-Deterministic Polynomial-Time (NP-hard) problems, optimization algorithms find the optimal solution in high-dimensional search space problems, but they usually do not guarantee finding an exact solution [19–21]. In addition, No-Free-Lunch (NFL) [22] asserts that there is no specific metaheuristic algorithm that can provide best solution for every optimization problem. This means that if one algorithm can solve one type of problem effectively, this does not imply that it will be effective to solve another kind of problem. As such, there is considerable demand to develop new metaheuristic algorithms that can be used in various problems. Many optimization problems are concerned with extremely expensive

[*] Corresponding author.
E-mail addresses: amirseyyedabbasi@beykent.edu.tr (A. Seyyedabbasi), raliyev@stu.aydin.edu.tr (R. Aliyev), farzad.kiyani@gmail.com (F. Kiani), muratgulle@stu.aydin.edu.tr (M.U. Gulle), hasanbasyildiz@stu.aydin.edu.tr (H. Basyildiz), ahmed.shah@gmail.com (M.A. Shah).

https://doi.org/10.1016/j.knosys.2021.107044
0950-7051/© 2021 Elsevier B.V. All rights reserved.

objective functions. It is well known that with the increase in the dimensions of a problem, its search area grows exponentially, and therefore the complexity increases. Metaheuristic algorithms can provide reasonable solutions in an acceptable time frame. Recently, these algorithms have been applied successfully in solving global optimization problem [23] as well as in various applications such as wireless networks [24], image processing [25], path planning [26], transistor analyses [27], neural networks [28], deep learning [29], and software [30].

In recent years, Machine Learning (ML) algorithms have also come to the fore in solving the mentioned problems [31]. ML methods generally fall into three categories as is shown in Fig. 1; Supervised Learning, Unsupervised Learning, and Reinforcement Learning (RL). In RL algorithms, an agent is placed in a complex environment to learn optimal actions. These algorithms train the agent to perform various tasks in different environments [32]. Reinforcement agent observes the environment and provides feedback to the search space. The agent is trained, afterwards, it uses this training experience in its subsequent actions. Training is carried out through different methods, as a result the agent is expected to be trained in the best way. Besides, the environment is the place where the agent interacts in the system. As shown in Fig. 1, RL methods are generally divided into model-free and model-based approaches, however, in general, model-free techniques are considered [33,34]. Algorithms currently comprising the model-free approaches can be categorized into two subgroups: policy-based, and value-based methods. The value-based algorithms are convenient to work in harmony with metaheuristic algorithms [35]. This is because it is model-free and does not follow a specific policy, providing higher flexibility. In the value-based RL methods, a reinforcement agent does not have any prior knowledge about the environment [35] and it learns to decide based on its actions, and the experience gained, in order to achieve its goal. In these methods, the agents are aware of only the tasks given to them. They have access to no answer keys or model that they can train on, unlike the supervised and unsupervised learning methods. Hence, the agent learns from their experience in the environment through methods, such as reward and penalty. The reward–penalty is a measure of how successful the action is in terms of completing the task goal. Therefore, reinforcement agent makes a decision based on their achievements.

In value-based learning methods, Q-Learning method is one of the famous algorithms. This algorithm is also used in the methods proposed in this study. In the Q-Learning algorithm, the agent tries to find the best action given its current state according to a value-based mechanism. The agent learns from its actions and states. In this algorithm, a predetermined policy is not necessary, since an RL agent takes random actions, then gets a reward or penalty, and gradually an experience is constructed for agent based on its actions that lead to rewards. In the Q-Learning algorithm, a table called Q-Table is defined [36] and the agent tries to update its state to select the best action based on the Q-Table values considering all possible actions that it can take. Therefore, each agent in action decides to either explore or exploit the environment. In methods proposed in this study, the Q-Learning algorithm is used to make the exploration and exploitation phases of metaheuristic algorithms more effective. At the same time, metaheuristic algorithms store the information they obtain about the search space throughout the iteration, which is necessary for Q-Learning. Therefore, this study proposes three hybrid algorithms based on the combination of reinforcement learning and the metaheuristic algorithms that take advantage of both categories while eliminating and/or minimize their deficiencies. Some differences between the metaheuristic and the RL based algorithms can be summarized as follows.

(1) Although the functionality of metaheuristic algorithms is effective in solving NP-hard problems, the design of an appropriate environment and establishing the balance between the exploration and exploitation phases to find the best solutions can often require deep expert knowledge. RL algorithms can be used to discover good designs for parameters of metaheuristic algorithms [5,31,33].
(2) RL-based methods have a high success rate in finding new local areas when compared with metaheuristic algorithms and tend to have a more balanced behavior when switching between related phases. However, the learning process of agent used in RL algorithms may take a long time. Metaheuristic algorithms do not have the learning process and are better in this regard. Therefore, the combination of both tends to provide a good solution to this problem [19,31,33, 37–39].
(3) Metaheuristic methods operate in a static search manner as they use randomness and specific policies in certain situations. Therefore, their dynamism is lower than RL-based methods, especially value-based methods. Because, in a value-based RL method, agent is online and discovers the environment through a reward and penalty mechanism without following any model or method [19,31,33,40].

In the proposed hybrid algorithms, Q-Learning method is implemented in I-GWO [19], Ex-GWO [19], and WOA [11] algorithms, and are named $\mathrm{RL}_{\mathrm{I-GWO}}$, $\mathrm{RL}_{\mathrm{Ex-GWO}}$, and $\mathrm{RL}_{\mathrm{WOA}}$. The aim of each proposed algorithm is to solve global optimization problems as the best solutions without falling into the local optimal trap. It should be noted that the proposed methods do not guarantee the best solution since they do

(1) They try to find a better global solution and explore new possible local areas. The search agents have a chance to find other possible local optima in the desired search space to later aid in finding the global optimal solution.
(2) They use, not only the parameter of the metaheuristic algorithms, but also define another parameter, ensuring the best transition decisions by balancing the exploration and exploitation phases.
(3) The control of two phases to make more effective decisions is handled by Q-Table, which tries to guide search agents in finding new sections of the global search space to be discovered.
(4) In order for the decisions taken by the agent to become more regular and efficient two effective and dynamic parameters (learning rate, discount factor) are used.
(5) They increase the balance ratio between the exploration and exploitation phases. At the same time their ability to switch between exploration and exploitation phases, as and when needed, make them more successful in finding better solutions in the search space of related problems.
(6) They guarantee high performance in exploitation, which ensures the fast convergence of the proposed hybrid algorithm.

The rest of this paper is organized as follows. Section 2 describes the related works. Section 3 explains the detail of the proposed methods. Section 4 reports the obtained results and analysis. Section 5 examines the behavior and the performance of proposed algorithms for inverse kinematics of robot arms. Finally, conclusion and future works are given in the last section.

2. Related works

Metaheuristic and RL-based hybrid studies are summarized in this section. This hybridization, either, uses RL to develop the

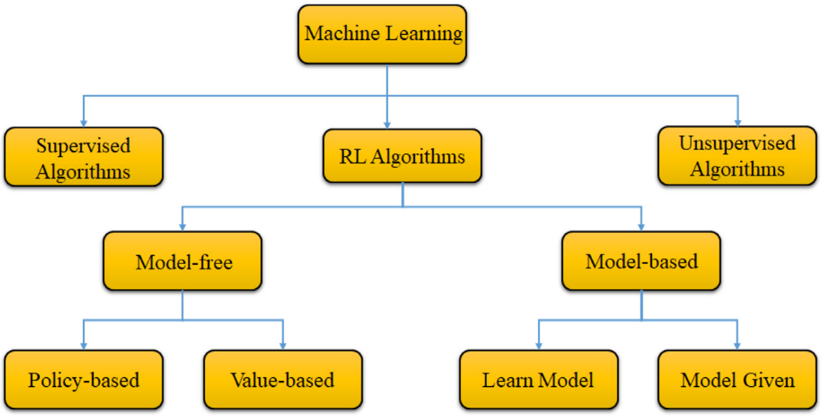


Fig. 1. Taxonomy of the reinforcement learning algorithms.

metaheuristic algorithm or, uses these metaheuristics to improve the performance of RL techniques. Therefore, researchers try to suggest new methods by using the advantages of the one or the other to solve various problems. Solving global optimization problems has attracted the attention of many researchers and as such numerous studies have been proposed in the literature [41–46]. In this paper, a few of the existing hybrid works based on the combinations of the RL and metaheuristic are detailed.

In [47], authors combined the standard PSO and Q-Learning (QL) algorithm. As a result of the QLPSO algorithm, every particle acts as an independent agent and selects the optimal topology controlled by Q-Learning in each iteration. Two QLPSO variants take different dimensions of the communication topology into account. In PSO, the movement of a particle is repeatedly influenced by the best previous solutions along with the best previous neighbor's best solution. In their study, QL has been integrated into PSO to choose the most suitable neighborhood size based on reward from the search regions. Each particle is allowed to act as an agent and is considered as a search area. Another thing to note is that each particle uses its own Q-Table. The status of each particle is visible in neighborhoods that can be selected to form a regular topology structure. Moving from one neighborhood to another is an action, whereas advances in suitability and diversity are the measurements of the reward. During repetitions, each particle acts as a learning substance and selects the best neighborhood within a regular structure. The herd population size is randomly started with an N and a D size. QL algorithm gives the best neighbor topology (action) based on the state of the current particle and instant reward. Each particle then updates its speed and position separately. QLPSO repeats these steps until the maximum number of function evaluations is achieved. Compared with the basic PSO, the main difference of QLPSO is that QL is included in the PSO to optimize performance by improving the neighborhood structure during the search process.

In [48], authors have introduced so-called Gravitational Search Algorithm (GSA-LA) based on Learning Automata (LA). The main reason to develop this algorithm is that the evolutionary algorithms get stuck on local optima. LA is a method that helps in finding optimum action when there is more than one option available. In LA, first, a potential action is chosen randomly from a set of potential actions. This choice is based on a certain distribution probability. Later, the automation signal is used to update the probability distribution for each potential action. The iterative process for updating the signal provides the best answers through learning. GSA is an evolution-based algorithm that is inspired by the theory of gravity. The foundation of GSA is the physical laws of the gravitational force between masses which direct lighter masses to heavier masses. In the GSA, there are four particulars: position, inertial mass, active gravitational mass, and passive gravitational mass. Each mass position corresponds to a candidate solution. The inertial and gravitational mass are calculated using a fitness function. In the GSA-LA process, the G value usually decreases with the decrease in the search area; however, the G trend fluctuates throughout the evolutionary process. The general trend is in line with the "answers over time" concept, which is the main basis of the algorithm. It tries to find the best G value for GSA-LA to find the best solution. Radical changes show that GSA-LA has always recovered from local optima by increasing the search area. Classical GSA is especially stuck in high-dimensional test functions. In this case, it has no course of action that can help it avoid local optima. LA makes it possible for GSA-LA to properly use the G value, thereby helping the algorithm avoid local optima.

In [49], authors proposed an improved genetic algorithm with reinforcement mutation named RMGA to solve Traveling Salesman Problem (TSP). This RMGA algorithm emphasizes the construction of reinforcement mutation operator so-called RL-M using heterogeneous matching selection instead of random matching selection of *EAX* (edge assembly crossover) by changing the Q-Learning algorithm and applying it to the populace produced from modified *EAX*. The *EAX* performs best in terms of inheritance compared to other transitions. This algorithm is especially useful for evolutionary algorithms in solving large TSP. In the proposed method the reward of solution of TSP is dependent on the distance between the two neighboring cities: shorter the distance, higher the reward. The adjustments made to the instance obtained from *EAX* using Q-Learning in RMGA is called a reinforcement mutation. Unlike traditional Q-Learning, the reinforcement mutation considers looking for the next random city in TSP. The authors claim, based on their experimental results, that the proposed algorithm performs well for up to 3000 cities.

In [50], authors proposed a novel hybrid method using reinforcement learning along with grey wolf optimization. In this study, *ExpRate* formula proves to be effective in solving numerous optimization problems. Their match between states and actions is typically nonlinear. For each wolf, 3 actions are suggested to adapt the rate of exploration. (1) Increase the exploration rate: It takes place as a result of the wolf's self-confidence and expertise. This action commonly happens when the wolf finds itself succeeding in consecutive iterations. This increases its own confidence and hence increases its exploration rate. (2) Decrease the exploration rate: Agent's oscillation of fitness may motivate such action and it reflects wrong decision taken by the agent, and hence, it should be cautious in its movements. (3) Keep the exploration rate: The current exploration rate is kept static as there is no motivation for either an increase or a decrease. For state transitions of the agent, the authors propose to change the status of a search tool and then reposition the agent in the search field. This results in a new availability value. In such cases, the history of the fitness values is used to determine the next action. From a feedback-based approach, the agent learns to move from a worst fitness area to an area with a good fitness value providing a positive feedback, while a negative feedback is given in the contrasting case where the agent finds itself in a worst fitness area. In their study, a GWO variant is recommended for each agent (wolf), which learns the rate of discovery individually. Experienced GWO (EGWO) uses reinforcement learning principles to learn what needs to be done in different situations of optimization and in various areas of the search area. A neural network-based model was used to hold experience knowledge, the recommended EGWO was compared to the original GWO, PSO, and GA in feature selection and weight adaptation parameters.

In [51], authors proposed a novel Ant Colony Optimization (ACO) method applied in reinforcement learning methods. Within the ant colony optimization, a family of evolutionary methods is employed to find the shortest path between a pair of nodes in the graph. They use a system of ant iterations termed *nt* with ants in each *nt*. Each ant creates a path from the start node *S* to the target node *T*. In their Sub-goal termed Discovery, aspiring from a simple network world environment, the agent's task is to reach goal states *g* from some starting states. In this environment, a door is a sub-goal and there are two rational possibilities, one that helps the agent in reaching the *door*, and the other helps it get closer to the *door*. Upon applying the Ant System method to the transition graph for this environment, after several iterations of route creation from the beginning to the target node, a shorter path to the target node is found. In transitional graphs, which have a clear social structure, there are some edges, so-called bottleneck edges, on the shortest path that act as a bridge between neighboring societies. The main step in the skill detection algorithm is to determine the edges of the bottleneck. Using the definition of roughness, each edge is awarded a degree of bottleneck. When edges are sorted based on the shortest path and the roughness values in the list, the edges of the bottleneck have the least roughness. The next step is to disconnect the edges of the bottleneck from the others on the list. In the proposed method they analyze roughness growth in the list of edges classified in the shortest path. The place where meaningful slope increases are found by the stroke of bottleneck and edges other than a bottleneck.

In [52], authors proposed an RL-based GWO method to find the optimal path for Unmanned Aerial Vehicles (UAVs). Their proposed method has four operations: *exploration*, *exploitation*, *geometric adjustment*, and *optimal adjustment*, while changing the state is an action. According to cumulative performance, the individual state switching process is adaptive. If the execution of the operation leads to better performance, a positive reward is awarded, in the contrary scenario, a negative feedback is given. In this way, the Q-Table is updated. In their work, columns of the Q-Table represent the actions and the rows illustrate the states (exploration, exploitation, geometric adjustment, and optimal adjustment). Every agent has its own Q-Table, ensuring the independence of the learning process. In the exploration process, affinity parameter *aw* is set to a high value, forcing an individual agent to perform a global search. Moreover, an agent is influenced by three best current agents, working uniformly in exploration mode to weaken the effect of the alpha wolf α. In the process of exploitation, all agents move slowly towards the current best individual. Therefore, *aw* is set with a very low ensuring local search around α. They also make sure that every agent gets affected primarily by the alpha wolf during the exploitation mode. The movements in geometric adjustment operation of agents are influenced by the lead group and the *aw* parameter, while the distribution of an agent items is scattered based on appropriate fitness. When an agent is viewed as a set of coordinates, the viability and safety of the UAVs flight path are greatly affected. Moreover, the optimal adjustment operation on the problem of three-dimensional UAVs route planning is performed in [52]. They benefit from the fact that the best flight path is often associated with the straight path between the starting point and the target point. When obstacles are present, optimal path is the appropriate path near the straight path. This process updates all dimensions of the current individual, starting with the (y-axis) and elements (the ophthalmic axis) corresponding to the first coordinate, moving them along the direction of the straight path. If a better result is obtained every time, the element is refreshed, otherwise, the model remains unchanged, and each waypoint is triggered.

3. Proposed methods

Metaheuristic methods are very successful in solving NP-hard and complex problems to find solutions of good quality in a reasonable time [5,19]. The search agents in these algorithms find the best solution based on the goal of the optimization problem. Typically, the best solution is the maximum or minimum score defined as a global optimum. In order to achieve the global optimum point, search agents have to move through all the points while ensuring no point exists with a better optimum score than the selected point. Since the search agents search from a random point, the algorithm guides the search agent in their quest. Sometimes, these agents get trapped on a local optimum confusing it with the global optimum. Therefore, the agent needs to search further in the search space if it is to find the global optimum. Exploration and exploitation phases are employed for this purpose. With metaheuristic algorithms, search agents explore the search space during the exploration phase, whereas, they decide upon the best solution during the exploitation phase. It is important to note that a balance between these two phases is of utmost importance.

In line with the purposes and reasons explained in Section 1, this study proposes three hybrid algorithms based on the combination of reinforcement learning and metaheuristic algorithms. In this study, Q-Learning method is applied on the I-GWO, Ex-GWO, and WOA algorithms, so-called $\mathrm{RL}_{\mathrm{I-GWO}}$, $\mathrm{RL}_{\mathrm{Ex-GWO}}$, $\mathrm{RL}_{\mathrm{WOA}}$, in order to control exploration and exploitation using the Q-Table and to solve global optimization problems. The reinforcement agent uses the Q-Table values as guide to the search agents of metaheuristic algorithms to decide between the exploration or exploitation phases, being the two actions possible. This table is updated using a reward and penalty mechanism. Details of the proposed methods is explained Section 3.3. First, the three metaheuristic algorithms used are summarized.

3.1. I-GWO and Ex-GWO

This subsection describes I-GWO and Ex-GWO algorithms, which are used in the proposed hybrid methods. Both algorithms are inspired from the grey wolves' behavior in their natural habitat. In nature there are four types of wolves in a pack: Alpha (α), beta (β), delta (δ), and omega (ω). Each wolf has different responsibilities in its pack. α wolf is the leader of the pack whether male or female. β wolf is the leader of the pack when α is absent, besides the β wolf helps the α wolf in commands and feedbacks. The third type of wolf in the hierarchy is the δ wolf that helps the pack in maintaining the dominance structure. The wolves that do not belong to the aforementioned three levels are termed as ω wolves [5]. The social behavior of the grey wolves,

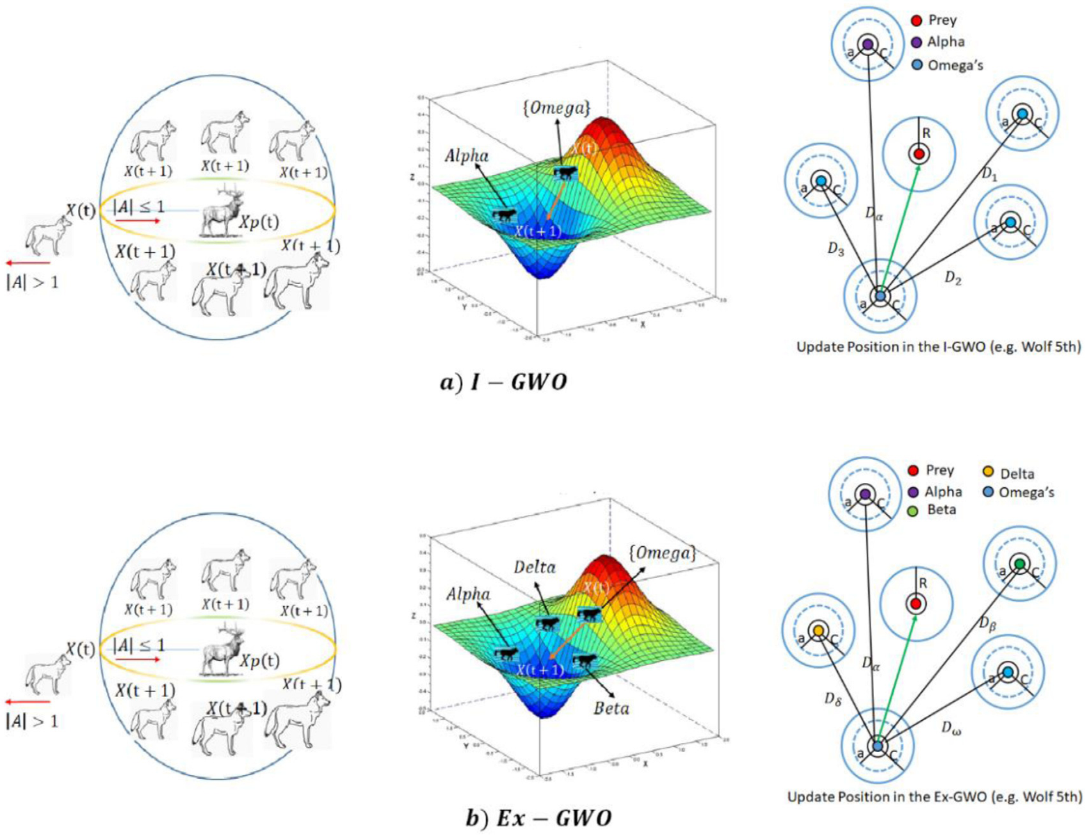


Fig. 2. Working mechanism considering exploration and exploitation (a) I-GWO, (b) Ex-GWO.

i.e., *encircling*, *hunting*, and *tracking* a prey are simulated using GWO algorithm. Fig. 2 shows the working mechanisms of both algorithms considering their *exploration* and *exploitation* phases.

Eqs. (1) and (2) have been proposed to model a pray being encircled in I-GWO and Ex-GWO. Both equations are applied to the respective algorithms in a similar fashion. In these equations: t indicates the current iteration, T is the maximum number of iterations, $\vec{X_p}$ is the position vector of the prey, $\vec{X}$ indicates the position vector of a wolf. Furthermore, D is a vector dependent on the prey's location. The coefficient vectors $\vec{A}$, and $\vec{C}$ are considered to lead in encircling their prey (Eqs. (3) and (4)). These parameters control the tradeoff between exploration and exploitation phase in both I-GWO and Ex-GWO. It should be noted that $\vec{a}$ linearly decreases from 2 to 0 over the course of iterations. This is done to get closer to the solution range. r_1 and r_2 are the random vectors in a range of [0,1]. $\vec{a}$ is used to improve the convergence speed, defined by Eq. (5) for I-GWO and Eq. (6) for Ex-GWO algorithms respectively. There are two possible states for each leader (alpha) in the pack; attack or search. When $|\vec{A}|$ is less than 1, the wolves in the pack attack to hunt, otherwise they try to find a prey to be hunted. In this way, these algorithms try to find possible solutions in the whole area.

The first metaheuristic algorithm employed in this study is I-GWO. The main difference between GWO and I-GWO lies in the hunting mechanism. The I-GWO assumes that the first wolf (alpha) has better knowledge of the preys' position. Other wolves in the pack follow the $n-1$ wolves before them so as to update their position relative to the prey. The α wolf guides the remaining wolves in the pack in hunting. Given this scenario, when the first wolf (α) is in the best position to the prey, the other wolves also follow the best path, otherwise, the remaining wolves get farther away from the prey (Eqs. (7) to (9)). The success rate of the I-GWO may be not suitable in finding best solutions to some complex problems because it is heavily dependent on the position of the first wolf (α).

$$\vec{\mathrm{D}} = \left|\vec{\mathrm{C}} \cdot \vec{\mathrm{X}_\mathrm{p}}\,(\mathrm{t}) - \vec{\mathrm{X}}\,(\mathrm{t})\right| \tag{1}$$

$$\vec{\mathrm{X}}\,(\mathrm{t}+1) = \vec{\mathrm{X}}_\mathrm{p}\,(\mathrm{t}) - \vec{\mathrm{A}} \cdot \vec{\mathrm{D}} \tag{2}$$

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

$$\vec{C} = 2 \cdot \vec{r_2} \tag{4}$$

$$\vec{a} = 2\left(1 - \frac{t^2}{T^2}\right) \tag{5}$$

$$\vec{a} = 2\left(1 - \frac{t}{T}\right) \tag{6}$$

$$\vec{D_\alpha} = \left|\vec{C_\alpha} \cdot \vec{X_\alpha} - \vec{X}\right| \tag{7}$$

$$\vec{X_\alpha} = \vec{X}_\alpha - \vec{A_\alpha} \cdot \vec{D_\alpha} \tag{8}$$

$$\vec{X_n}\,(t+1) = \frac{1}{n-1}\sum_{i=1}^{n-1} X_i\,(t)\,;\, n = 2, 3, \ldots, m \tag{9}$$

The second metaheuristic algorithm used in this paper is Ex-GWO which is a modified version of GWO algorithm. In the Ex-GWO, two-level hierarchy is proposed in the pack, where the first three grey wolves (alpha, beta and delta) are in the first level of the hierarchy and the remaining wolves (omega) are in the second. This method has a strong effect on the hunting process, particularly in crowded environments. In this method, the nth wolf (n = 4, 5, …, m) updates its own position based on the first three wolves and the $n-1$ wolf before it. As such, the wolves in the pack benefit from expanded knowledge regarding their preys' position. In the hunting mechanism of Ex-GWO, the wolves try to find the best solution by nearing the prey. Ex-GWO algorithm is

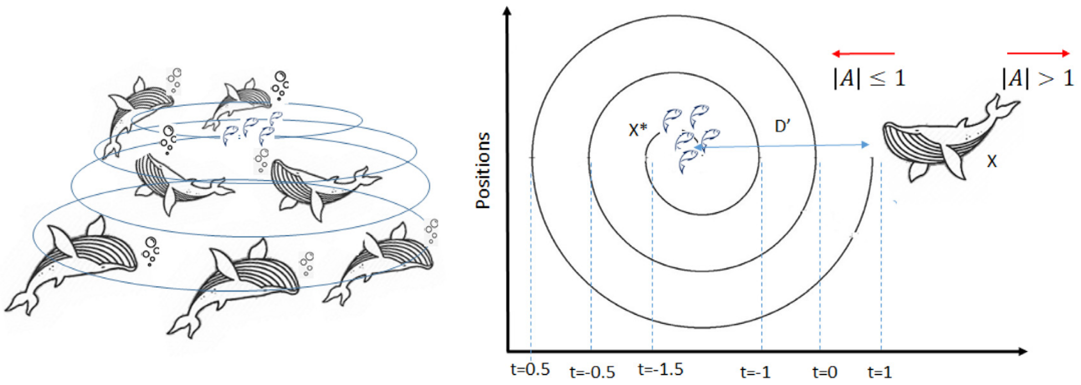


Fig. 3. Working mechanism considering exploration and exploitation of WOA [53].

represented mathematically in (Eqs. (10)–(12)).

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \quad \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \quad \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \tag{10}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \tag{11}$$

$$\vec{X}_n\,(t+1) = \frac{1}{n-1}\sum_{i=1}^{n-1} X_i\,(t)\,;\ \mathrm{n} = 4, 5, \ldots, \mathrm{m} \tag{12}$$

3.2. WOA

The WOA is the third nature-inspired metaheuristic algorithm used in the proposed methods. It mimics the hunting mechanism of whales (exploration) and uses a spiral bubble-net attacking mechanism of humpback whales in finding the prey (exploitation). Similar to most metaheuristic algorithms, in the exploration phase, search agents are in the global search for optimal solution whereas in the exploitation phase they perform a local search. There are three phases in the hunting mechanism for WOA namely, encircling, searching (for exploring), and attacking (for exploiting) the prey. In the encircling phase, each whale updates its position around the current best search agent. (Eqs. (13) and (14)). In these equations $\vec{X}^*$ is the position vector of the best solution obtained so far, and $\vec{X}$ is the position vector. The functionality and purpose of the $\vec{A}$, $\vec{C}$, and a parameters are the same as in the GWO. There are two methods in the exploitation phase (attacking of prey). A random value (p), assigned between 0 and 1, determines the performance of each of these mechanisms. Shrinking encircling and spiral updating position mechanisms are chosen from Eq. (17). Where D' indicates the distance between the whale's prey and its best solution so far. In addition, b is a constant used to define the shape of the logarithmic spiral, l is a random number in [−1,1]. The exploration phase (searching of prey) is also determined by a vector $\vec{A}$. If the value of |A| is greater than 1, search agents are forced to find other global spaces (Eqs. (18) and (19)). Where $\vec{X}_{rand}$ is a random position vector (a random whale) chosen from the current population. The working mechanism considering exploration and exploitation phases of the WOA is represented in Fig. 3.

$$\vec{D} = \left|\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)\right| \tag{13}$$

$$\vec{X}\,(t+1) = \vec{X}^*\,(t) - \vec{A} \cdot \vec{D} \tag{14}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{15}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{16}$$

$$\vec{X}\,(t+1) = \begin{cases} \vec{X}^*\,(t+1) - \vec{A} \cdot \vec{A} & \text{if } p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \tag{17}$$

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_{rand} - \vec{X}\right| \tag{18}$$

$$\vec{X}\,(t+1) = \vec{X}_{rand} - \vec{A} \cdot \vec{D} \tag{19}$$

3.3. Development of proposed hybrid algorithms: RL_{I-GWO}, RL_{Ex-GWO}, RL_{WOA}

Q-learning algorithm, a value-based learning algorithm, is used in each hybrid algorithm proposed in this study. This is because Q-learning algorithm is very convenient to work in coordination with metaheuristic algorithms [31,39]. In Q-Learning, main task assigned to an agent is updating its states by taking actions with the highest Q-values at every step. In other words, the best state is selected, among the states found, after calculating the degree of benefit that results from being in each of the possible states, for the next step. After each step, a reward or a penalty is assigned to the agent based on its actions. Consequently, the agent updates its Q-Table and prepares the table for the next episode using the Bellman Equation (Eq. (20)). The general schema of the reinforcement learning agent and environment architecture is shown in Fig. 4.

$$Q_{(t+1)}\,(s_t, a_t) \leftarrow Q_t\,(s_t, a_t) + \lambda\left[r_{t+1} + \gamma MaxQ_{t\,(s_{t+1},a)} - Q_t\,(s_t, a_t)\right] \tag{20}$$

where s_t and s_{t+1} represent the current and the next states relatively, a_t is the current action. λ is learning rate value and γ is the discount factor. In addition, λ and γ are a number between 0 and 1. The λ defines how fast our algorithm should learn whereas γ defines how much our algorithm learns from its mistakes. Indeed, the learning rate determines the extent to which newly acquired information overrides old information and is used to control the convergence of learning process. On the other hand, the discount factor determines the importance of future rewards. Indeed, the discount factor is a constant that represents how the agent will pay attention to rewards in the distant future. It controls the importance of long-term rewards. r_{t+1} is the immediate reward an agent is awarded or the amount it is penalized for taking current action. Q_{t+1} is the Q-value pre-estimated for the next state s_{t+1}. Hence, for solving the problem using Q-Learning there are three essential components of this algorithm: *Reward Table, Q-Table,* and *Bellman Equation*. Using all these components, a particular problem can be solved as shown in Algorithm 1.

In the Q-Learning algorithm, reward table matrix is used for penalizing or rewarding an agent for its action/state compositions. This table can be provided by end-user, or similar to the experiments performed in this study, a reward table can be created using programming techniques based on the architecture

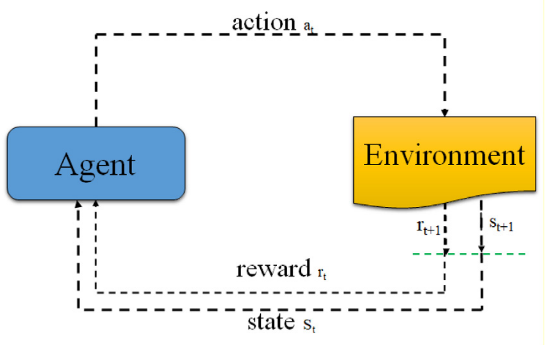


Fig. 4. Reinforcement learning agent and environment architecture [34].

of the solving methods. The Q-Table matrix can be assumed as experience of an agent. During the discovering phase, an agent does not have any knowledge or experience about the environment that it belongs to. Expressing this mathematically, initially all units of the Q-Table matrix are assigned a zero value (concisely stating it is zero matrices). Each agent gains experience as they explore the environment in specific episodes/iterations and update the corresponding Q-Table matrix with a third component (Bellman equation). Using this equation, the agent evaluates the likeliness of getting closer to the solution by weighing every possible next state and each possible action that can be taken from that state.

In all the hybrid algorithm proposed in this study, the two tables (*Reward Table* and *Q-Table*) are defined as two-dimensional matrices. The reward matrix contains the positive (+1) or negative (−1) reward for each state and action couples, while values inside the Q-Table matrix describe the result of choices that are taken by the RL agent in the previous episodes. The relevant reward table and Q-Table are shown in Fig. 5. The proposed mechanism helps in finding the most appropriate decision that can be made during exploring and exploiting phases. One of the following four options can take place in line with the most appropriate decision taken.

(1) While the agent is in the explore state, it decides to stay in explore (action)
(2) While the agent is in the exploit state, it decides to stay in exploit (action)
(3) While the agent is in the explore state, it decides to transit to exploit (action)
(4) While the agent is in the exploit state, it decides to transit to explore (action)

As mentioned earlier, |A| parameter controls both the exploration and the exploitation phases I-GWO, Ex-GWO, and WOA algorithms. Value of |A| is used to detect if the agent currently explores the prey in its search space or exploits it. However, in the proposed hybrid algorithms, the Q-Table is also included along with |A|, in the account and the most appropriate exploration or exploitation decision are made according to the values in this table. Indeed, in the proposed hybrid algorithms, |A| parameter checks the phases and Q-value controls and then allows the most appropriate decision to be made for exploring and exploiting. When the most appropriate decision is made, it is applied to the fitness function of the relevant problem or application. This whole process continues until it either finds the best possible solution or reaches the end of the defined iterations. In proposed method, |A| parameter value and Q-values are compared to update the position for each search agent in the metaheuristic algorithms employed. In this step, the new fitness value follows two mechanisms as are calculated using the respective equations. If the Q-value of the exploration phase is less than the Q-value of the exploitation value, Eqs. (21) to (23) are used, otherwise, Eqs. (9), (12), (17) are used. Similarly, current fitness value and new fitness value are compared to assign the reward value. If the new fitness value is better than the current fitness value, reinforcement agent gets a positive reward (+1), otherwise, as a penalty the reward is negative (−1). In $\mathrm{RL_{I-GWO}}$, $\mathrm{RL_{Ex-GWO}}$, and $\mathrm{RL_{WOA}}$, fitness value is calculated utilizing Eqs. (21), (22), and (23), respectively. Where σ is weighting parameter that is a value between 0 and 1.

Accordingly, another important point considered is how much and for how long reinforcement agent is allowed to be trained, as well as the extent to which whatever it learned is effective in making its next decisions. Therefore, the agent in the decision-making process needs to be trained. In order to make the decisions taken more regular and efficient, the rewards and penalties to be received by the relevant agent should be more effective, assuming the discount factor value is slightly higher. On the other hand, the learning rate parameter is started at a high value and is decreased as the iterations progress. This way, this rate for the agent that is just starting to learn is high in the first iterations, but with time this agent will be able to use more and more of its past experiences.

The most distinctive feature of the proposed method is its ability to switch between exploration and exploitation phases as and when needed. Due to this, it can be more successful in finding better solutions in the search space of related problems. Besides, they will be able to find the best solutions faster and be able to explore and exploit in more effective local areas. Therefore, they are likely to find solutions in an acceptable time and in minimum number of iterations. This feature can be considered as the most significant difference when compared to other metaheuristic algorithms. However, it should be kept in mind that the proposed methods do not guarantee the best solution since they do not employ deterministic approaches as is the case in metaheuristic algorithms. A schematic example of the proposed methods is presented in Fig. 6. The pseudocode and flowchart of the proposed algorithm are presented in Algorithm 2 and Fig. 7, respectively.

$$X_i\left(t+1\right) = \frac{1}{n-1}\sum_{i=1}^{n-1}\sigma_i X_i\left(t\right);\ \mathrm{n} = 2, 3, \ldots \mathrm{m} \tag{21}$$

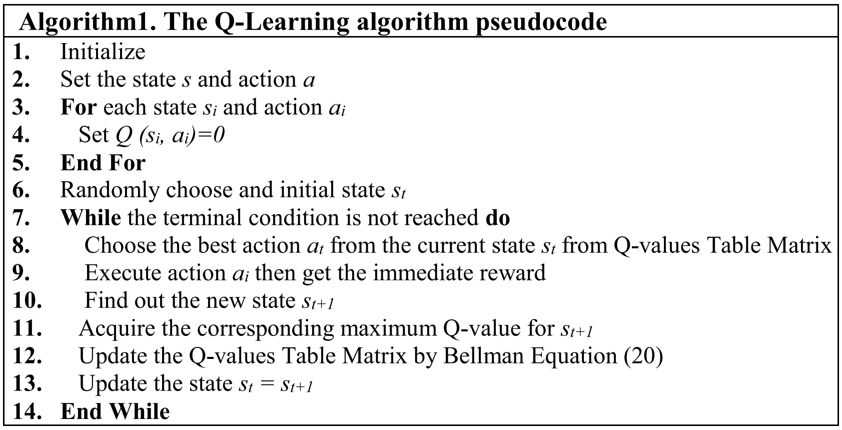
$$\vec{X}_n\left(t+1\right) = \frac{1}{n-1}\sum_{i=1}^{n-1}\sigma_i X_i\left(t\right);\ \mathrm{n} = 4, 5, \ldots \mathrm{m} \tag{22}$$

$$\vec{X}\left(t+1\right) = \begin{cases} \sigma\cdot\vec{X}^{*}(t) - \vec{A}\cdot\vec{D} & if\ p < 0.5 \\ \sigma\cdot\left(D'\cdot e^{bl}\cdot\cos(2\pi l) + \vec{X}^{*}(t)\right) & if\ p \geq 0.5 \end{cases} \tag{23}$$

4. Results and discussion

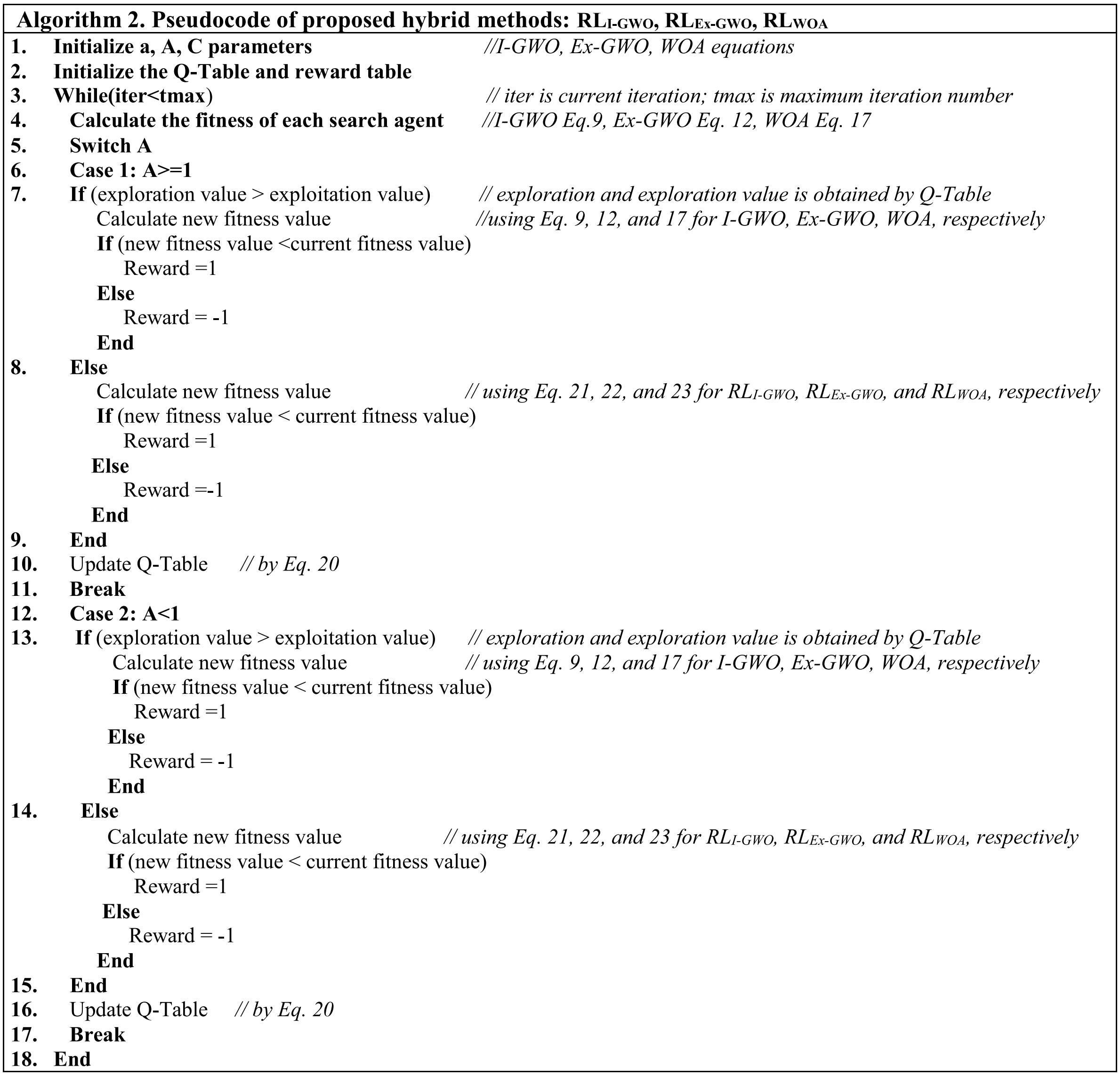
4.1. Experiment setup

In this section, the simulation parameters and benchmark functions information are detailed. The algorithms proposed in this study are simulated using MATLAB and are performed on a Core i7-5500 U 2.4 processor with 8 GB of RAM. The three proposed algorithms ($\mathrm{RL_{I-GWO}}$, $\mathrm{RL_{Ex-GWO}}$, and $\mathrm{RL_{WOA}}$) are compared with five algorithms in the literature (GWO [5], $\mathrm{RL_{GWO}}$ [52], I-GWO [19], Ex-GWO [19], and WOA [11]). Each of the eight algorithms is simulated under similar conditions with 10 independent runs consisting of 30 search agents and 500 iteration numbers. Ten (10) independent runs are performed to manage the effects generated from random parameters in the methods used. The value of the learning rate parameter is between 0 and 1. When this value is 0, it ensures that the agent does not learn



Algorithm1. The Q-Learning algorithm pseudocode

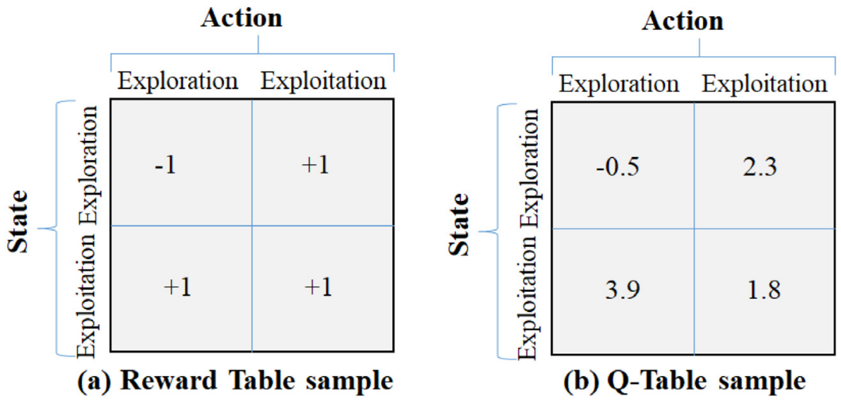
```
Initialize
Set the state $s$ and action $a$
For each state $s_i$ and action $a_i$
  Set $Q (s_i, a_i)=0$
End For
Randomly choose and initial state $s_t$
While the terminal condition is not reached do
   Choose the best action $a_t$ from the current state $s_t$ from Q-values Table Matrix
   Execute action $a_i$ then get the immediate reward
   Find out the new state $s_{t+1}$
  Acquire the corresponding maximum Q-value for $s_{t+1}$
  Update the Q-values Table Matrix by Bellman Equation (20)
  Update the state $s_t = s_{t+1}$
End While
```



Algorithm 2. Pseudocode of proposed hybrid methods: RL_{I-GWO}, RL_{Ex-GWO}, RL_{WOA}

```
Initialize a, A, C parameters                    //I-GWO, Ex-GWO, WOA equations
Initialize the Q-Table and reward table
While(iter<tmax)                                  // iter is current iteration; tmax is maximum iteration number
  Calculate the fitness of each search agent      //I-GWO Eq.9, Ex-GWO Eq. 12, WOA Eq. 17
  Switch A
  Case 1: A>=1
  If (exploration value > exploitation value)     // exploration and exploration value is obtained by Q-Table
      Calculate new fitness value                 //using Eq. 9, 12, and 17 for I-GWO, Ex-GWO, WOA, respectively
      If (new fitness value <current fitness value)
         Reward =1
      Else
         Reward = -1
      End
  Else
      Calculate new fitness value                 // using Eq. 21, 22, and 23 for $RL_{I-GWO}$, $RL_{Ex-GWO}$, and $RL_{WOA}$, respectively
      If (new fitness value < current fitness value)
         Reward =1
     Else
         Reward =-1
     End
  End
  Update Q-Table      // by Eq. 20
  Break
  Case 2: A<1
   If (exploration value > exploitation value)    // exploration and exploration value is obtained by Q-Table
        Calculate new fitness value               // using Eq. 9, 12, and 17 for I-GWO, Ex-GWO, WOA, respectively
        If (new fitness value < current fitness value)
           Reward =1
       Else
          Reward = -1
       End
   Else
       Calculate new fitness value                // using Eq. 21, 22, and 23 for $RL_{I-GWO}$, $RL_{Ex-GWO}$, and $RL_{WOA}$, respectively
       If (new fitness value < current fitness value)
           Reward =1
      Else
          Reward = -1
      End
  End
  Update Q-Table     // by Eq. 20
  Break
End
```

anything, so it will only benefit from previous information. On the other hand, when this value is selected 1, it will only consider the most recent information that is learned. In the proposed methods, learning rate parameter has the highest value in the first iteration and it decreases as iterations progress. This enables more learning in the first iterations and the learning rate goes



(a) Reward Table sample

State \ Action	Exploration	Exploitation
Exploration	-1	+1
Exploitation	+1	+1

(b) Q-Table sample

State \ Action	Exploration	Exploitation
Exploration	-0.5	2.3
Exploitation	3.9	1.8

Fig. 5. The reward table and Q-Table example of the proposed method.

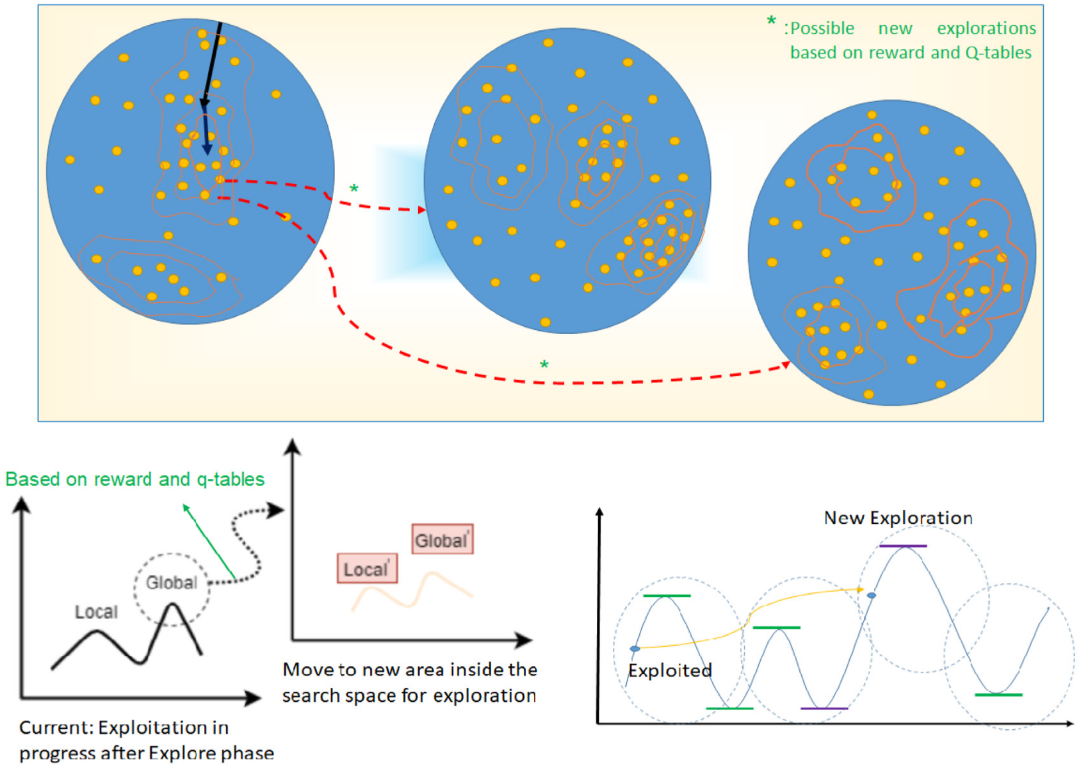

* :Possible new explorations based on reward and Q-tables
*
*
Based on reward and q-tables
Global
Local
Current: Exploitation in progress after Explore phase
Global
Local
Move to new area inside the search space for exploration
New Exploration
Exploited

Fig. 6. A schematic of the proposed method considering exploration and exploitation.

down as experience increases (the learning process depending on the progress). Hence, in the proposed algorithms, it starts at 0.9 in the first iteration and later on this value reduces by a factor of $\lambda - (2\lambda/i - 1)$ in subsequent iterations. The lowest value it can have is limited to 0.1. Where i is the iteration number. In addition, gamma value is assumed 0.8. When gamma is 0, the agent acts narrowly, using only the available rewards. Gamma value is close to 1 in cases where long-term rewards are important. This value was set to 0.8 based on experimental tests and as is the case in similar studies [47,54–57]. The value of σ is between 0 and 1, and in cases where it is required to have more than one weight. Note that the σ values must be assigned so that their sum is 1. In RL$_{\mathrm{GWO}}$, the sum of $\sigma_1, \sigma_2, \sigma_3$ are 1 where σ_1 is 0.6, σ_2 is 0.3, and σ_3 is 0.1. The reduced weight indicates that the wolves are affected according to their hierarchy in the relevant operation. These weight values were assigned by trial and error according to their importance. In the case in RL$_{\mathrm{I-GWO}}$, σ_i depends on the number of wolves in a pack, it should be noted that by increasing the iteration steps of the algorithm, the number of grey wolves' changes. During the first iteration, i is 1 then $\sigma_i = 1/i$. In the second iteration, σ_1 is 0.7 and σ_2 is 0.3. In I-GWO, the first wolf (α) has a higher impact on the pack, so from the third iteration, σ_1 is 0.5 and $\sigma_i = 0.5/(i-1)$, where i is the iteration numbers ($i > 2$). In the RL$_{\mathrm{Ex-GWO}}$, $\sigma_1, \sigma_2, \sigma_3$ are 0.6, 0.3, and 0.1, respectively. After the third iteration, the σ_i value follows the $\sigma_i = 1/i$ for the remaining wolves in the pack. The RL$_{\mathrm{WOA}}$ has a constant value for all σ_i, which is assigned 0.6. For each of the proposed methods, these values have been assigned through the experimental method relying on trial and error techniques. The results of the proposed algorithms in this study and their comparison with the other five algorithms are detailed in two sections of this paper. One of them being the benchmark functions and the other, the Kinematic problem.

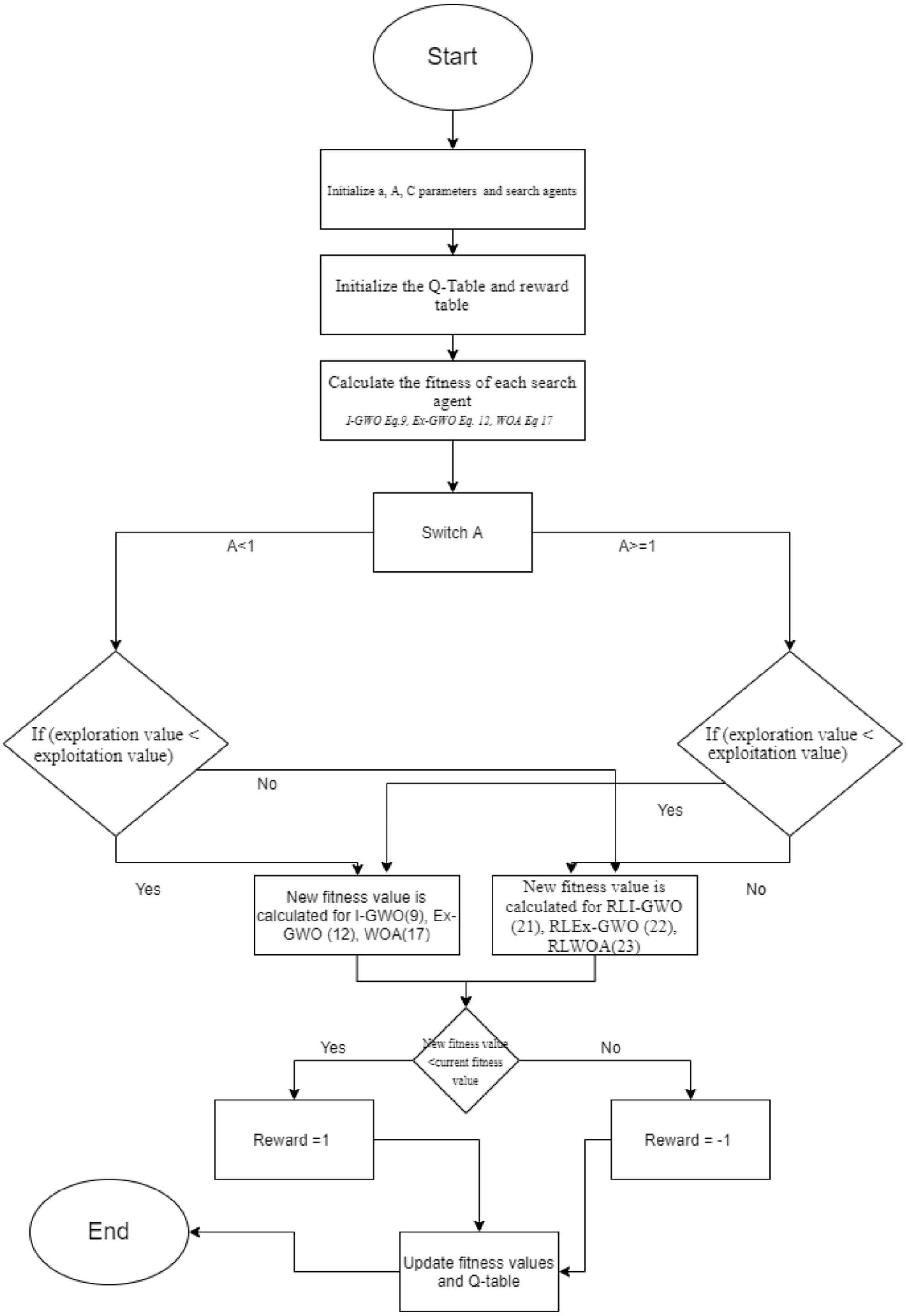
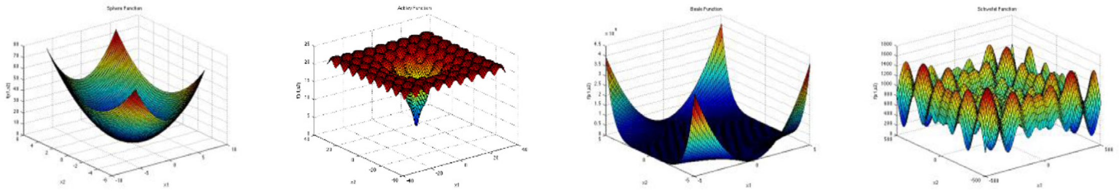

Start
Initialize a, A, C parameters and search agents
Initialize the Q-Table and reward table
Calculate the fitness of each search agent
I-GWO Eq.9, Ex-GWO Eq. 12, WOA Eq 17
Switch A
A<1
A>=1
If (exploration value < exploitation value)
If (exploration value < exploitation value)
No
Yes
Yes
No
New fitness value is calculated for I-GWO(9), Ex-GWO (12), WOA(17)
New fitness value is calculated for RLI-GWO (21), RLEx-GWO (22), RLWOA(23)
Yes
New fitness value <current fitness value
No
Reward =1
Reward = -1
End
Update fitness values and Q-table

Fig. 7. The flowchart of the proposed hybrid algorithms.



a) sphere function b) ackley function c) beale function d) schwef function

Fig. 8. Typical 2D representations of various benchmark functions [5].

Table 1
Benchmark functions used in the current study.

Func.	Benchmark Function	Formula	Dim	Range	Optima
F1	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
F2	Schwefel 2.22	$f_2\,(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10,10]	0
F3	Schwefel 1.2	$f_3\,(x) = \sum_{i=0}^{n-1} \{ \sum_{j=0}^{j<i} x_i \}^2$	30	[−100,100]	0
F4	Schwefel 2.21	$f_4\,(x) = max_i \{ \lvert x_i, 1 \le i \le n \}$	30	[−100,100]	0
F5	Generalized Rosenbrock	$f_5\,(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	30	[−30,30]	0
F6	STEP	$f_6\,(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
F7	Quartic	$f_7\,(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[−1.28,1.28]	0
F8	Generalized Schwefel	$f_8\,(x) = \sum_{i=1}^{n} -xi \sin(\sqrt{\lvert x_i \rvert})$	30	[−500,500]	−418.9829*5
F9	Rastrigin	$f_9\,(x) = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	[−5.12, 5.12]	0
F10	Ackley	$f_{10}\,(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	30	[−32,32]	0
F11	Griewank	$f_{11}\,(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{X_i}{\sqrt{i}} \right) + 1$	30	[−600,600]	0
F12	Generalized Penalized	$f_{12}\,(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10 \sin^2 (\pi y_{1+1}) \right] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u\,(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u\,(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < 1 \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[−50,50]	0
F13	Generalized Penalized	$f_{13}\,(x) = 0.1 \left\{ \sin^2 (3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2 (3\pi x_i + 1) \right] + (x_n - 1)^2 \left[1 + \sin^2 (2\pi x_{ni}) \right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50,50]	0
F14	Shekel's Foxholes	$f_{14}\,(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	[−65,65]	1
F15	Kowalik's	$f_{15}\,(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[−5,5]	0.00030
F16	Six-Hump Camel-Back	$f_{16}\,(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
F17	Branin	$f_{17}\,(x) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	[−5,5]	0.398
F18	Goldstein Price	$f_{18}\,(x) = \left[1 + (x_1 + x_2 + 1)^2 \times \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2 \right) \right] . [30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2 \right)]$	2	[−2,2]	3

(*continued on next page*)

Table 1 (*continued*).

Func.	Benchmark Function	Formula	Dim	Range	Optima
F19	Hartman's Family	$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3	[1, 3]	−3.86
F20	Hartman's Family	$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	[0, 1]	−3.32
F21	Alpine	$f_{21}(x) = \sum_{i=0}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	30	[−10,10]	0
F22	Beale	$f_{22}(x) = (1.5 - x_0 + x_0x_1)^2 + \left(2.25 - x_0 + x_0x_1^2\right)^2 + \left(2.625 - x_0 + x_0x_1^2\right)^2$	2	[−4.5,4.5]	0
F23	Cigar	$f_{23}(x) = x_0^2 + \sum_{i=1}^{n} x_i^2$	30	[−10,10]	0
F24	Matyas	$f_{24}(x) = 0.25\left(x_0^2 + x_1^2\right) - 0.48x_0x_1$	2	[−10,10]	0
F25	Michalewicz	$f_{25}(x) = -\sum_{i=0}^{n} \sin(x_i)\, sin^{20}\left(\frac{ix_i^2}{\pi}\right)^{2m}$; $m = 10$	10	[0, π]	−9.66015
F26	Booth	$f_{26}(x) = (x_0 + 2x_1 - 7)^2 + (2x_0 + x_1 - 5)^2$	2	[−10,10]	0
F27	Easom	$f_{27}(x) = -\cos(x_0)\cos(x_1)\exp(-(x_0 - \Omega)^2 - (x_1 - \Omega)^2)$	2	[−100,100]	−1
F28	Sum Squares	$f_{28}(x) = \sum_{i=0}^{n-1} ix_i$	30	[−10,10]	0
F29	Egg crate	$f_{29}(x) = x^2 + y^2 + 25(Sin_x^2 + Cos_y^2)$	30	[0,10], [-2π, 2π]	0
F30	Zettl	$f_{30}(x) = (x_i^2 + x_{i+1}^2 - 2x_i)^2 + 0.25x_i$	30	[−5,5]	−0.00379

All of the three proposed algorithms are evaluated using 30-benchmark functions and results are compared with five well-known algorithms, available in the literature. These benchmark functions are chosen from CEC 2014 and 2015 [58,59]. The first 21 benchmark functions are the classical functions utilized by many researchers [5,60–64]. The last 9 functions have been recently discussed in new studies [4,10,19,64]. The variety of the selected functions has been increased to increase the accuracy of, both, the results obtained and the analysis performed. Generally, the selected benchmark functions are divided into unimodal, multimodal, fixed-dimension multimodal, and composite function groups [5,11,19,60–63]. In this study, this variety of richness has been used. Unimodal functions have one global optimum and no local optima. Multimodal functions have more than one local optima. While these can be of various sizes, the fix-dimensional ones are fixed-size function types, as highlighted by their name. New optimization algorithms or their enhanced versions are usually tested on any benchmark function. This way, performance of proposed algorithms in various problems can be monitored. Each benchmark function has a specific dimension size, moreover, the boundary is important in the benchmark function. The range of boundary indicates the lower and upper bounds of the search space. The details of all functions are presented in Table 1. Furthermore, the 2D surface of the relevant benchmark is presented in Fig. 8.

4.2. Results analysis

Experimental results, of each proposed algorithm, are presented and discussed in this section. Tables 2, 3, and 4 present the results obtained after 10 times run for each algorithm. The performance of each algorithm is calculated using the mean and standard deviation in each respective benchmark function. The best results are illustrated in bold in the relevant tables. If more than one best solution is found for a benchmark function, the score is equally divided among the algorithms. For example, all three algorithms found the best solution in F1, and in this case, each algorithm was awarded 0.33 points. Hence, the sum of the scores achieved by all the algorithms will be equal to the sum of the functions.

As shown in Table 2, $\mathrm{RL}_{\mathrm{I-GWO}}$ found best solution in F1, F3, F9, and F10. Likewise, $\mathrm{RL}_{\mathrm{Ex-GWO}}$ found best solution in F9 and $\mathrm{RL}_{\mathrm{WOA}}$ found best solution in F5, F6, F8, and F9. In this table, F1 to F7 are unimodal and F8 to F10 are multimodal functions. It can be seen from the results of Table 2 that the number of best solutions found by the proposed algorithms was not high in unimodal functions compared to their metaheuristic versions. This is in the nature of unimodal functions. This type of function does not have more than one global optimum. Therefore, the exploration and exploitation phases do not necessitate many flexible optimized transitions and searching at different locals, as a result, hybrid algorithms have not been better than their metaheuristic counterparts. However, this does not mean that hybrid algorithms are necessarily worst at unimodal functions. For example, $\mathrm{RL}_{\mathrm{I-GWO}}$ found the best solution in F1 and F3, which are unimodal, and on the other hand, $\mathrm{RL}_{\mathrm{WOA}}$ is the best in F5 and F6.

In Table 3, comparison results of mean value and standard deviation for function F11–F20 are presented. In F11, all algorithms except GWO and $\mathrm{RL}_{\mathrm{GWO}}$ reach the optimum value. In F12–F15, it is seen that, $\mathrm{RL}_{\mathrm{WOA}}$ performs better compared to other algorithms. It demonstrates that $\mathrm{RL}_{\mathrm{WOA}}$ has a better balance in exploration and exploitation phases, besides its added advantage of being able to avoid getting stuck in the local trap. In F16–F19 all algorithms reached the optimum value; these benchmark functions are fixed dimension multimodal. In F20, $\mathrm{RL}_{\mathrm{WOA}}$ and GWO have better performance. Overall, $\mathrm{RL}_{\mathrm{WOA}}$ has a better performance for function F11–F20. As can be seen from the results, the proposed algorithms, especially $\mathrm{RL}_{\mathrm{WOA}}$, performed better than others in these functions consisting of 10 multimodal and fixed-dimensional multimodal. The main reason for this is presence of more than one local in multimodal-based functions. The proposed methods are likely to be better at such problems in line with the features mentioned earlier.

Table 2
Comparison results of simulations optimization algorithms for F1–F10.

F	GWO		RL$_{GWO}$		I-GWO		RL$_{I\text{-}GWO}$	
	Mean	***Std***	***Mean***	***Std***	***Mean***	***Std***	***Mean***	***Std***
F1	1.29E-27	1.86E-27	1.67E-33	5.27E-33	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F2	9.99E-17	1.01E-16	2.02E-21	3.90E-21	**0.00E+00**	0.00E+00	5.40E-215	0.00E+00
F3	9.42E-05	1.61E-04	3.86E-14	1.22E-13	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F4	8.34E-07	1.31E-06	2.59E-12	7.68E-12	**0.00E+00**	0.00E+00	4.52E-237	0.00E+00
F5	2.67E+01	8.78E-01	2.90E+01	6.86E-01	2.78E+01	1.50E-02	2.85E+01	4.05E-01
F6	1.20E+00	1.38E+00	8.52E-01	3.82E-01	6.75E+00	4.11E-01	4.86E+00	9.50E-01
F7	1.62E-03	3.09E-03	**4.11E-05**	0.000492	1.31E-03	4.57E-05	5.51E-05	6.27E-05
F8	-6.21E+03	7.89E+02	-2.09E+03	1.33E+03	-2.99E+03	5.44E+02	-4.09E+03	1.00E+03
F9	1.92E+00	2.93E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**
F10	1.09E-13	1.16E-14	**8.88E-16**	1.62E-14	**8.88E-16**	0.00E+00	**8.88E-16**	0.00E+00
	Ex-GWO		**RL$_{Ex\text{-}GWO}$**		**WOA**		**RL$_{WOA}$**	
	Mean	***Std***	***Mean***	***Std***	***Mean***	***Std***	***Mean***	***Std***
F1	**0.00E+00**	0.00E+00	5.08E-35	1.61E-34	4.29E-77	1.34E-76	2.09E-90	6.59E-90
F2	**0.00E+00**	0.00E+00	8.97E-24	2.80E-23	1.82E-49	5.57E-49	3.42E-59	8.53E-59
F3	**0.00E+00**	0.00E+00	1.25E-17	3.95E-17	3.87E+04	18249.18	1.30E-24	2.29E-24
F4	**0.00E+00**	0.00E+00	6.52E-16	2.06E-15	3.68E+01	27.56598	2.17E-20	3.98E-20
F5	2.78E+01	6.18E-01	2.83E+01	2.66E-01	2.81E+01	5.15E-01	**2.66E+01**	3.96E-01
F6	2.49E+00	4.57E-01	4.08E+00	7.15E-01	2.92E-01	1.16E-01	**1.54E-01**	1.28E-01
F7	1.31E-03	1.34E-04	1.26E-04	3.50E-04	2.79E-03	2.60E-03	4.80E-04	2.11E-04
F8	-2.99E+03	2.88E+02	-3.61E+03	2.91E+02	-1.01E+04	1.53E+03	**-1.07E+04**	1.69E+03
F9	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F10	**8.88E-16**	0.00E+00	2.31E-15	2.48E-15	4.80E-15	2.02E-15	2.66E-15	1.87E-15

*: bold is best results

Table 3
Comparison result of simulations optimization algorithms for F11–F20.

F	GWO		RL$_{GWO}$		I-GWO		RL$_{I\text{-}GWO}$	
	Mean	***Std***	***Mean***	***Std***	***Mean***	***Std***	***Mean***	***Std***
F11	5.71E-03	1.21E-02	1.60E-03	5.06E-03	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F12	3.72E-02	9.72E-03	1.07E-01	8.39E-02	1.17E+00	1.66E-01	5.97E-01	1.40E-01
F13	5.69E-01	1.94E-01	1.26E+00	3.39E-01	3.00E+00	1.55E-03	2.72E+00	1.08E-01
F14	4.90E+00	5.04E+00	7.05E+00	5.22E+00	6.28E+00	4.78E+00	6.67E+00	4.81E+00
F15	1.04E-02	1.03E-02	2.34E-03	6.34E-03	5.97E-03	6.32E-03	1.40E-03	2.66E-04
F16	**-1.0316**	3.18E-08	**-1.0316**	1.95E-08	**-1.0316**	3.18E-06	**-1.0316**	2.68E-05
F17	**0.398**	1.03E-06	**0.398**	4.02E-07	**0.398**	1.12E-04	**0.398**	1.72E-03
F18	**3.00**	4.52E-05	**3.00**	3.42E-05	**3.00**	6.92E-06	**3.00**	1.97E-03
F19	**-3.86**	3.28E-03	**-3.86**	2.13E-03	**-3.86**	3.06E-03	**-3.86**	1.31E-02
F20	**-3.26E+00**	8.48E-02	-3.29E+00	1.56E-01	-3.00E+00	1.59E-01	-3.01E+00	5.70E-01
	Ex-GWO		**RL$_{Ex\text{-}GWO}$**		**WOA**		**RL$_{WOA}$**	
	Mean	***Std***	***Mean***	***Std***	***Mean***	***Std***	***Mean***	***Std***
F11	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F12	9.13E-01	1.63E-01	4.89E-01	2.47E-01	3.42E-02	3.01E-02	**8.50E-03**	8.11E-03
F13	2.86E+00	3.58E-02	2.69E+00	1.06E-01	4.63E-01	1.44E-01	**2.11E-01**	1.20E-01
F14	1.79E+00	1.02E+00	1.99E+00	9.35E-01	4.72E+00	4.43E+00	**1.59E+00**	9.58E-01
F15	7.40E-04	2.88E-04	5.66E-04	2.55E-04	7.90E-04	3.54E-04	**5.57E-04**	1.25E-04
F16	**-1.0316**	3.47E-08	**-1.0316**	4.59E-05	**-1.0316**	7.76E-10	**-1.0316**	2.82E-10
F17	**0.398**	4.33E-06	**0.398**	1.18E-06	**0.398**	4.87E-06	**0.398**	9.20E-05
F18	**3.00**	2.83E-05	**3.00**	1.81E-04	**3.00**	1.73E-04	**3.00**	9.09E-04
F19	**-3.86**	3.10E-03	**-3.86**	3.50E-03	**-3.86**	2.62E-03	**-3.86**	1.71E-02
F20	-3.17E+00	1.23E-01	-3.05E+00	2.69E-01	-3.30E+00	1.54E-01	**-3.26E+00**	1.00E-01

*: bold is best results

The results, reported in Table 3, consists of all types of functions (F21 to F30). F24, F26, F28, and F30 are unimodal and the other functions are multimodal in fixed or unfixed dimensions. RL$_{I-GWO}$ found the best solution in F21, 23, 24, 27, 28, 30. Whereas, RL$_{Ex-GWO}$ is best in F25, 27, 30 and RL$_{WOA}$ is better than others in F22, 27, 30. In F21, RL$_{I-GWO}$ and Ex-GWO give a better solution. RL$_{WOA}$ obtains the best value for F22. In F23 I-GWO and RL$_{I-GWO}$ earned better mean value. In F24, RL$_{I-GWO}$, and in F25, RL$_{Ex-GWO}$ and in F26, GWO algorithms have better performance. In F27 and F30, all algorithms reach the optima value. In F28, I-GWO, RL$_{I-GWO}$, and Ex-GWO have obtained a better solution. In F29, RL$_{GWO}$ earned better value in comparison to others.

Fig. 9 is prepared as a percentage of the performance of each algorithm in finding the best solutions using the results that have been presented in the three tables given above. Overall, as shown in Fig. 9(a), RL$_{WOA}$ has earned better values, and I-GWO, RL$_{I-GWO}$, Ex-GWO, GWO, RL$_{Ex-GWO}$, RL$_{GWO}$, and WOA are placed in other ranks, respectively. According to these results, presented

Table 4
Comparison result of simulations metaheuristic algorithms for F21–F30.

F	GWO		RL$_{GWO}$		I-GWO		RL$_{I-GWO}$	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*
F21	0.000706	5.44E-04	8.51E-06	2.69E-05	4.79E-237	0.00E+00	**0.00E+00**	0.00E+00
F22	1.22E-07	1.47E-07	7.62E-02	3.51E-07	1.72E-05	1.98E-05	1.05E-04	3.21E-01
F23	1.61E-29	1.39E-29	3.14E-35	2.63E-34	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F24	2.59E-103	6.07E-105	9.45E-106	5.36E-107	9.64E-298	0.00E+00	**0.00E+00**	0.00E+00
F25	-1.01E+01	2.04E+00	-8.93E+00	9.44E-01	-5.93E+00	6.86E-01	-6.53E+00	5.90E-01
F26	**3.08E-07**	5.26E-07	5.49E-07	4.47E-07	6.13E-05	6.57E+00	4.32E-04	6.94E-04
F27	**-1.00E+00**	5.78E-07	**-1.00E+00**	2.21E-07	**-1.00E+00**	1.33E-04	**-1.00E+00**	3.16E-01
F28	9.57E-29	1.31E-28	1.56E-33	4.79E-33	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F29	2.58E-016	1.11E-016	**1.29E-26**	2.40E-26	4.13E-11	2.33E-11	6.88E-22	3.59E-22
F30	**-0.00379**	3.35E-10	**-0.00379**	2.33E-10	**-0.00379**	9.07E-05	**-0.00379**	2.25E-08
	Ex-GWO		**RL$_{Ex-GWO}$**		**WOA**		**RL$_{WOA}$**	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*
F21	**0.00E+00**	0.00E+00	3.29E-20	1.04E-19	1.07E-44	3.37E-44	1.60E-59	5.04E-59
F22	3.57E-07	5.85E-07	7.77E-07	5.82E-07	7.62E-02	2.41E-01	**1.87E-09**	5.24E-09
F23	2.36E-28	0.00E+00	1.48E-41	1.98E-32	5.27E-75	1.84E-76	7.35E-92	7.97E-91
F24	5.94E-101	1.01E-95	1.21E-121	6.39E-111	1.47E-187	0.00E+00	4.20E-197	0.00E+00
F25	-6.49E+00	6.84E-01	**-9.29E+00**	9.36E-01	-1.12E+01	2.95E+00	-1.21E+01	2.18E+00
F26	1.62E-06	3.36E-02	1.61E-06	2.56E-06	1.02E-03	2.80E-03	1.66E-03	1.13E-03
F27	**-1.00E+00**	2.62E-06	**-1.00E+00**	2.76E-06	**-1.00E+00**	3.35E-06	**-1.00E+00**	1.92E-06
F28	**0.00E+00**	0.00E+00	6.25E-38	1.98E-37	7.56E-74	1.52E-73	6.56E-93	1.47E-92
F29	2.87E-16	7.62E-12	4.82E-26	4.45E-26	1.67E-05	3.60E-15	8.17E-25	3.66E-25
F30	**-0.00379**	3.09E-10	**-0.00379**	3.89E-10	**-0.00379**	2.25E-09	**-0.00379**	2.36E-08

*: bold is best results

Table 5
Rank summary of statistical assessment results for F1–F30.

F	GWO	RL$_{GWO}$	I-GWO	RL$_{I-GWO}$	Ex-GWO	RL$_{Ex-GWO}$	WOA	RL$_{WOA}$
F1	8	7	1	1	1	6	5	4
F2	8	7	1	3	1	6	5	4
F3	7	6	1	1	1	5	8	4
F4	7	6	1	3	1	5	8	4
F5	2	5	8	7	3	6	4	1
F6	4	3	8	7	5	6	2	1
F7	7	4	1	2	6	3	8	5
F8	3	4	8	5	7	6	2	1
F9	8	7	1	1	1	1	1	1
F10	8	7	1	1	1	4	6	5
F11	8	7	1	1	1	1	1	1
F12	3	4	8	6	7	5	2	1
F13	3	4	8	6	7	5	2	1
F14	3	4	7	8	5	6	2	1
F15	5	7	8	6	3	2	4	1
F16	1	1	1	1	1	1	1	1
F17	1	1	1	1	1	1	1	1
F18	1	1	1	1	1	1	1	1
F19	1	1	1	1	1	1	1	1
F20	1	4	8	7	5	6	3	1
F21	8	7	3	1	1	6	5	4
F22	2	8	5	7	6	3	4	1
F23	7	6	1	1	8	5	4	3
F24	7	6	2	1	8	5	4	3
F25	2	3	8	6	7	1	4	5
F26	1	2	5	6	4	3	7	8
F27	1	1	1	1	1	1	1	1
F28	8	7	1	1	1	6	5	4
F29	2	1	8	7	3	4	5	6
F30	1	1	1	1	1	1	1	1

in Fig. 9(a), the success rate performance of each algorithm was calculated as a percentage based on finding the best solution for each benchmark function. Based on the results obtained, it can be noted that, RL$_{WOA}$ is generally better than other algorithms. RL$_{WOA}$ algorithm obtained the best solution in 17 out of 30 Benchmark functions compared to the other seven algorithms giving a 56.67% success rate. It should be noted that, in the overall comparison, RL$_{WOA}$ has a success rate of 32% amongst the algorithms compared. As mentioned before, when more than one best solution is found for a benchmark function, the score is equally divided among the algorithms. Results show that RL$_{WOA}$ is 49% better than the I-GWO algorithm, which is the second-ranked. At the same time, the success rate of RL$_{WOA}$ is 54% better than RL$_{I-GWO}$, 59% better than Ex-GWO algorithm, 76% better than GWO, 78% better than RL$_{Ex-GWO}$, 84% better than RL$_{GWO}$, and 88% better than WOA.

At the same time, the performance and success rate of each algorithm is compared in pairs in Fig. 9(b). This figure shows the performance of the metaheuristic versions of the algorithms with RL-based hybrid versions, in comparison. The full mark is awarded to the best of the two. According to these results, the hybrid versions were able to perform better in comparison to their non-hybrid counterparts, except for Ex-GWO algorithm. The main reason that Ex-GWO outperforms its hybrid version is that almost all wolves in the pack have an important role in each other's position update. Therefore, the wolves in the pack minimize the escape paths of the hunt (prey), and hence, the hunts can be caught faster. Based on this, even if a different mechanism is applied to this structure, it may be less likely to find a better answer than the original version. The results of this study also illustrate this analysis. The most important difference between the other three pairs seems to be between WOA and RL$_{WOA}$. The RL$_{WOA}$ also performed best for the benchmark functions in this study. The main reason for its success is that the exploration (searching of prey) and exploitation (attacking of prey) phases in WOA algorithm are less interdependent than the other three metaheuristic algorithms. On the other hand, in the WOA algorithm uses a threshold p, to exhibit different behaviors. In this way, the new RL-based hybrid mechanism (RL$_{WOA}$) has made the original algorithm more compatible and has even achieved better performance compared to all other algorithms.

Looking at particular result instances helps in understanding some values, in items (a) and (b) of Fig. 9, correctly. In Fig. 9(a), the success of GWO algorithm is 7.5%, while the performance percentage of RL$_{GWO}$ is 5.8%. However, opposite result is seen in Fig. 9(b) where RL$_{GWO}$ shows better performance compared

Table 6
Pair-wise statistical comparison using Wilcoxon signed-rank test (a = 0.05).

F	RL_{GWO} vs GWO	RL_{I-GWO}vs I-GWO	RL_{Ex-GWO}vs Ex-GWO	RL_{WOA} vs WOA
F1	+	∼	–	+
F2	+	–	–	+
F3	+	∼	–	+
F4	+	–	–	+
F5	–	+	–	+
F6	+	+	–	+
F7	+	–	+	+
F8	–	+	+	+
F9	+	∼	∼	∼
F10	+	∼	–	+
F11	+	∼	∼	∼
F12	–	+	+	+
F13	–	+	+	+
F14	–	–	–	+
F15	+	+	+	+
F16	∼	∼	∼	∼
F17	∼	∼	∼	∼
F18	∼	∼	∼	∼
F19	∼	∼	∼	∼
F20	+	–	+	+
F21	+	+	–	+
F22	–	–	–	+
F23	+	∼	+	+
F24	+	+	+	+
F25	–	+	+	–
F26	–	–	+	+
F27	∼	∼	∼	∼
F28	+	∼	–	+
F29	+	+	–	–
F30	∼	∼	∼	∼
Total of + signs	**16**	**10**	**10**	**20**

to GWO. This result is obtained using a dual comparison of the performance of GWO and RL_{GWO} in each benchmark function in Tables 2, 3, and 4, over mean values. These values show that RL_{GWO} was better than GWO on 16, worse in 8 and equal in 6 benchmark functions. Results further show that RL_{GWO} algorithm achieved a 63.3% share in success. Similar calculations and comparisons were applied to the other three pairs of algorithms. This is presented in Table 6 from a different perspective.

In Table 5, the rank summary of each algorithm is presented. In ranking summary, the mean fitness value is considered. Overall, the proposed method (RL_{WOA}) obtains a high-ranking value. RL_{WOA} algorithm has shown more successful performance than other algorithms in balancing between two exploration and exploitation phases. As in the Q-learning method, exploitation action allows an agent to search in other local spaces. Furthermore, applying Q-learning with a metaheuristic algorithm allows the metaheuristic search agent to search in other search spaces. In this way, the exploration and exploitation balance are controlled by Q-values in the Q-Table.

The results obtained for each algorithm are examined by a well-known non-parametric statistical test, namely the Wilcoxon rank-sum test, that generates three signs '-', '∼', and '+'. The test was conducted at a 0.05 significance level. Each hybrid (combination of RL and metaheuristic) algorithm is tested against its original metaheuristic algorithm, specifically, RL_{GWO} vs. GWO, RL_{I-GWO} vs. I-GWO, RL_{Ex-GWO} vs. Ex-GWO, and RL_{WOA} vs. WOA. Again, results are obtained from Tables 2, 3, and 4 with 10 independent runs. A value of '–' shows the obtained result of the RL-based algorithm is worse than the metaheuristic version of the relevant algorithm. '∼' indicates the obtained value of both algorithms is same, and '+' implies that the obtained value of the RL-based algorithm is better than the metaheuristic version. The RL-based algorithms are considered for comparison as the basic algorithm. For instance, looking at results in Table 1 for function 1 presents '+' for RL_{WOA} vs. WOA. It means that the value obtained for RL_{WOA} for function 1 is better than the value obtained for WOA. An RL-based algorithm with a larger number of '+' signs indicates that it is the best hybrid algorithm. The last row of Table 6 presents the total amount of '+' signs for each RL-based algorithm. This shows the degree to which it is better than its metaheuristic version. Based on this result, it shows that RL_{WOA} is better than others. The total amount of '+' signs gained for each function, guides to reach the performance of the respective hybrid algorithm (RL_{GWO}, RL_{I-GWO}, RL_{Ex-GWO}, and RL_{WOA}). In the same context, the total amount of '-' signs can show the analysis of each metaheuristic algorithm (GWO, I-GWO, Ex-GWO, and WOA). The score for the elements equal to '∼' is evenly distributed between the two algorithms. The results of this pairwise comparison have also been presented in Fig. 9(b).

4.3. Convergence rate analysis

This section analyzes the convergence speed of each algorithm used over some functions. The convergence curve also shows the balance between exploration and exploitation phases. Convergence curve analysis proves that metaheuristic algorithms avoid trapping in local optima. Although the proposed RL-based algorithms have the same goals, they show more successful performances in this regard. In the exploration and exploitation phases, the search agents visit new points in the exploration phase. After this, in the exploitation phase, these points are refined by the search agents to find the better solution. The reinforcement agent improves the search agents to use the points in search spaces efficiently. Therefore, the proposed algorithms are able to search better in the global space. Each of the eight algorithms' convergence curve on select benchmark functions is shown in Fig. 10. In the first steps of metaheuristics-based methods, the sudden movement changes of agents can help these algorithms overcome exploration. Then, these changes should be reduced in the other phase at the end of optimization. As aforementioned, the rate of these changes naturally decreases as

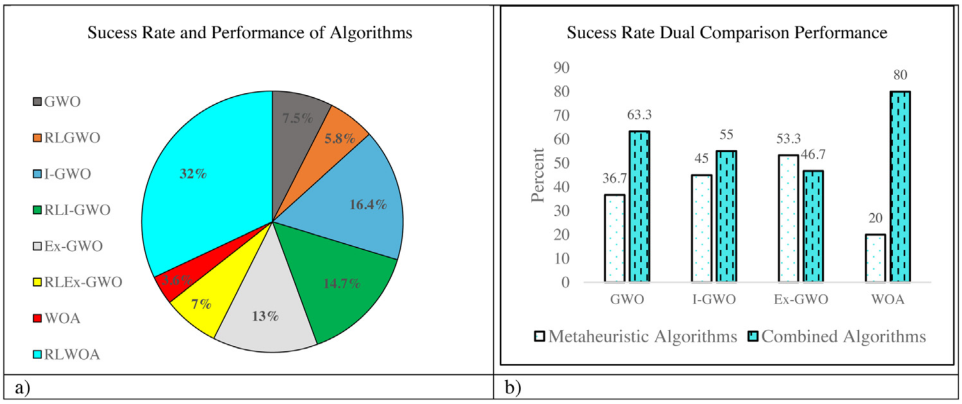


Fig. 9. (a) success rate of each algorithm with the best value in all benchmark functions, (b) success rate in dual comparison of algorithms, metaheuristic algorithm vs. combined reinforcement learning-based metaheuristic algorithm.

Table 7
D–H Kinematic Parameters of PUMA 560.

No	a_i (meters)	d_i (meters)	Range(°)	
1	0	0	−160	+160
2	0	0	−245	+45
3	0.4318	0.1491	−45	+225
4	0.0203	0.1331	−110	+170
5	0	0	−100	+100
6	0	0	−266	+266

iterations progress. According to [19,65], this behavior can guarantee that a metaheuristic algorithm eventually convergences to a point in search space. In the metaheuristic algorithms, with the control parameters available at exploration and exploitation phases, they try to guide search agents in finding better global solutions by fast convergence. In the proposed hybrid algorithms, a mechanism based on Q-Learning tries to guide search agents in finding new parts of the discoverable global search space. Based on the convergence behavior analysis, it can be observed that the search agents of $\mathrm{RL_{WOA}}$ tend to extensively search promising regions of the search spaces and exploit the best ones.

Fig. 11 presents the boxplot of each algorithm on F_1, F_5, F_6, F_{15}, F_{16}, and F_{29}. Boxplots are a standard method for displaying data distribution based on statistical indicators such as minimum, first quartile (Q1), median, third quartile (Q3), and maximum. This diagram also provides information regarding the existence of outlier data. Another purpose of this chart is to highlight the symmetry in the data. Each box may have lines extending from boxes, these lines indicate the upper and lower quartiles, sometimes these lines appear as points. In some boxplot figures, there is space between the different parts of the box. This space indicates the spread and skewness in the data. The boxplots of the proposed algorithms are shown in Fig. 11, for 10 independent runs with a population size of 30, using 500 iteration per run. Using the boxplot analysis, proposed algorithms was plotted with values from 10 different runs. In each metaheuristic algorithm based on the RL method, the distribution of the obtained value is near to each other and more balanced.

5. Proposed algorithms for inverse kinematics of robot arms

In this section, the proposed algorithms are applied to inverse kinematics of robot arms of the optimization problem. The performance evaluation of RL-based metaheuristic algorithms is calculated with inverse kinematics of a robot arms. This problem is also an important real-world engineering problem. The 6-DOF PUMA 560 robot arm has six joints. In the proposed problem, the first three joints locate the end-effector, the remaining joints provide the orientation of the end-effector [66], shown in Fig. 12. The Denavit–Hartenberg (D–H) parameters are presented in Table 7. In this problem, the objective function is calculated using Eq. (24). This equation describes the neighboring frames resulting from the multiplication of the transformations.

$$^{0}T_6 = {}^{0}T_1 \ {}^{1}T_2 \ {}^{2}T_3 \ {}^{3}T_4 \ {}^{4}T_5 \ {}^{5}T_6 \tag{24}$$

In the Eq. (24), $^{i}T_{i+1}$ indicates the transfer matrix of link i. Cartesian coordinate for each joint angle is defined by $^{0}T_6$ matrix. As discussed earlier, this equation is used as a cost function. The Euclidean distance between obtained and target points determines the cost of the Cartesian coordinate. Eq. (25) shows the matrix that results from running the $^{0}T_6$ equation. In other words, the result of Eq. (24) is a 4×4 transformation matrix given in Eq. (25).

$$^{0}T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

Each element of the $^{0}T_6$ matrix is calculated employing Eq. (26). The Cartesian coordinate (x_s, y_s, z_s) and the target coordinate (x, y, z) have been used to find the optimum angle value for each joint given from the θ value. The c_i indicates $cos(\theta_i)$ and s_i indicates $sin(\theta_i)$. Furthermore, Eq. (25) converts joint angles θ to the Cartesian coordinate (x', y', z'). The cost of each angle is calculated from the Euclidean distance between the initial and current coordinates (Eq. (27)).

$$\begin{aligned}
r_{11} &= c_1 \left[c_{23} \left(c_4 c_5 c_6 - s_4 s_6 \right) - s_{23} s_5 c_6 \right] + s_1 (s_4 c_5 c_6 + c_4 c_6), \\
r_{21} &= s_1 \left[c_{23} \left(c_4 c_5 c_6 - s_4 s_6 \right) - s_{23} s_5 c_6 \right] - c_1 \left(s_4 c_5 c_6 + c_4 c_6 \right), \\
r_{31} &= -s_{23} \left(c_4 c_5 c_6 - s_4 s_6 \right) - c_{23} s_5 c_6, \\
r_{12} &= c_1 \left[c_{23} \left(-c_4 c_5 c_6 - s_4 s_6 \right) + s_{23} s_5 c_6 \right] + s_1 (s_4 c_5 c_6 - c_4 c_6), \\
r_{22} &= s_1 \left[c_{23} \left(-c_4 c_5 c_6 - s_4 s_6 \right) + s_{23} s_5 c_6 \right] - c_1 (s_4 c_5 c_6 - c_4 c_6), \\
r_{32} &= -s_{23} \left(c_4 c_5 c_6 - s_4 s_6 \right) + c_{23} s_5 c_6, \\
r_{13} &= -c_1 \left(c_{23} c_4 s_5 - s_{23} c_5 \right) - s_1 s_4 s_5, \\
r_{23} &= -s_1 \left(c_{23} c_4 s_5 + s_{23} c_5 \right) + c_1 s_4 s_5, \\
r_{33} &= s_{23} c_4 s_5 - c_{23} c_5, \\
p_x &= c_1 \left[a_2 c_2 + a_3 c_{23} - d_4 s_{23} \right] - d_3 s_1, \\
p_y &= s_1 \left[a_2 c_2 + a_3 c_{23} - d_4 s_{23} \right] + d_3 c_1, \\
p_z &= -a_3 s_{23} - a_2 s_2 - d_4 c_{23}
\end{aligned} \tag{26}$$

$$d = \sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2} \tag{27}$$

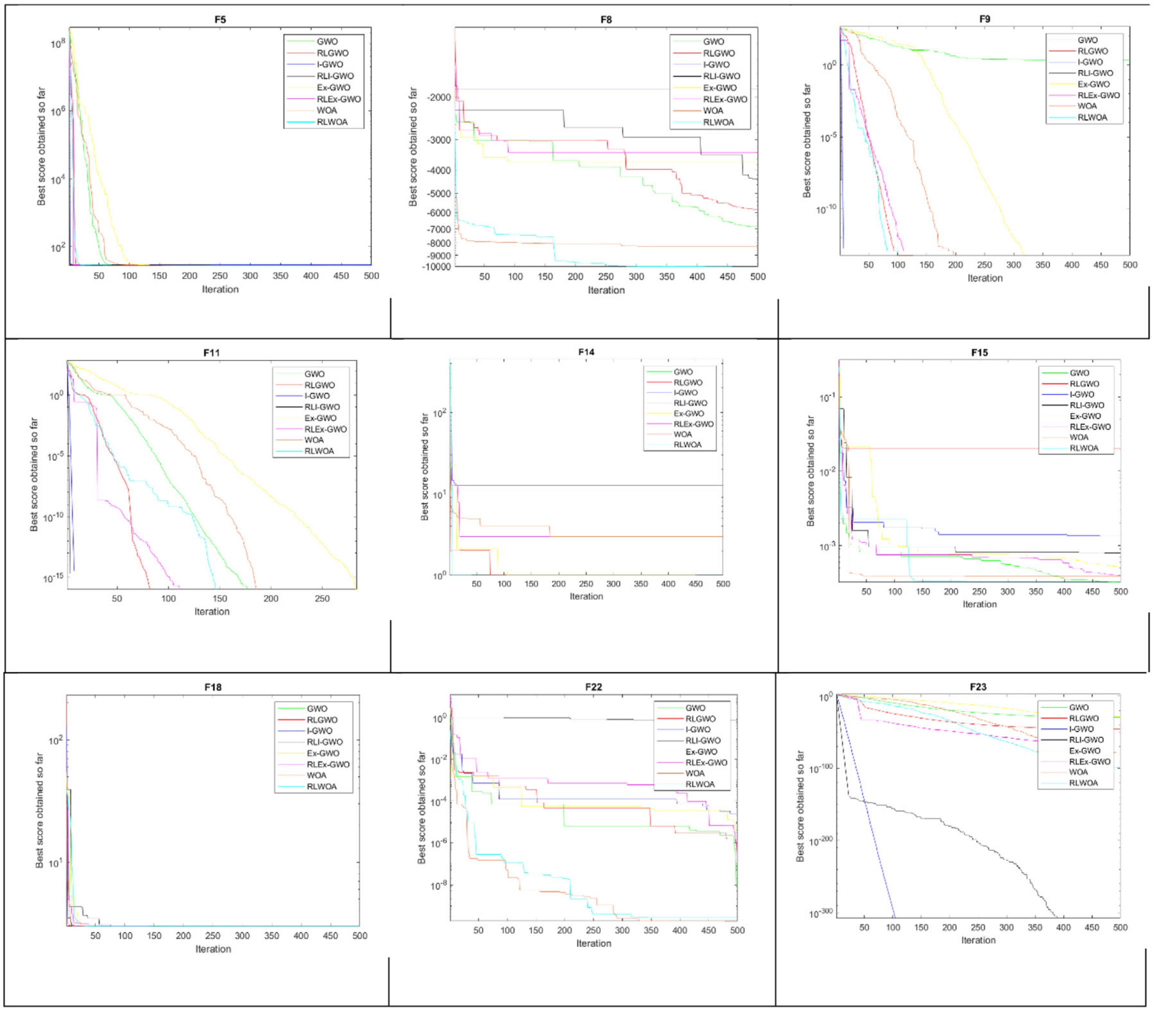


Fig. 10. The convergence curve of each algorithm in 500 iteration numbers.

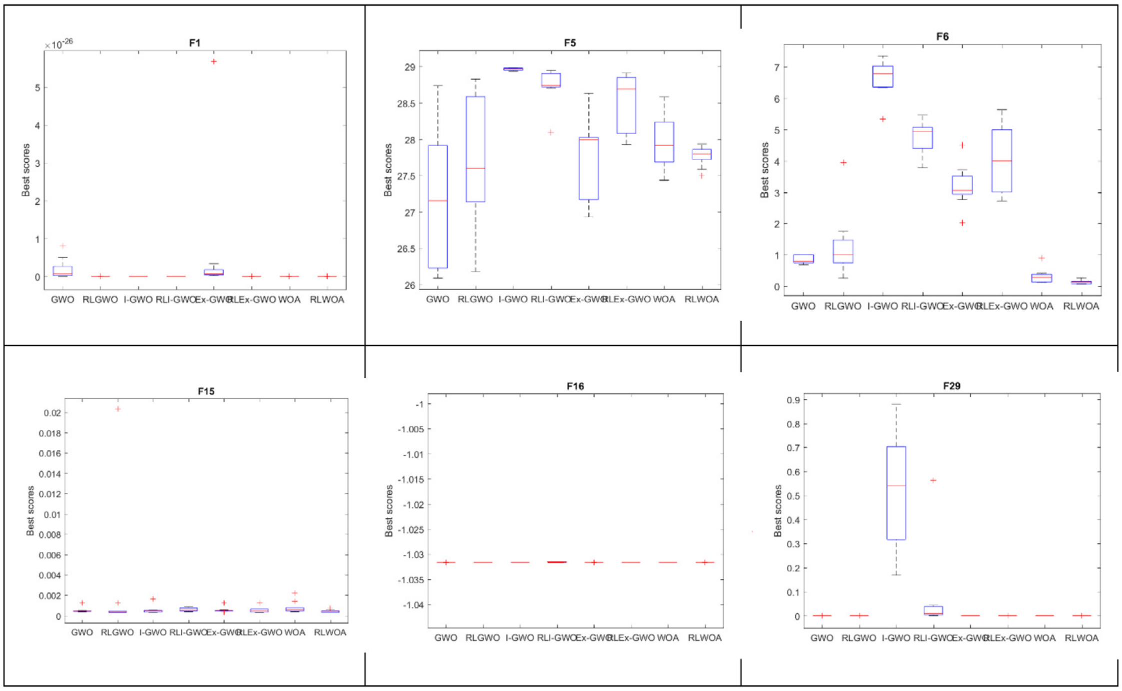


Fig. 11. The boxplot of each algorithm.

Table 8
The results of fitness function and joint variables by each algorithm.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Function value
GWO	19.842	−37.193	−1.787	78.657	52.772	55.905	8.14E−03
RL_{GWO}	−146.863	−111.790	3.8531	−104.408	64.927	10.855	1.94E−03
I-GWO	−14.33	−34.46	−24.723	51.224	77.602	3.925	2.24E−03
RL_{I-GWO}	−32.887	−120.018	172.920	42.731	88.0947	−218.780	5.14E−03
Ex-GWO	−19.901	−27.601	213.540	118.195	100	266	5.71E−03
RL_{Ex-GWO}	9.012	0.324	−40.983	−96.219	−48.565	233.375	4.61E−03
WOA	−49.177	−90.802	59.189	88.642	45.293	50.620	1.13E−03
RL_{WOA}	159.602	4.505	194.685	70.542	−52.521	−106.366	**9.70E−04**

Table 9
The obtained mean and standard deviation value for each algorithm.

Pop.	***Iter.***	***GWO***		***RL_{GWO}***		***I-GWO***		***RL_{I-GWO}***	
		Mean	***STD***	***Mean***	***STD***	***Mean***	***STD***	***Mean***	***STD***
50	500	4.09E-03	2.20E-03	1.70E-03	4.40E-03	4.20E-03	5.80E-03	4.90E-03	4.90E-03
	1000	9.68E-03	2.55E-03	2.04E-03	4.63E-03	2.54E-03	4.95E-03	5.39E-03	5.60E-03
	2000	7.57E-03	3.93E-03	4.60E-04	2.20E-03	2.90E-03	5.20E-03	5.00E-03	5.60E-03
100	500	1.81E-04	3.93E-04	4.11E-04	2.20E-03	7.56E-04	2.80E-03	7.00E-03	5.20E-03
	1000	6.23E-04	6.54E-04	8.17E-04	3.11E-03	8.72E-04	3.05E-03	5.67E-04	5.66E-03
	2000	2.19E-04	5.70E-04	4.16E-04	2.20E-03	1.20E-03	3.70E-03	3.60E-03	4.90E-03
200	500	4.36E-04	1.39E-04	4.09E-04	2.20E-03	4.69E-04	2.20E-03	5.10E-03	5.40E-03
	1000	3.14E-04	9.93E-04	3.18E-04	9.91E-04	3.92E-05	4.52E-05	3.71E-05	5.05E-03
	2000	1.64E-04	8.93E-04	1.20E-03	3.60E-03	1.66E-05	3.55E-05	**1.60E-05**	**3.70E-05**
300	500	1.78E-05	3.04E-05	**4.01E-07**	**2.44E-06**	7.16E-04	2.79E-04	6.08E-05	4.28E-03
	1000	4.90E-07	2.61E-06	**2.28E-07**	**1.18E-06**	5.72E-05	2.70E-04	4.77E-05	5.62E-04
	2000	4.81E-08	3.93E-08	4.11E-08	2.20E-07	7.56E-06	2.80E-05	6.12E-07	4.31E-06

Pop.	***Iter.***	***Ex-GWO***		***RL_{Ex-GWO}***		***WOA***		***RL_{WOA}***	
		Mean	***STD***	***Mean***	***STD***	***Mean***	***STD***	***Mean***	***STD***
50	500	8.21E-03	1.20E-03	2.31E-03	1.70E-03	4.60E-03	5.40E-03	**1.01E-03**	**2.20E-03**
	1000	5.73E-03	6.29E-03	4,75E-03	8,34E-03	1,09E-03	2,14E-03	**1.03E-03**	**3.32E-03**
	2000	2.04E-04	4.01E-04	3.80E-04	4.76E-04	1.50E-03	2.70E-03	**1.40E-04**	**2.10E-03**
100	500	2.91E-04	3.36E-04	9.05E-04	7.19E-04	6.04E-04	1.20E-03	**1.50E-03**	**2.20E-03**
	1000	4.19E-04	4.96E-04	4.48E-04	5.20E-04	2.30E-04	6.34E-04	**1.06E-04**	**1.69E-03**
	2000	1.46E-04	1.92E-04	2.45E-04	2.46E-04	7.87E-05	2.86E-04	**1.18E-04**	**2.87E-04**
200	500	3.65E-04	3.24E-04	**3.29E-04**	**2.60E-04**	3.96E-04	1.00E-03	1.60E-03	3.50E-03
	1000	3.93E-05	2.59E-04	**3.40E-05**	**4.19E-04**	1.39E-04	1.65E-05	1.13E-04	1.03E-03
	2000	1.28E-04	1.31E-04	1.72E-05	1.38E-04	3.77E-05	1.15E-04	1.84E-05	1.61E-04
300	500	1.01E-04	3.93E-04	4.04E-05	2.20E-04	6.22E-04	2.14E-04	2.00E-05	2.22E-03
	1000	1.92E-05	1.55E-04	1.95E-05	1.97E-04	3.54E-05	8.49E-05	2.34E-05	8.52E-04
	2000	3.55E-06	3.93E-06	**4.00E-08**	**5.20E-08**	8.12E-06	5.44E-05	4.12E-07	3.20E-07

*: bold is best results

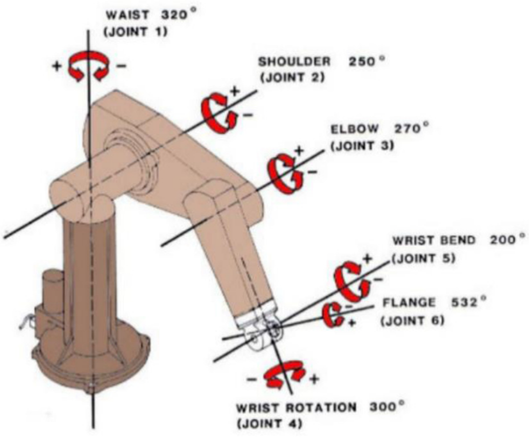


Fig. 12. PUMA 560 Robot Arm [66].

Table 8 reports the comparative results of optimization algorithms for the evaluation of inverse kinematic fitness function and joint variables. The conducted experiment does not follow any special set of associated parameters of all algorithms. With the angles obtained, a function value is calculated for each algorithm. In this scenario, the algorithm with the least value shows the best performance. It is observed from Table 8 that RL_{WOA} performs well, compared to other algorithms on the basis of fitness evaluations. These values have been obtained by 50 agent members in 1000 iterations for a single run.

The simulation results of the inverse kinematics of the PUMA 560 robot arm are presented in Table 9. The simulation is run 30 times independently with different search agents and various iterations to evaluate the performance of each algorithm used in this paper. To obtain trade-off parameters, experiments have been conducted with different population sizes and iteration counts. Population numbers are considered as 50, 100, 200, and 300. Each population was run in three different iterations (500, 1000, and 2000). According to the results, RL-based methods performed better. In addition, the results obtained of hypothetical cases show that RL_{WOA} is better than the other seven algorithms used. $\mathrm{RL_{WOA}}$ ranked first in solving the kinematics problem, finding the best solution in 6 out of 12 cases. Also, in some scenarios with a large number of populations and iterations, $\mathrm{RL_{Ex-GWO}}$ appears to perform relatively better than others.

6. Conclusion and future work

In this paper, three hybrid algorithms, which are the combination of metaheuristic and reinforcement learning-based algorithms, were presented. These algorithms are named $\mathrm{RL_{I-GWO}}$, $\mathrm{RL_{Ex-GWO}}$, and $\mathrm{RL_{WOA}}$. Utilizing the advantages of metaheuristics and RL algorithms, at the same time trying to eliminate (or minimize) their shortcomings, these algorithms aim at finding better solutions with appropriate decisions between the exploration and exploitation phases. The proposed algorithms can also provide appropriate transitions between these two phases, each exhibiting a more balanced behavior, and as a result, they exhibit a good convergence rate. In the proposed hybrid algorithms, a mechanism based on Q-Learning tries to guide search agents in finding new parts of the global search space to be explored. Q-Learning method was applied in I-GWO, Ex-GWO, and WOA algorithms to reach optimal solutions employing a Q-Table In the Q-Table, each action and task have a reward or a punishment, where the current and previous information is compared. It is used to guide the search agent in exploring the new global environment from its exploitation phase(s). This way, the search agents of proposed algorithms have a higher chance to move in the search space compared to their metaheuristic versions. With exploration, they discover a new global search space and exploit it to find the new optimal value. Therefore, the proposed algorithms have advantages compared to the metaheuristic algorithms, which try to explore possible search space better. Furthermore, the performances of the proposed algorithms were analyzed, commented on, and compared to the other five algorithms. The results obtained show that $\mathrm{RL_{WOA}}$ algorithm is more efficient in finding optimum solutions in 30 benchmark test functions used the and Kinematic problem. The algorithms proposed in this study are new in approach and can be used in many problems in different fields. In particular, they may be suitable for solving multi-objective problems and for applying in concurrent or parallel systems. Therefore, in future, the proposed algorithms can be used to solve various real-world problems such as mechanical engineering, optimization in electronic circuit designs, optimization in feature selection, optimization in IoT, and wireless sensor network routing algorithms.

CRediT authorship contribution statement

Amir Seyyedabbasi: Conceptualization, Methodology, Software, Validation, Formal analysis. **Royal Aliyev:** Software, Validation, Investigation. **Farzad Kiani:** Conceptualization, Supervision, Project administration, Methodology. **Murat Ugur Gulle:** Investigation, Writing - original draft. **Hasan Basyildiz:** Investigation, Writing - original draft. **Mohammed Ahmed Shah:** Investigation, Formal Analysis, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2021.107044.

References

[1] O. Olorunda, A.P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in: Evolutionary Computation, CEC 2008, (IEEE World Congress on Computational Intelligence). IEEE Congress on, 2008, pp. 1128-1134.

[2] T. Eftimov, P. Korosec, Understanding exploration and exploitation powers of meta-heuristic stochastic optimization algorithms through statistical analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 21–22.

[3] E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular genetic algorithms, IEEE Trans. Evol. Comput. 9 (2005) 126–142, 2005.

[4] T. Rahkar Farshi, Battle royale optimization algorithm, Neural Comput. Appl. (2020) http://dx.doi.org/10.1007/s00521-020-05004-4.

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

[6] J.H. Holl, Genetic algorithms, Sci. Am. 267 (1992) 66–72.

[7] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE international conference on neural networks, 1995, pp. 1942-1948.

[8] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.

[9] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.

[10] V. Hayyolalam, A.A.P. Kazem, Black widow optimization algorithm: A novel metaheuristic approach for solving engineering optimization problems, Eng. Appl. Artif. Intell. 87 (2020) 103249.

[11] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.

[12] A. Kaura, L.K. Awasthia, A.L. Sangal, G. Dhiman, Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization, Eng. Appl. Artif. Intell. 90 (2020) 103541, 1–29.

[13] M. Yazdani, F. Jolai, Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm, J. Comput. Des. Eng. 3 (1) (2016) 24–36.

[14] Q. Askari, I. Younas, M. Saeed, Political Optimizer: A novel socio-inspired meta-heuristic for global optimization, Knowl.-Based Syst. 195 (2020) 105709, 1–25.

[15] J. Wu, et al., An improved firefly algorithm for global continuous optimization problems, Expert Syst. Appl. 149 (2020) 113340, 1–12.

[16] X. Fung, Xu H., Y. Wang, H. Yu, The social team building optimization algorithm, Soft Comput. 23 (2019) 6533–6554.

[17] M.H. Sulaiman, Z. Mustaffa, M.M. Saari, H. Daniyal, Barnacles Mating Optimizer: A new bio-inspired algorithm for solving engineering optimization problems, Eng. Appl. Artif. Intell. 87 (2020) 103330.

[18] S.H.S. Moosavi, V.K. Bardsiri, Poor and rich optimization algorithm: A new human-based and multi populations algorithm, Eng. Appl. Artif. Intell. 86 (2019) 165–181.

[19] A. Seyyedabbasi, F. Kiani, I-GWO and Ex-GWO: improved algorithms of the Grey Wolf Optimizer to solve global optimization problems, Eng. Comput. 37 (2021) 519–532.

[20] J. Žerovnik, Heuristics for NP-hard optimization problems - simpler is better!? Logist. Sustain. Transp. 6 (1) (2015) 1–10.

[21] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.

[22] David H. Wolpert, G.Macready. William, et al., No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

[23] X. Cai, H. Qiu, L. Gao, C. Jiang, X. Shao, An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems, Knowl.-Based Syst. 184 (2019) 104901, 1–17.

[24] A. Seyyedabbasi, F. Kiani, MAP-ACO: An efficient protocol for multi-agent pathfinding in real-time WSN and decentralized IoT systems, Microprocess. Microsyst. 79 (2020) 1–9.

[25] P. Mesejo, O. Ibáñez, O. Cordón, S. Cagnoni, A survey on image segmentation using metaheuristic-based deformable models: state of the art and critical analysis, Appl. Soft Comput. 44 (2016) 1–29.

[26] M.A. Contreras-Cruz, V. Ayala-Ramirez, U.H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Appl. Soft Comput. 30 (2015) 319–328.

[27] F. Güneş, M.A. Belen, P. Mahouti, Competitive evolutionary algorithms for building performance database of a microwave transistor, Int. J. Circuit Theory Appl. 46 (2) (2018) 244–258.

[28] Y. Zhang, Z. Jin, Y. Chen, Hybridizing grey wolf optimization with neural network algorithm for global numerical optimization problems, Neural Comput. Appl. 32 (2020) 10451–10470.

[29] F. Soleymani, E. Paquet, Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—Deepbreath, Expert Syst. Appl. 156 (2020) 113456, 1–16.

[30] H. Akbari, A. Kazerooni, KASRA: A Kriging-based Adaptive Space Reduction Algorithm for global optimization of computationally expensive black-box constrained problems, Appl. Soft Comput. 90 (2020) 106154, 1–22.

[31] E. Talbi, Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics, 2020, 2020, hal-02745295.

[32] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, second ed., The MIT Press, Cambridge, Massachusetts, London, England, 2018, pp. 119–140.

[33] S. Richard, G. Andrew, Introduction To Reinforcement Learning, Vol. 135, second ed., A Bradford Book, MIT Press, 2015.

[34] H. Zhang, T. Yu, Taxonomy of reinforcement learning algorithms, in: H. Dong, Z. Ding, S. Zhang (Eds.), Deep Reinforcement Learning, Springer, Singapore, 2020, http://dx.doi.org/10.1007/978-981-15-4095-0_3.

[35] M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, Swarm Evol. Comput. 44 (2019) 228–246.

[36] G. Lingam, R.R. Rout, D.V.L.N. Somayajulu, Adaptive deep Q-learning model for detecting social bots and influential users in online social networks, Appl. Intell. 49 (2019) 3947–3964.

[37] G. Xu, et al., Particle swarm optimization based on dimensional learning strategy, Swarm Evol. Comput. 45 (2019) 33–51.

[38] H. Iima, Y. Kuroe, S. Matsuda, Swarm reinforcement learning method based on ant colony optimization, in: IEEE International Conference on Systems, Man and Cybernetics, 2010, pp. 1726–1733.

[39] T. Wauters, K. Verbeeck, P. De Causmaecker, G. Vanden Berghe, Boosting metaheuristic search using reinforcement learning, in: Hybrid Metaheuristics. Studies in Computational Intelligence, Vol. 434, Springer, 2013, pp. 1–23.

[40] J. Oh, M. Hessel, M.W, et al., Discovering reinforcement learning algorithms, in: 34th Conference on Neural Information Processing Systems, NeurIPS 2020, Vancouver, Canada, 2020, pp. 1–19, arXiv:2007.08794v2.

[41] N. Mittal, U. Singh, B.S. Sohi, Modified grey wolf optimizer for global engineering optimization, Appl. Comput. Intell. Soft Comput. 2016 (2016) 1–17.

[42] D. Pandit, L. Zhang, S. Chattopadhyay, C.P. Lim, C. Liu, A scattering and repulsive swarm intelligence algorithm for solving global optimization problems, Knowl.-Based Syst. 156 (2018) 12–42.

[43] F. Al-Obeidat, N. Belacel, B. Spencer, Combining machine learning and metaheuristics algorithms for classification method PROAFTN, in: Enhanced Living Environments, in: Lecture Notes in Comput. Sci., vol. 11369, Springer, 2019, pp. 53–79.

[44] H. Samma, J. Mohamad-Saleh, S.A. Suandi, B. Lahasan, Q-learning-based simulated annealing algorithm for constrained engineering design problems, Neural Comput. Appl. 32 (9) (2020) 5147–5161.

[45] Q. Chen, M. Huang, Q. Xu, H. Wang, J. Wang, Reinforcement learning-based genetic algorithm in optimizing multidimensional data discretization scheme, Math. Probl. Eng. 2020 (2020) 1–13.

[46] S. Gupta, K. Deep, Improved sine cosine algorithm with crossover scheme for global optimization, Knowl.-Based Syst. 165 (2019) 374–406.

[47] Y. Xu, D. Pi, A reinforcement learning-based communication topology in particle swarm optimization, Neural Comput. Appl. (2019) 1–26.

[48] M. Alirezanejad, R. Enayatifar, H. Motameni, H. Nematzadeh, GSA-LA: gravitational search algorithm based on learning automata, J. Exp. Theor. Artif. Intell. (2020) 1–17.

[49] F. Liu, G. Zeng, Study of genetic algorithm with reinforcement learning to solve the TSP, Expert Syst. Appl. 36 (3) (2009) 6995–7001.

[50] E. Emary, H.M. Zawbaa, C. Grosan, Experienced gray wolf optimization through reinforcement learning and neural networks, IEEE Trans. Neural Netw. Learn. Syst. 29 (3) (2017) 681–694.

[51] M. Ghafoorian, N. Taghizadeh, H. Beigy, Automatic abstraction in reinforcement learning using ant system algorithm, AAAI Spring Symposium Series, 2013, pp. 9–14.

[52] C. Qu, W. Gai, M. Zhong, J. Zhang, A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning, Appl. Soft Comput. 89 (2020) 106099.

[53] M.M. Mafarja, S. Mirjalili, Hybrid whale optimization algorithm with simulated annealing for feature selection, Neurocomputing 260 (2017) 302–312.

[54] H. Samma, C.P. Lim, J.M. Saleh, A new reinforcement learning-based memetic particle swarm optimizer, Appl. Soft Comput. 43 (C) (2016) 276–297.

[55] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L.C. Jain, A.K. Nagar, Realization of an adaptive memetic algorithm using differential evolution and Q-learning: a case study in multirobot path planning, IEEE Trans. Syst. Man Cybern. Syst. 43 (4) (2013) 814–831.

[56] Y.J. Gong, J.J. Li, Y. Zhou, Y. Li, H.S. Chung, Y.H. Shi, J. Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46 (10) (2017) 2277–2290.

[57] M. White, Unifying task specification in reinforcement learning, in: The Thirty-fourth International Conference on Machine Learning, ICML'17: Proceedings of the 34th International Conference on Machine Learning, 70, 2017, pp. 3742–3750.

[58] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Vol. 635, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore, 2013, pp. 1–32.

[59] J.J. Liang, B.Y. Qu, P.N. Suganthan, Q. Chen, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-Based Real-Parameter Single Objective Optimization, Vol. 29, Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014, pp. 625–640.

[60] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.

[61] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (2001) 481–506.

[62] M. Molga, C. Smutnicki, Test functions for optimization needs, Test functions for optimization needs, 2005, 2005.

[63] X. Yang, Test problems in optimization, 2010, pp. 1–6, arXiv preprint arXiv:1008.0549.

[64] M. Jamil, X. Yang, A literature survey of benchmark functions for global optimization problems, Int. J. Math. Modell. Numer. Optim. 4 (2) (2013) 1–47.

[65] F.V. Bergh, A. Engelbrecht, A study of particle swarm optimization particle trajectories, Inf. Sci. 176 (2006) 937–971.

[66] T. Çavdar, M. Milani, R.A. Milani, A new heuristic approach for inverse kinematics of robot arms, Adv. Sci. Lett. 19 (1) (2013) 329–333.