Original Manuscript

Journal of LOW FREQUENCY NOISE, VIBRATION AND ACTIVE CONTROL

Journal of Low Frequency Noise, Vibration and Active Control
2024, Vol. 0(0) 1–24

DOI: 10.1177/14613484241242737
journals.sagepub.com/home/lfn
Sage

Opposition-based manta ray foraging algorithm for global optimization and its application to optimize nonlinear type-2 fuzzy logic control

Ahmad Azwan Abdul Razak[1], Ahmad Nor Kasruddin Nasir[1], Nor Maniha Abdul Ghani[1] and Mohammad Osman Tokhi[2]

Abstract

Interval Type-2 Fuzzy Logic Control (IT2FLC) possesses a high control ability in a way that it can optimally handle the presence of uncertainty in a system dynamic. However, the design of such a control scheme is a challenging task due to its complex structure and nonlinear behavior. A Manta Ray Foraging Optimization (MRFO) is a promising algorithm that can be applied to optimize the control design. However, MRFO still suffers the local optima problem due to unbalance exploration-exploitation of the MRFO agents and hence limiting the performance of the desired control. In this paper, Standard, Quasi, Super, and Quasi-Reflected opposition strategies are integrated into the MRFO structure. Each strategy enhances the exploration-exploitation capability and offers different approaches of varying agent's step size relative to the algorithm's iteration. The proposed opposition-based MRFO (OMRFO) algorithms are applied to optimize the IT2FLC control design for a laboratory-scaled inverted pendulum system. Moreover, as the algorithms are also promising strategies to other problems, they are applied to solve 50D of 30 IEEE CEC14 benchmark functions representing problems with different features. Performance analysis of the algorithms is statistically conducted using Wilcoxon sign rank and Friedman tests. The result shows that the performance of MRFO and Quasi-Reflected-OMRFO are equal, while all other OMRFO variants show a significant improvement and better rank over the MRFO. The Super and Quasi OMRFO-IT2FLC schemes acquired the best responses for the cart and pendulum, respectively.

Keywords

manta rays foraging, opposition based learning, inverted pendulum, interval type-2 fuzzy logic, parameter optimization

Introduction

Metaheuristic algorithm is an advanced type of heuristic algorithm.[1] Specifically, it is a repetitive procedure of gradient-free and sub-ordinated heuristic, which is a combination of a simple local search strategy and a nature-inspired search mechanism. Metaheuristic algorithm becomes a more demanding tool to solve many engineering and other optimization-related problems due to the reliability and optimal accuracy of its solution. The most common strategy of metaheuristic algorithm is inspired from the biological concept of many living organisms and creatures. Table 1 shows some of the well-known biological-inspired metaheuristic optimization algorithms developed by researchers worldwide. These algorithms are different than each other in a way they are formulated to explore an optimal solution in a pre-defined search space. Genetic Algorithm (GA) is formulated based on the strategy of evolution of a genetic population such as mutation and

[1]Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan, Malaysia
[2]School of Engineering, London South Bank University, London, UK

Corresponding author:
Ahmad Nor Kasruddin Nasir, Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan Campus, Pekan, Pahang 26600, Malaysia.
Email: kasruddin@umpsa.edu.my

crossovers,[2] Particle Swarm Optimization (PSO) is based on the social interaction of a group of birds,[3] Bacterial Foraging Algorithm (BFA) is based on the foraging strategy of a group of bacteria in a human intestine,[4] and Chimp Optimization Algorithm (ChOA) is based on a group hunting strategy of a Chimpanzee population.[5,6] Although they have a unique formula as compared to others, they also share the same criteria in their strategy. First, they are commonly found as a group-based searching strategy. There is communication between the agents to share and exchange individual information within their population. Another common criterion of these algorithms imposed in their concept is an elitism strategy. A currently best-found agent guides the movement of all other agents in the next iterations. The biological feature of the elite agent changes in every iteration and it makes the motion of the search agents more dynamic. Exploration and exploitation are also common criteria of these algorithms that are imposed in their strategy. They complement each other in which an agent moves faster or slower if it is located at a far away or a closer location to the currently best-found solution. The exploration and exploitation are commonly found at the beginning and end of the searching process, respectively. These common criteria are also some of the general reasons of the effectiveness of metaheuristic algorithm in producing a promising solution for various global optimization problems, and hence, gaining popularity as the main tool in solving many complex and real-world problems.[7,8] The research in this field is growing significantly. This is evidenced by the rising of various improved strategies to tackle a stagnant problem and unbalance exploration-exploitation. These include an incorporation of a fuzzy technique into Chimp,[9] Grasshopper,[10] Whale,[11] and Slime Mould[12] optimization algorithms for solving classification and recognition problems.

In recent years, a relatively new metaheuristic algorithm inspired from a cartilaginous fish called a Manta Ray Foraging Optimization (MRFO) has been introduced.[21] Manta ray is a large marine creature found mostly in the Indian Ocean, and tropical, sub-tropical, and warm oceans. Its body has a flat surface from head-to-tail and contains a pair of cephalic lobes located on their large mouth. It feeds on plankton which is a living microorganism in the ocean. Manta ray channels the plankton into its mouth by using its cephalic lobe and filters the plankton from the water using its gill. A matured manta ray requires up to 5 kg of plankton per day. The plankton location depends on the flow of the ocean tides and varies over the seasons. Amazingly, due to its foraging strategy, manta rays always find enough food even though the plankton locations are scattered in the ocean.

The first foraging strategy of manta rays is known as Chain foraging. Manta rays hunt in a group of up to 50 members.[22] They hunt in a line-up position; one behind another, forming a uniform line. Naturally, a small male manta ray hops on and swims on top of the female in order to match the rhythm of the female's pectoral fins movement.[23] If there is any plankton missed by any manta ray in the front, the manta ray that follows behind will scoop up the plankton. This collaboration strategy draws a larger number of planktons into their mouth, and thus, improves their food bounty. Second, a manta ray employs the Cyclone foraging in its strategy. Once it has found an area that is full of plankton, all of the manta ray members in the group move closer toward each other. Next, they start to link-up to each other using their tails and heads to create a spiraling vertex in the form of a cyclone. In this approach, manta rays move toward the water surface while doing the spiral movement. This strategy is very important to trap and force plankton to become more concentrated in a closed area for ease of feeding. The third foraging strategy of manta rays is Somersault foraging. It is one of the uttermost magnificent settings in nature. Somersault is a series of backward movements and circling around the plankton area in order to draw the plankton toward them. Somersault features random, continual, local, and repetitive movements, which help the manta rays to maximize their feeding. The strategy of the MRFO algorithm mimics the Chain, Cyclone, and Somersault foraging strategies of the manta ray fish.[21]

Table 1. Metaheuristic optimization algorithm.

No.	Algorithm	Ref.
1	Genetic Algorithms (GA)	2
2	Particle Swarm Optimization (PSO)	3
3	Bacterial Foraging Algorithm (BFA)	4
4	Chimp Optimization Algorithm (ChOA)	5,6
5	Seagull Optimization Algorithm (SOA)	13
6	Whale Optimization Algorithm (WOA)	14,15
7	Ant Lion Optimizer (ALO)	16
8	Lion Optimization Algorithm (LOA)	17
9	Social Mimic Optimization Algorithm (SMO)	18
10	Snap-drift Cuckoo Search (SCS)	19,20

Literature states that the MRFO algorithm has a competitive performance over other well-known optimization problems in solving global optimization problems.[24,25] These include unimodal, multi-modal, hybrid, and composite problems that have various fitness landscapes.[21] This is due to the unique randomness strategy, a combination of linear and spiral motions with an elitism concept in its Chain, Cyclone, and Somersault foraging. The promising performance of MRFO has attracted many researchers around the world to adopt the algorithm in solving various real-world problems.[26–28] However, the algorithm still has limitations and suffers from stagnation problems; hence, unable to give the best accuracy solution. The search agents of MRFO insufficiently explore and exploit the whole feasible region. In both Chain and Spiral foraging strategies, agents move toward the region that consists of the current fittest agent and the front agent. There is a high possibility that the agents miss a better solution located on the mirrored sides or any opposite region in the feasible search space. Due to the heuristic nature of the algorithm, the current best-found solution is still not a guarantee that the solution leads other agents to the global optimal solution in the feasible region. This obviously occurs in a multi-modal fitness landscape in which there exist both local and global optimal solutions at various points in the region. A good strategy to encounter the problem is by improving the exploration and exploitation of search agents from the beginning until the end of the search operation. This is done through the incorporation of the opposition concept into the MRFO algorithm.

The term "opposite" is defined as "being the other of two complementary or mutual exclusive thing" and the term "oppositional" is defined as "placement opposite to or in contrast with other."[29,30] Through the opposition concept, a novel algorithm that is known as the Opposition-Based Learning (OBL) is introduced. The main idea of OBL is the concurrent evaluation of a solution candidate and its analogous opposite candidate in the feasible search area.[31] This mechanism enhances the exploration and exploitation strategies in such a way that a single agent can be used to evaluate two different locations in the search area. The two locations are the current and mirrored locations of the agent where the center point of the search area is taken as the mirror line. The OBL scheme alone, however, is useless. It is just an opposition-learning scheme which provides a solution to determine a mirrored-location of an agent and is commonly used with an optimization algorithm. The OBL complements the drawback of exploration and exploitation strategies imposed in the algorithm. Several opposition variants have been introduced in literatures to improve the original OBL. They are different than each other in a way that the opposite location of a current agent is determined. Different types of random features are incorporated into the OBL. The well-known OBL variants include Super, Quasi, and Quasi-Reflected Oppositions. Many researchers utilized OBL as a complementary strategy in soft computing technique. It is proven as a good learning scheme to enhance the accuracy performance of metaheuristic algorithms like GA,[32] PSO,[33] and DE.[34,35] There are also works on incorporating OBL to speed up convergence rate as well as to improve accuracy of the parent algorithm.[36,37] The authors incorporated greedy and weighted-opposition methods into Chimp optimization algorithm. The algorithm was tested on various real-world problems and showed superior performance over the original algorithm and other state-of-the-art algorithms. OBL also has been applied successfully in other applications including opposition-based reinforcement learning[38,39] and opposition-based neural network.[40]

The hybrid strategy between the OBL concept and MRFO algorithm is a good potential solution to solve a complex fuzzy control problem in engineering and robotics. An inverted pendulum system is a robotic system that has a typical control problem and is commonly used in control engineering.[41] The inverted pendulum system consists of an inverted pole or pendulum hinged at a center body of a cart. The cart moves horizontally on a guided track while the pendulum rod freely rotates in a 360° direction around its axis. The inverted pendulum system is a nonlinear and highly unstable system.[42,43] At rest, the pendulum rod is pointing vertically downward due to the attraction of gravitational force on the pendulum mass. However, when in operation mode, the pendulum rod should be kept in a vertically upright position while the cart moves to a predetermined distance along its horizontal axis. Example applications of the pendulum system are a two-wheeled human transporter system and a two-wheeled mobile wheelchair. Launching a space shuttle rocket into the air is another sophisticated example application of the inverted pendulum system. Controlling the highly unstable inverted pendulum system with a complex structure of Interval Type-2 Fuzzy Logic Control (IT2FLC) is more challenging. The IT2FLC model is an advanced version of a fuzzy model and other closed-loop control strategies.[44] Its universe of discourse comprises of lower and upper boundaries.[45–47] It added more complexity into the fuzzy model but has a more promising performance to handle the uncertainty of a controlled robotic system. The nonlinear relationship between the fuzzy input–output and a complex fuzzy model structure makes the design of the fuzzy control a highly challenging work. This includes determining fuzzy variables such as if-then rules, universe of discourses, firing angle, and fuzzy input–output gains. Through solely expert knowledge, the performance of a designed fuzzy control might not be at an optimal level. An OBL-based MRFO is a good strategy of optimization algorithm with a balanced exploration and exploitation and is a potential solution to solve the aforementioned complex control problem.

This paper presents Opposition-based Manta Ray Foraging Optimization (OMRFO) algorithms for global optimization and to solve IT2FLC design for an inverted pendulum system. Opposition-based learning (OBL) schemes are incorporated

into the MRFO algorithm to enhance exploration and exploitation capabilities of the algorithm. Four variants of OMRFO are presented in the work, which comprises of Standard (St-) opposition, Quasi (Q-) opposition, Quasi-Reflected (QR-) opposition, and Super (S-) opposition. The objective of the paper is twofold. First, it shows the superiority of the proposed OMRFO over its parent algorithm in solving global optimization problems. This is shown in the performance test of the proposed algorithms in solving 30 black-box global optimization problems of CEC14 benchmark functions. The second objective is to show the superiority of the proposed algorithms in solving a complex structure of the IT2FLC model for a highly unstable inverted pendulum system. It is a real-life control problem in engineering, robotics, and control areas. The proposed algorithms optimize the structure of the IT2FLC, which includes its fuzzy if then-rules, universe of discourse, and input-output gains. The organization of this paper is as follows. The remaining sections of the paper explain the concept, structure of the IT2FLC, and its block diagram used for optimization and control for the inverted pendulum system; the details of the OBL, MRFO and the proposed OMRFO algorithms; the experimental setup, result and discussion for both benchmark functions; and finally the application to optimize the IT2FLC model. The paper ends with a conclusion of the work presented in the paper.

OMRFO-IT2FLC for an inverted pendulum system

An inverted pendulum system

The free-body diagram of an inverted pendulum system used in the work is shown in Figure 1. Its mechanical and electrical parameters are shown in Table 2. The inverted pendulum system consists of a cart that moves forward and backward in a translational motion on a guiding rail and a pole that is pivoted on the moving cart.[33] At rest, the loose end of the pole is pointing downward. This is due to the gravitational force that is imposed on the pendulum body. As the cart moves horizontally back and forth along the guided rail, the pivoted pendulum also moves linearly following the cart motion while at the same time rotates about 360° around its axis. The rotating axis of the pendulum is perpendicular to the side plane of the cart body and guiding rail of the cart. During operating mode, the cart is set to move to a certain distance while the pendulum should be kept in a vertically upright position at all times. A DC motor actuator is connected to the cart through a ball screw and drives the cart linearly along its axis. An encoder sensor is attached at the DC motor shaft to measure the actual linear position of the cart. A second encoder sensor is attached on the pivoted pendulum to measure the rotational position of the pendulum. The DC motor input is a continuous voltage signal from a controller. The outputs of the system are the rotational and linear positions of the pendulum and cart, respectively. The configuration defines the system as a single-input multi-output system.[48–50]

The mathematical model of the inverted pendulum system is derived using the Newtonian second law based on the schematic shown in Figure 1.[51,52] Dynamic equations of the system can be simplified as (1) and (2).

$$(M+m)\ddot{x} + ml\ddot{\theta} + F_r\dot{x} = F_V V \tag{1}$$

$$m\ddot{x} + ml\ddot{\theta} = mg\theta \tag{2}$$

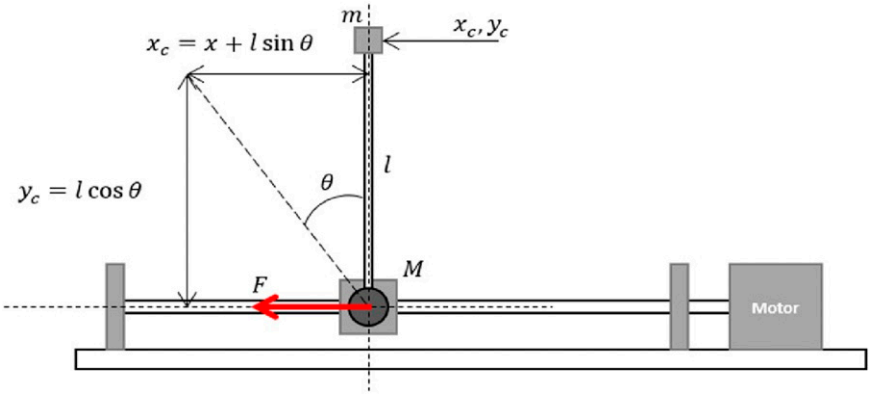

Figure 1. Free-body diagram of an inverted pendulum system.

Interval type-2 fuzzy logic control

The type-2 fuzzy logic (T2FL) model is an extension of a conventional fuzzy logic model or also known as type-1 fuzzy logic (T1FL) model. The major difference between type-1 and type-2 fuzzy logic models is shown by the fuzzy set $\tilde{A}$. Unlike the T2FL, the fuzzy set of T1FL has only a single membership. Another difference of T2FL over the T1FL is the defuzzification process.[53,54] It requires a Type Reducer which converts the inference output of type-2 to type-1 fuzzy sets before the final or crisp output can be calculated using a defuzzifier.[55–57]

The general process of the T2FL model consists of a fuzzifier, an inference mechanism, a type reducer, and a defuzzifier. The process begins by fuzzifying the input crisp value of IT2FLC, which is converting the input value into type-2 fuzzy set using a fuzzifier. The general representation of type-2 fuzzy set is depicted in Figure 2. It consists of lower $u_{\overline{A}}$ and upper $\overline{u}_{\overline{A}}$ membership functions which determine the minimum and maximum boundaries of the fuzzy set, respectively. Both upper and lower boundaries define the primary ($\overline{\mu}_{\overline{A}}(x)$) and secondary $u_{\overline{A}}(x)$ membership functions. The area between the boundaries is an uncertainty region and is considered as Foot of Uncertainties (FOU). The configuration enables the type-2 fuzzy set encounter uncertainty more efficiently than the type-1 fuzzy set.[58]

The interval type-2 fuzzy set is defined as (3).

$$\tilde{A} = \left(\int_{x \in X} \int_{u Jx \subseteq [0,1]} \frac{1}{x, u} = \int_{x \in X} \left[\int_{u \in Jx \subseteq [0,1]} \frac{1}{u} \right] \right) / x \tag{3}$$

Table 2. Parameter values of the inverted pendulum system.

Parameter	Value
Mass of cart, M	0.1 kg
Mass of pendulum, m	0.05 kg
Friction or cart, b	$0.1 Nm^{-1}s^{-1}$
Length of pendulum, l	0.3 m
Motor torque constant, K_m	$4.9 Ncm A^{-1}$
Motor back emf constant, K_b	$0.0507 V rad^{-1}s^{-1}$
Motor armature resistant, R	0.3Ω

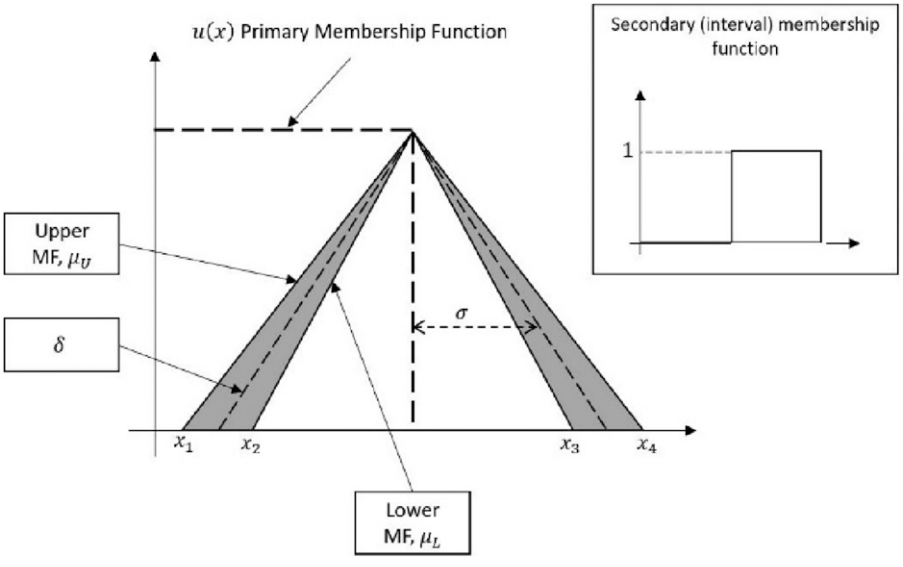

Figure 2. Interval type-2 fuzzy set.

where x is the primary variable, u is a secondary variable, and Jx is a domain for each variable called the primary membership of x, which is defined as (4).

$$Jx = \left\{ (x,u) : u \in \left[u_{\overline{A}}(x), \overline{u}_A(x) \right] \right\} \tag{4}$$

where $u_{\overline{A}}(x)$ and $\overline{u}_{\overline{A}}(x)$ are the lower and upper membership functions, respectively, which are defined as (5) and (6).

$$u_{\overline{A}}(x) = FOU\left(\overline{A}\right) \forall x \in X \tag{5}$$

$$\overline{u}_A(x) = FOU\left(A\right) \forall x \in X \tag{6}$$

where FOU is the union of the primary membership of fuzzy set $\tilde{A}$ and is defined as (7).

$$\delta\left(\tilde{A}\right) = \cup_{\forall x \in X} Jx = \{(x,u) : u \in Jx \subseteq [0,1]\} \tag{7}$$

The inference mechanism interprets the fuzzy value based on a set of predefined fuzzy rules. In the work, a decomposition approach is utilized for both fuzzifier and inference mechanism.[48] They are decomposed into upper and lower parts which comprise of upper and lower membership functions, respectively. Three fuzzy sets are defined as Negative Small (NS), Zero (Z), and Positive Small (PS) to form fuzzy rules which consist of antecedent and consequence parts. The fuzzy rules are made from IF-THEN statements, which is generally represented as (8).

$$\text{IF } '\!x \text{ is and } y \text{ is } B', \text{ then } '\!z \text{ is } c' \tag{8}$$

where 'x is A and y is B' is the antecedent and 'z is C' is the consequence. Considering an error and a derivative of the error as the input to the T2FL, nine fuzzy rules are generated.

A type reducer is then utilized to convert outputs of the inference mechanism engine to a type-1 fuzzy output. The type-reducer can be mathematically expressed as (9).

$$Y = \left(\sum_{n=1}^{N} Y^n \left(f^n + \overline{f}^n \right) \right) \Big/ \left(\sum_{n=1}^{N} \left(f^n + \overline{f}^n \right) \right) \tag{9}$$

where Y is the crisp output, Y^n is the n^{th} output of fuzzy set, and f^n and $\overline{f^n}$ are the lower and upper membership functions.

The final process of the T2FL is to defuzzify the type-1 fuzzy output to a crisp value. This is accomplished by computing the center-of-area (centroid) of the FOU using a geometrical approach. It represents the area under the upper and lower membership functions and is computed using (10).

$$y = (c_U A_U - c_L A_L)/(A_U - A_L) \tag{10}$$

where A_U and A_L are the area under the upper and lower output membership functions, respectively, while c_U and c_L are the centroid of the upper and lower output membership function, respectively. Both centroids are calculated using the T1FL centroid computation method. As the outcome, if both sets are equal, then there is no FOU and the system reduces to an equivalent T1FL system. In this case, the crisp output is simply equivalent to c_U which is also identical to c_L.

OMRFO-based interval type-2 fuzzy logic control

The closed-loop block diagram of the OMRFO-based IT2FLC scheme for controlling the inverted pendulum system is illustrated in Figure 3. The outputs of inverted pendulum system are an inverted pendulum's angle, θ, and a cart's position, x, while the input is an electrical voltage, v, signal from the controller. In the work, the desired pendulum's angle was set as 0°. This is to ensure the pendulum is pointing at a vertically upright position during the control mode operation. Considering the maximum range of the cart rail is [-15, 15] cm, the desired cart's position was set to 10 cm. The cart is forced to move 10 cm away from its original position, which is located at 0 cm. It has to maintain the position for a certain period of time while balancing the inverted pole in an upright position. The actual response of the cart is compared with the desired value.

The difference between the actual and desired response is considered as the error of the cart's position and is defined as (11).

$$e_{cart}(t) = x_{act}(t) - x_{des}(t) \tag{11}$$

where $x_{act}(t)$ and $x_{des}(t)$ are the actual and desired responses of the cart, respectively.

The actual response of the pendulum is fed back and considered as the error of the pendulum's angle, which is defined as (12).

$$e_{pend}(t) = \theta_{act}(t) \tag{12}$$

where $\theta_{act}(t)$ is the actual response of the pendulum.

The errors of the pendulum's angle and the cart's position are set as the first input of the first IT2FLC (IT2FLC-1) and the second IT2FLC (IT2FLC-2), respectively. The derivative of the errors of the pendulum's angle and the cart's position are set as the second input of the IT2FLC-1 and IT2FLC-2, respectively. Each of these two controllers produces a required value of a voltage signal, which is then combined and fed into the inverted pendulum.

From Figure 4, the mapping of input for the controllers for both error (rad) and derivative of error (rad/sec) is shown. The x-axis range between $[-5,5]$ means that the algorithms were ran to optimize the parameters in that range while y-axis shows that the peak of the signal is between $[0,1]$. Furthermore, N, Z and P are the negative, zero, and positive ranges of the output signal, while σ and $\Delta\mu$ are the parameters referred as standard deviation and mean, respectively, which are to be optimized in the study. The following constraints then were applied, $\mu_1 < \mu_2 < \mu_3$ and $\sigma_L < \sigma_U$, where μ_1, μ_2, and μ_3 are defined as mean for membership functions 1, 2, and 3, respectively, while σ_L and σ_U are defined as standard deviation for the lower and upper membership functions, respectively.

In the work, the proposed OMRFO variants were utilized to optimize both IT2FLCs' performance. The optimized parameters of the controllers include the location and width of all upper and lower fuzzy membership functions on the universe of discourse and fuzzy rules. Each controller comprises of four variables used to vary the location and width of the membership functions on both fuzzy inputs and nine fuzzy rules. Considering both controllers, the total optimized parameters involved in the work were 26 parameters. The optimization process was initiated by feeding both errors from the cart and pendulum responses into the OMRFO. The summation of these two errors is called the cost function of the OMRFO and the root mean square (RMS) was adopted to complete the formula. Two constants w_1 and w_2 are defined in the cost function to control the weightage of each error. A complete cost function, f_{cost} of the OMRFO is shown in

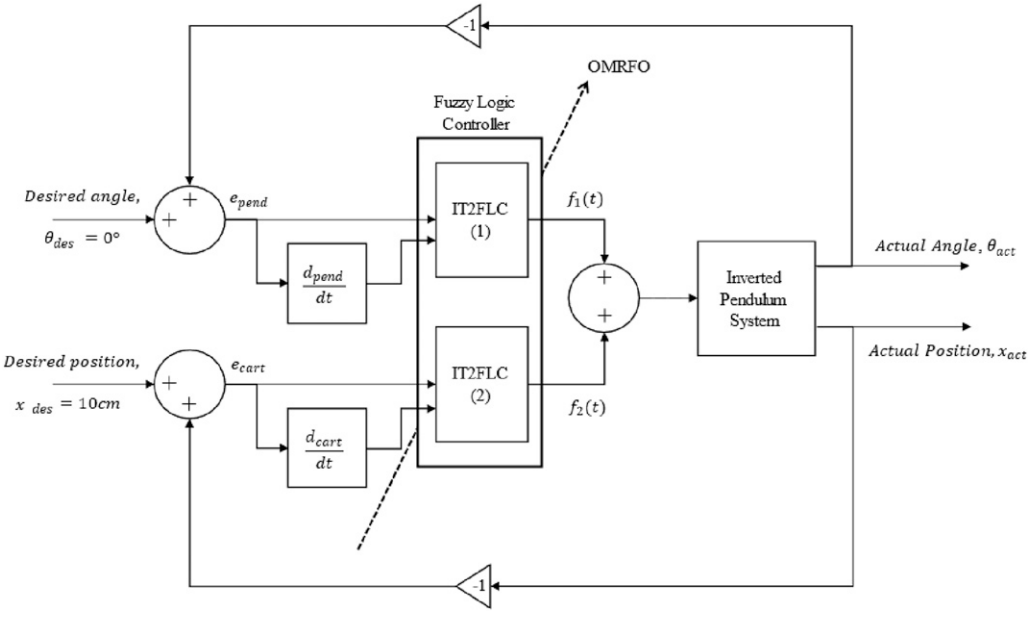

Figure 3. Block diagram of the OMRFO-based interval type-2 fuzzy logic control.

$$f_{\cos t}(t) = w_1\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(e_{pend}{}^2(t)\right)} + w_2\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(e_{cart}{}^2(t)\right)} \tag{13}$$

where w_1 is the weightage of the pendulum's error and w_2 the weightage of the cart's error. In the work, both constants were defined as 0.5. The controllers will reach the optimal performance if the cost function is at the minimum value.

Opposition-based manta ray foraging optimization

Manta ray foraging optimization

The MRFO algorithm mimics the behavior of a group of Manta Ray population in the ocean in finding an area with high concentration of plankton. Searching agents in MRFO swim in a predefined feasible region to search for an optimal solution. The optimal solution is also known as the best solution in MRFO, which is equivalent to an area with the highest concentration of plankton in the ocean. All Manta Rays are considered as searching agents while an individual Manta Ray that finds an area with the highest concentration of plankton is considered as the fittest agent in the algorithm. There are three foraging strategies adopted in the algorithm, which are known as Chain, Cyclone, and Somersault foraging.

The Manta Ray foraging strategy is considered as a group-based strategy where there can be up to a maximum of 50 members during the foraging process. The first foraging process of the Manta Rays is called Chain foraging. As the name implies, the Manta Rays line up and form a chain by linking an individual Manta Ray to another individual Manta Ray. In this form, they swim and scoop all the planktons that come their way. Figure 5 depicts the formation of the Chain foraging of Manta Rays in a feasible region. It shows that the fittest agent, x_{best} serves as a guide to their formation while all other agents move based on the locations of the fittest agent, x_{best} and its front agent, x_{i-1}. As an example, the $x_i^d(k)$ moves forward based on the information of the fittest agent, x_{best} and its front agent $x_{i-1}^d(k)$. The resultant location of the i^{th} agent is $x_i^d(k+1)$. Apart from that, x_{best} and $x_{i-1}^d(k)$ help the algorithm to determine the moving step size of the i^{th} Manta Ray. Noted from the figure, the red-diamond is the fittest solution found so far $x_{best}^d(k)$, while green-diamonds are the searching agents and yellow-diamonds are the resultant position of the searching agents after undergoing the Chain foraging strategy.

$$x_i^d(k+1) = \begin{cases} x_i^d(k) + r_1\left(x_{best}^d(k) - x_i^d(k)\right) + \alpha\left(x_{best}^d(k) - x_i^d(k)\right), i = 1 \\ x_i^d(k) + r_1\left(x_{i-1}^d(k) - x_i^d(k)\right) + \alpha\left(x_{best}^d(k) - x_i^d(k)\right), i = 2, 3, .., n \end{cases} \tag{14}$$

$$\alpha = 2r_1\sqrt{|\log(r_1)|} \tag{15}$$

where $x_i^d(k+1)$ is the position of the i^{th} agent in $(k+1)^{th}$ iteration of d^{th} dimensional space, $x_i^d(k)$ is the location of the i^{th} agent in $(k)^{th}$ iteration of d^{th} dimensional space, r_1 is the random number between (0,1), $x_{best}^d(k)$ is the fittest agent found so far, and α is the coefficient of chain foraging.

The Cyclone strategy is the second process of the Manta Rays foraging and it is illustrated in Figure 6. The red-diamond is the fittest agent found so far $x_{best}^d(k)$, green-diamonds are the searching agents, and the yellow-diamonds are the next position of the searching agents after undergoing Cyclone foraging. Unlike the Chain strategy, the Cyclone strategy moves an individual Manta Ray through a combination of straight line and spiral forms. As shown in the figure, the i^{th} agent, $x_i^d(k)$

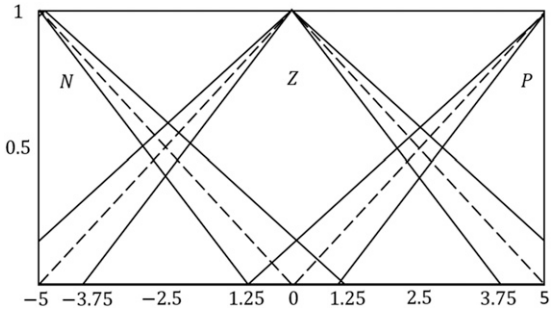

Figure 4. The mapping of input for the controllers for both error (rad) and derivative of error (rad/sec).

moves in a straight-line form based on the information extracted from $x_{i-1}^{d}(k)$. On top of that, it also moves in a spiral form following the information received from x_{best}.

These two motions are mathematically expressed as (16) and (17).[21]

$$x_i^d(k+1) = \begin{cases} x_{best}^d(k) + r_2\left(x_{best}^d(k) - x_i^d(k)\right) + \beta\left(x_{best}^d(k) - x_i^d(k)\right), i = 1 \\ x_{best}^d(k) + r_2\left(x_{i-1}^d(k) - x_i^d(k)\right) + \beta\left(x_{best}^d(k) - x_i^d(k)\right), i = 2, 3, .., n \end{cases} \tag{16}$$

$$\beta = 2e^{r_3\frac{K-i+1}{K}}\sin(2\pi r_3) \tag{17}$$

where r_2 and r_3 are the random numbers between (0,1), β is the coefficient of the Cyclone, and K is the maximum iteration.

The position update equations show that the newly generated position is an updated location of x_{best}. Thus, the strategy demonstrates a good exploitation process within a confined region near to the x_{best} agent. Implementing the strategy alone tends to trap all searching agents to a local optimal location. Instead of updating the x_{best} agent location, the equations update a randomly generated solution based on the information from $x_i^d(k)$ and $x_{i-1}^d(k)$ agents. It is a good strategy for exploring an optimal solution in a diverse or large search space. Therefore, the problem is avoided by modifying the position update equations as shown in (18) and (19).[21]

$$x_{rand}^d = Lb^d + r\left(Ub^d - Lb^d\right) \tag{18}$$

$$x_i^d(k+1) = \begin{cases} x_{rand}^d + r_2\left(x_{rand}^d - x_i^d(k)\right) + \beta\left(x_{rand}^d - x_i^d(k)\right), i = 1 \\ x_{rand}^d + r_2\left(x_{i-1}^d - x_i^d(k)\right) + \beta\left(x_{rand}^d - x_i^d(k)\right), i = 2, 3, ..., n \end{cases} \tag{19}$$

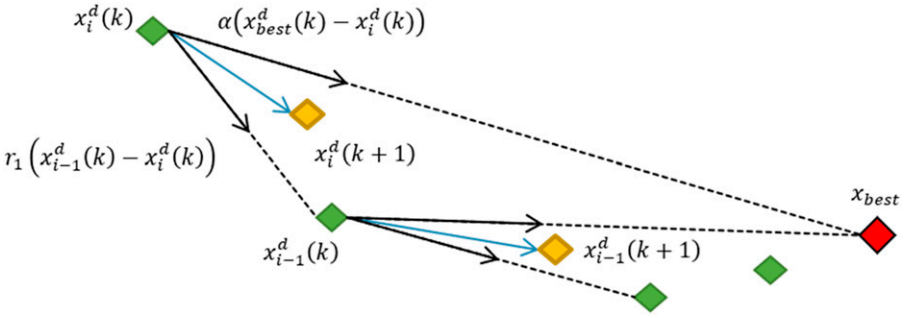

Figure 5. Illustration of chain foraging.

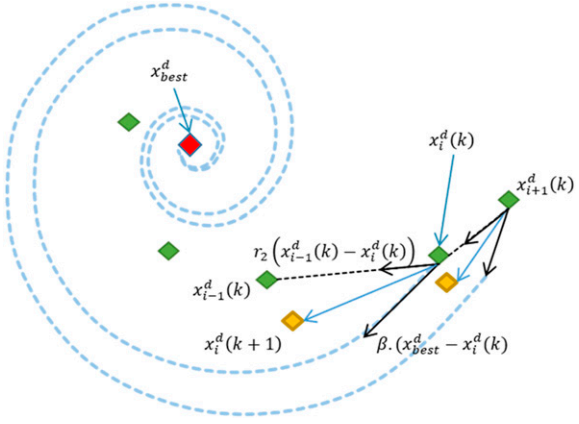

Figure 6. Illustration of cyclone foraging.

where x_{rand} is the random position of the agent, and Ub^d and Lb^d are the upper and lower boundaries of the d^{th} dimension in the feasible region.

The third process of the Manta Ray foraging is Somersault foraging and it is depicted in Figure 7. Somersault foraging is the process where all searching agents move towards the other side of the fittest agent, x_{best} at the same distance between the agent and x_{best}. Taking x_{best} as the pivoting location, all other searching agents are forced to move around x_{best}. Therefore, the Somersault is considered as a good exploitation strategy in finding an optimal solution. The red-diamond is the fittest solution found so far $x^d_{best}(k)$, green-diamonds are the searching agents, and yellow-diamonds are the next position of the searching agents after undergoing Somersault foraging.

The mathematical expression of Somersault foraging is shown as (20).

$$x_i^d(k+1) = x_i^d(k) + S\left(r_1.x^d_{best} - r_2.x_i^d(k)\right), i = 1, 2, ..., N \tag{20}$$

where S is the coefficient of Somersault foraging and is defined as 2, and r_1 and r_2 are random numbers between (0,1). Noted that S is defined as 2 where it results in the next position, $x_i^d(k+1)$ of the i^{th} agent which mirrors its current location, $x_i^d(k)$ at the pivoting point.

Oppositional-based learning

The general idea of OBL is to check for a possible optimal solution on the opposite side of the current location of an agent. Taking the boundaries of a feasible search space as both upper and lower bounds, the middle point of the search space can be determined. An opposite location of an agent is defined as a location of the agent mirrored at the middle point of the lower and upper bounds of the search space. An illustration of the idea is shown in Figure 8.

Consider a and b as the lower and upper bounds of the feasible search space, respectively, and c is the middle point between the boundaries. A searching agent is represented as $\hat{x}$ while its opposite location is stated as $\hat{x}_O$. The middle point between the boundaries is determined using the equation shown in (21).[59]

$$c = (a+b)/2 \tag{21}$$

Applying (21), the resultant location of the agent is a mirrored location, $\hat{x}_O$ with a fixed distance similar to the distance of the agent, $\hat{x}$ and the middle point, c. The mirrored location, $\hat{x}_O$ is known as a Standard Opposition. The mirrored location of the searching agent is varied by placing the agent closer to the middle point, c or further away from the mirrored location $\hat{x}_O$.

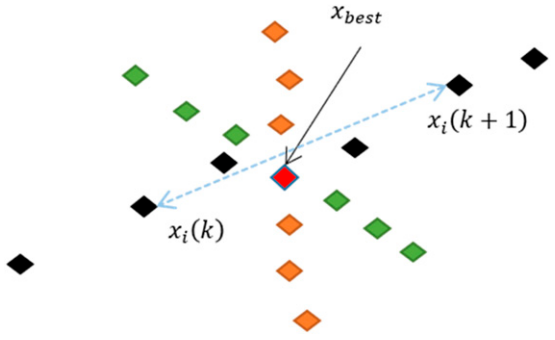

Figure 7. Illustration of somersault foraging.

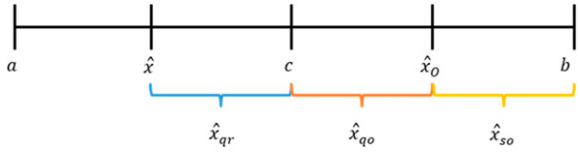

Figure 8. Mapping of opposite individuals based on three types of OBL.

The resultant locations that are closer to the middle point, c are known as Quasi-Opposition, $\widehat{x}_{qo}$ and Quasi-reflected Opposition, $\widehat{x}_{qr}$. The $\widehat{x}_{qo}$ is a mirrored location of the agent between c and $\widehat{x}_O$ while $\widehat{x}_{qr}$ is a mirrored location of the agent between c and $\widehat{x}$. In all cases, the distance of Quasi Opposition schemes is always less than the distance of the standard opposition scheme, $\widehat{d}(c,\widehat{x}_{qo}), \widehat{d}(c,\widehat{x}_{qr}) < \widehat{d}(\widehat{x}_o, c)$. The further away mirrored location that is placed between $\widehat{x}_O$ and b is known as a Super-Opposition, $\widehat{x}_{so}$. In all cases, the distance of the Super Opposition scheme is always greater than the distance of the standard opposition scheme, $\widehat{d}(c,\widehat{x}_{so}) > d(\widehat{x}_o, c)$. Table 3 shows variants of opposition schemes and their corresponding mathematical equations.

Opposition-based manta ray foraging optimization

The OMRFO algorithm is an incorporation of opposition schemes into the original MRFO. The strategy complements several drawbacks of the MRFO algorithm in locating an optimal solution, and thus, offers several advantages which can be described as follows.

First, an integration of the opposition schemes into the standard MRFO improves an exploration strategy in locating an optimal solution. One of the stochastic natures of a group-based optimization algorithm is placing all searching agents into a feasible search space using a random feature. It allows an algorithm to distribute the searching agents thoroughly within a predefined searching boundary. However, the agents are not well and evenly distributed. Some of the small regions within the searching boundaries contain more agents than the other small regions. As the searching operation continues and iteration increases, the exploration of the agents does not thoroughly cover the whole search space. Some small regions might not be touched by the agents, and thus, leaving those regions unexplored. The following illustration elaborates the idea. For the case of MRFO, at the k^{th} iteration of the d^{th} dimensional problem, a group of searching Manta Rays, pop consists of m agents, which can be represented as (26).

$$pop(k) = \left[\widehat{x}_i^d(k), \widehat{x}_i^d(k), \widehat{x}_i^d(k), ..., \widehat{x}_m^d(k)\right] \tag{26}$$

where $i = 1, 2, 3 \ldots m$, $d = 1, 2, 3 \ldots D$, $\widehat{x}_i^d(k)$ represents the location of i^{th} agent in k^{th} iteration on d^{th} dimensional search space. The m and D are the total number of searching agents and dimension of search space, respectively. The locations of all searching agents are initiated onto the search space randomly with respect to two predefined lower and upper boundaries of the feasible search region using .

$$\widehat{x}_i^d(0) = (b - a)^* rand(0, 1] + a \tag{27}$$

where a and b are the upper and lower boundaries, respectively, while rand (0,1] is a random value between [0,1].

An illustration of the initialization of agents' position based on (26) and (27) is shown in Figure 9. The blue-dots are the initial locations of searching agents while the grey area within the dotted line is a feasible search region. The middle line between the a and b boundaries of the feasible region is denoted as c. Noted from the figure that the number of agents on the left side is more than the number of agents on the right side. The situation will result in two possible drawbacks. First, in case an optimal solution is located on the right-side region with fewer agents, a slower convergence rate might be achieved and more computation is needed to achieve the optimal solution. This is because more iterations are required to search for the optimal location. Second, in the region with less agents, there is a higher possibility for the agents to miss certain areas in the feasible search during the search operation with a limited number of iterations. If the optimal solution is located in the unexplored region, the agents will get trapped into other local optimal points, and thus, resulting in a less accurate solution. By generating opposite individuals in every iteration throughout the search operation, the problem can be solved. Figure 9 shows 25 agents that are initialized into the feasible region. The blue-dots are the original searching agents that are located

Table 3. Opposition-based learnings and their mathematical representations.[59,60]

	Types of oppositions	Mathematical representations	
$f_{St}(\widehat{x})$	Standard opposition, St-O	$\widehat{x}_o = a + b - \widehat{x}$	(22)
$f_Q(\widehat{x})$	Quasi-opposition, Q-O	$\widehat{x}_{qo} = \text{rand}[c, \widehat{x}_O]$	(23)
$f_{QR}(\widehat{x})$	Quasi-reflected opposition, QR-O	$\widehat{x}_{qr} = \text{rand}[c, \widehat{x}]$	(24)
$f_{SO}(\widehat{x})$	Super-opposition, SO-O	$\widehat{x}_{qo} = \text{rand}[\widehat{x}, b]$	(25)

in the feasible region, the red-dots are their corresponding opposite locations, while the grey area within the dark-dotted line is the feasible region in which a and b are its lower and upper boundaries, and c is the center of the feasible region. $x_i^d \leftrightarrow x_{i,opp}^d$, $x_{i+1}^d \leftrightarrow x_{i+1}^d opp$ and $x_{i+2}^d \leftrightarrow x_{i+2}^d opp$ are examples of agents and their corresponding opposite locations in the region. The illustration shows how the agents are distributed after undergoing the Standard opposition scheme, *St*-. The scheme generates the opposite location and mirrors the agents at the center line, *c* of the search space. It is noted from the figure that the agents' distribution in the feasible search space is more thorough and balance. Therefore, incorporating the opposition scheme throughout the search operation offers an opposition learning feature to the MRFO.

Second, the advantage of incorporating the opposition scheme into MRFO is that it improves the stochastic feature of the algorithm. Based on *Q*-, *QR*-, and *S*-opposition schemes, an additional random feature is incorporated into the formulas. The mirrored location of the newly generated agents does not follow the same distance of the agents to the reflection point. However, the distance of the resulting location is defined within a certain range with respect to a random number, and thus, improving the exploration capability. The third advantage of the opposition scheme is an adaptive strategy is incorporated to determine the resulting opposite location of the agents. The agent's fitness cost is adopted into the formula to generate the opposite location of the agent. The adaptive feature allows the algorithm to generate a more dynamic mirrored distance for each agent and increase the dynamic behavior of the agents throughout the search operation. In the case where the agent has a smaller cost, the algorithm generates a smaller mirrored distance for the agent and vice versa. In the beginning of the search operation, the fitness costs of all agents are large enough and it generates a large mirrored distance and a good exploration. Towards the end of the iterations, the fitness cost approaches minimum value, and thus, leads to a better exploitation.

In the work, four variants of the opposition schemes, that is, *St*-, *Q*-, *QR*- and *S*-, are incorporated into MRFO. These four variants produce four variants of OMRFO. Each variant is implemented by applying the opposition schemes shown as (22), (23), (24), and (25), respectively. Details of the step-by-step pseudocode of the proposed OMRFO variants are shown in Figure 10. The number of agents in OMRFO is doubled via the opposition equations (22)–(25) where a total of 50 agents are utilized to find a solution similar to the number of agents in MRFO. In order to maintain the same number of agents during the search and for a fair comparison, the agents are sorted based on their fitness cost, and consequently, the first half of the agents' population are retained while the rest are eliminated. In the sorting process, agents with the lowest and highest fitness costs are defined as the first and the last, respectively. In other words, the sorting process retains all the fittest agents in the population. This is also to avoid the increasing complexity of the algorithm which may lead to the increase of computation cost.

Experimental setup, result and discussion

Benchmark functions

Experimental setup and evaluation criteria. This section discusses the experimental test setup and performance of the proposed OMRFO algorithm and its variants tested on a set of global optimization problems. The performance test consists of 30 benchmark functions adopted from the Congress of Evolutionary Computation 2014 (CEC14).[61] They are considered as good test suites to test a newly developed algorithm due to their wide range of fitness landscapes which represent the most likely real-world problems. It covers a wide range from basic to advanced features like unimodal and multimodal, shifted and rotated, separable, non-separable and composition of several basic functions. There are also hybrid functions which

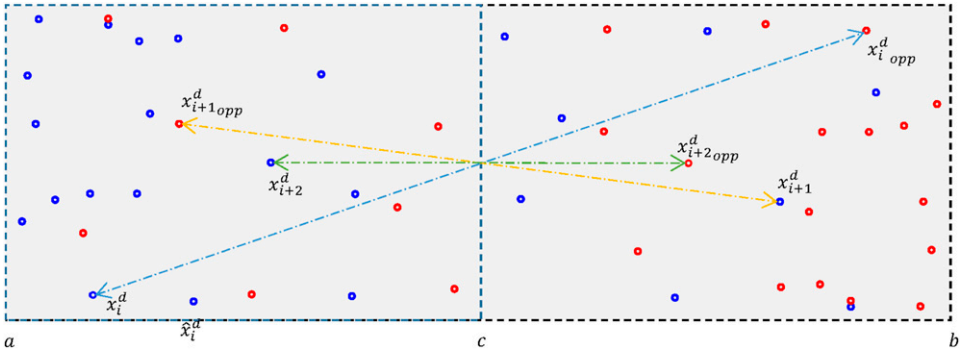

Figure 9. Initial and opposite locations of agents on a 2-dimensional search space.

Initialization phase
Step 1: Initialize population of m searching agents on a feasible search space, pop.
Step 2: Generate opposite location of the population, pop_{opp}. Apply equations (22), (23), (24) and (25).
Step 3: Calculate the fitness cost, $f(x)$ of each agent.
Step 4: Assign the agent with smallest fitness cost, f_{min} as the fittest agent, x_{best}.
Main phase
Step 5: Apply Cyclone Foraging. Use equation (16).
Step 6: Apply Chain Foraging. Use equation (14).
Step 7: Calculate the fitness of each agent and update x_{best}.
Step 8: Generate the opposite location of the population, pop_{opp}. Apply equations (22), (23), (24) and (25).
Step 9: Calculate the fitness cost, $f(x)$ of each agent.
Step 10: Assign the agent with the smallest fitness cost, f_{min} as the fittest agent, x_{best}.
Step 11: Apply Somersault Foraging. Use equation (20).
Step 12: Calculate the fitness cost, $f(x)$ of each agent.
Step 13: Assign the agent with the smallest fitness cost, f_{min} as the fittest agent, x_{best}.
Step 14: Generate the opposite location of the population, pop_{opp}. Apply equations (22), (23), (24) and (25).
Step 15: Calculate the fitness cost, $f(x)$ of each agent.
Step 16: Assign the agent with the smallest fitness cost, f_{min} as the fittest agent, x_{best}.
Termination phase
Step 17: Check if the termination criterion is satisfied and stop the algorithm. Otherwise, return to **Step 5**.
Step 18: Return the x_{best}.

Figure 10. Pseudocode of the proposed OMRFO variants.

combine several advanced features that lead to a more complex fitness landscape. Details of the CEC14 benchmark functions such as mathematical expression, graphical 2-D and 3-D representations can be referred to the CEC14 competition manual.[61] Following the CEC14 manual, the experimental setup for the performance test of the developed OMRFO and original MRFO on CEC14 functions is shown in Table 4. The test was conducted on 50-Dimension problems. A total of 51 runs were consecutively executed for each function. This is to allow for a statistical analysis to be conducted on the acquired result. Since the algorithms are heuristic in nature and the generated result is different from one to another, the statistical analysis leads to a more precise interpretation and conclusion. A stopping condition was set in terms of the number of function evaluation, NFE_{max}. Its value depends on the pre-defined dimension, D of the functions. This is to allow a fair comparison to be performed since the complexity of algorithms is different from one another. OMRFO generally comprises more NFE in one iteration than the MRFO and its searching operation will be forced to a stop once it's NFE has reached the maximum.

The generated result was statistically analyzed via the Wilcoxon sign rank as well as Friedman Tests. The Wilcoxon signed-rank test is a nonparametric statistical hypothesis test to correlate two complementary samples or related samples.[62–67] It is also used to test a paired difference of N repeated measurements on two different samples by assessing their population mean ranks.[68] The Friedman test, on the other hand, is a nonparametric statistical tool that is used to measure performance difference across multiple or more than two algorithms.[69–73] It compares the mean rank of all algorithms under study. Algorithms with the lowest and highest mean ranks are considered the best and worst performing algorithms, respectively. In both tests, a percentage of confidence interval was defined as 95 %. It indicates that if the test gives a two-tailed probability, p less than 0.05 or 5 % level of significance, the performance of one algorithm over another algorithm has a significant improvement. In the work, 51 independent runs were conducted, and thus, the repeated measurements, N, were defined as 51.

Parameter setting. All algorithms under the study generated 51 independent data of achieved accuracy in locating theoretical optimal solution on each function. Based on the generated data, Mean and Standard Deviation (SD) values were calculated to represent the algorithms accuracy and consistency performances, respectively. For the algorithm execution, the common parameters used in both MRFO and OMRFOs are based on the previous work of MRFO.[74–76] The proposed OMRFOs required no extra parameters to generate the opposite agents. It is noted that the proposed algorithms maintained a similar number of user-defined parameters although there were changes in its structure. Table 5 shows the parameters and their corresponding values used in the algorithms for the performance test on benchmark functions.

Table 4. Experimental setup for the performance test on CEC14 functions.

Parameter	Value
Dimension	50
Number of run	51
Max. no. of function eval.,	10000*D
Search range	[−100, 100]
Initialization	Uniform random within the search range
Termination	Maximum number of function evaluation

Table 5. Parameter setting for the performance test on CEC14 functions.

MRFO	St-OMRFO	Q-OMRFO	QR-OMRFO	S-OMRFO
$m = 50$		$m = 50$		
		$k_{max} = 1000$		
		$NFE_{max} = 10000 \times D$		
		$S = 2$		

Performance comparison on result and discussion. The mean and SD results of the algorithms on 50D problems of CEC14 are shown in Table 6. Based on the mean value, MRFO achieved the best performance among the algorithms for functions f_{11} and f_{28} while St-OMRFO achieved the best performance among the algorithms for seven functions, $f_1, f_4, f_6, f_8, f_{17}, f_{20}$, and f_{26}. Q-OMRFO did not achieve any best performance for 50D problems. QR-OMRFO achieved the best performance among the algorithms for nine functions, $f_2, f_3, f_9, f_{10}, f_{18}, f_{19}, f_{21}, f_{22}$, and f_{27}. S-OMRFO achieved the best performance among the algorithms for functions f_{29} and f_{30}. St- and QR-OMRFOs shared the best performance for 10 functions, f_5, f_7, f_{12}–f_{16}, and f_{23}–f_{25}. The results show that QR-OMRFO achieved the unshared best performance with nine scores, followed by St-OMRFO with seven scores. Taking into account both shared and unshared best performances, QR-OMRFO led the number with 19 scores, followed by St-OMRFO with 17 scores. MRFO and St-OMRFO shared the third place with two scores, followed by Q-OMRFO which achieved 0 score. On the contrary, MRFO attained the unshared worst performance among the algorithms for 17 functions which include f_1–f_4, f_6–$f_{10}, f_{15}, f_{17}, f_{18}, f_{20}$–$f_{22}, f_{26}$, and f_{30}. St-OMRFO did not attain any worst performance for 50D problems. Q-OMRFO attained the unshared worst performance among the algorithms for two functions, f_{19} and f_{28}. QR-OMRFO attained the unshared worst performance for functions f_{29} and S-OMRFO attained the least unshared worst performance for function f_{11} and f_{27}. S-OMRFO shared the worst performance with Q-OMRFO and MRFO for functions f_5, f_7, f_{12}–f_{15}, and f_{23}–f_{25}. Taking into account both shared and unshared worst performances, MRFO led the number followed by Q-, S-, QR-, and St-OMRFOs which attained scores of 25, 9, 11, 1, and 0 functions, respectively. These data show that the proposed OMRFOs outperformed MRFO on the accuracy performance for 50D of CEC14 problems.

On the other hand, based on the SD value, MRFO attained the unshared worst performance among other algorithms for 18 functions which include f_1–f_4, f_7, f_8, f_{12}–f_{14}, f_{16}–f_{22}, f_{27}, and f_{28}. Q-OMRFO attained the unshared worst performance among the algorithms for 6 functions, $f_5, f_6, f_9, f_{15}, f_{27}$, and f_{29}. S-OMRFO attained the unshared worst performance among the algorithms for three functions, f_{10}, f_{11}, and f_{28}. St- and QR-OMRFOs did not attain any unshared worst performance for 50D problems. All algorithms shared the worst performance for functions f_{23}–f_{25}. The worst performance for function f_{26} was shared by Q- and S-OMRFOs. Considering both shared and unshared, the results show that MRFO led the worst performance on the consistency achievement followed by Q- and S-OMRFOs with the scores 17, 11, and 9, respectively. QR- and St-OMRFOs shared the fourth position on the worst performance with three scores. On the contrary, MRFO achieved the unshared best performance among the algorithms for two functions which include f_{27} and f_{28}. St-OMRFO achieved the unshared best performance among the algorithms for 11 functions, f_1, f_4, f_{10}–f_{12}, f_{14}, and f_{18}–f_{22}. Q- and S-OMRFOs did not achieve any unshared best performance for 50D problems. QR-OMRFO achieved the unshared best performance among the algorithms for 11 functions, f_3–f_8, f_{13}, f_{15}–f_{17}, and f_{30}. All algorithms shared the best performance for functions f_{23}–f_{25}. QR-OMRFO shared the best performance with St- and S-OMRFOs for functions f_{26} and f_{29}, respectively. Taking into account both shared and unshared best performances, QR-OMRFO led the number followed by St-OMRFO, MRFO, and S- and Q-OMRFOs which attained 17, 15, 5, 4, and 3 scores, respectively. Considering both worst

and best achievements, the results indicate that the proposed OMRFOs outperformed MRFO on the consistency performance for 50D of CEC14 problems.

The last row of Table 6 shows the summary of the performance test on 50D of the 30 functions. It shows that St-, Q-, QR-, and S-OMRFOs outperformed MRFO for 27, 14, 27, 16 functions, respectively, and have worse performances than MRFO for 3, 4, 3, 3 functions, respectively. St- and QR-OMRFOs have no equal performances with MRFO while Q- and S-OMRFOs have equal performances with MRFO for 12 and 11 functions, respectively. The results of both the Wilcoxon and Friedman Tests based on the data presented in Table 6 are shown in Table 7. It shows that all the two-tailed, p values of the Wilcoxon sign rank test are less than 0.05. This indicates that the improvement made by the proposed OMRFOs over the MRFO algorithm is significant. On the other hand, the Friedman test shows that St-OMRFO achieved the smallest mean, which is the best rank followed by QR-, S-, Q-OMRFOs, and MRFO with the mean rank as 1.67, 1.70, 3.62, 3.77, and 4.25, respectively.

Application to optimize fuzzy control for an inverted pendulum system

Experimental setup and parameter setting. The proposed OMRFOs were applied to optimize the fuzzy model to control the cart's position and pendulum's angle of an inverted pendulum system. The parameters used for the work such as maximum number of function evaluation, NFE_{max}, maximum iteration, $Iter_{max}$, and searching agents, m were defined as 50,000, 1000, and 50, respectively. A total of 25 independent runs were conducted to generate 25 different data on the fitness cost attainment of expression shown in (13). The performance analysis was statistically conducted via Friedman and Wilcoxon sign rank tests. Their evaluation criteria are similar as those stated in the benchmark functions.

Performance comparison on result and discussion. The generated data and their corresponding mean and SD values are shown in Table 8. The best fitness cost among the five algorithms is highlighted in bold font. The result shows that MRFO, St-, Q-, QR-, and S-OMRFOs achieved 1, 3, 5, 4, and 12 best runs, respectively, out of the 25 runs. Comparing the fitness cost of all 25 runs on a single algorithm, the result shows that MRFO, St-, Q-, QR-, and S-OMRFOs achieved the best cost on the 25th, 18th, 21st, 7th, and 10th runs, respectively. Among the algorithms, S-OMRFO led the best accuracy performance followed by QR-, Q-, and St-OMRFOs and MRFO which attained the best fitness cost of 2.116, 2.280, 2.289, 2.289, and 2.293, respectively. The Mean result shows that S-OMRFO achieved the best value, followed by Q-, St-, and QR-OMRFOs and MRFO which scored 2.292, 2.319, 2.322, 2.323, and 2.333, respectively. On the other hand, the SD result indicates a consistent performance of the algorithms in generating a solution. It shows that St- and S-OMRFOs attained the best and the worst consistent performance, respectively. MRFO and QR-OMRFO attained an equal score on the consistency performance. Figure 11 shows the convergence curves of the contested algorithms for the first 100 iterations. The convergence curves are generated based on the average value of fitness cost of the 25 runs. The plots are in-line with the Mean result shown in Table 8. All plots show a fast convergence from the beginning until the 15th iteration. They converge slowly towards the optimal solution until the end of operation.

The result presented in Table 8 was further analyzed via the Friedman and Wilcoxon sign rank tests with a 95% confidence interval setup. The results of both tests are shown in Table 9. It shows that all two-tailed, p resulted from the Wilcoxon sign test comparing MRFO with St-, Q- and S-OMRFO have values less than 0.05 or 5% interval. It indicates that the improvement made by the proposed OMRFOs over the MRFO is significant. The most and the least significant differences in the accuracy improvement over the MRFO are shown by S-OMRFO and St-OMRFOs, respectively. However, the table shows the two-tailed, p value as a result from comparing MRFO with QR-OMRFO, is 0.083 or greater than 0.05. It indicates that the improvement made by QR-OMRFO is not significant. In other words, both algorithms have equal accuracy performance. On the other hand, the Friedman test shows the rank acquired by the algorithms. S-OMRFO attained the best rank, followed by Q-, St-, and QR-OMRFOs and MRFO with the scores 2.20, 2.64, 3.08, 3.08, and 4.00, respectively. St- and QR-OMRFOs shared the third rank with the score 3.08. Hence, from these evidences, utilization of opposition-based learning mechanism into MRFO have significantly improved the original MRFO accuracy performance in searching an optimal solution of a nonlinear fuzzy logic model to control an inverted pendulum system.

Figure 12 shows the output response of the cart's position given a step input function as a test signal. The cart is required to move to the final position at 10 cm from its initial position, 0 cm at the center of the horizontal axis. In general, it shows that all the optimized fuzzy controllers had successfully controlled the cart to move to the desired 10 cm position. However, there is a little offset from the desired 10 cm position shown by the graphs, indicating that the cart did not settle at exactly the 10 cm position. The response is clearly observed from the zoomed-in picture. All the graphs show that the cart initially moved beyond the desired position. It then moved back and forth near the 10 cm location before it finally settled at the final location.

Table 6. Results of the Friedman and Wilcoxon tests on 50D of CEC14.

	MRFO	St-OMRFO	Q-OMRFO	QR-OMRFO	SO-MRFO
Function	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)
1	1.90E + 07 (6.83E + 06)	3.42E + 06 (1.36E + 06)	5.45E + 06 (1.67E + 06)	3.54E + 06 (1.94E + 06)	5.46E + 06 (2.38E + 06)
2	4.42E + 06 (4.25E + 06)	3.51E + 03 (6.11E + 03)	7.48E + 03 (7.49E + 03)	2.69E + 03 (6.03E + 03)	7.35E + 03 (7.40E + 03)
3	9.69E + 04 (1.79E + 04)	2.53E + 04 (6.79E + 03)	4.08E + 04 (9.48E + 03)	2.32E + 04 (5.76E + 03)	3.85E + 04 (8.04E + 03)
4	6.49E + 02 (6.64E + 01)	3.22E + 02 (3.17E + 01)	5.40E + 02 (3.53E + 01)	3.29E + 02 (3.91E + 01)	5.41E + 02 (4.36E + 01)
5	5.21E + 02 (3.62E-02)	3.54E + 02 (4.03E-02)	5.21E + 02 (4.52E-02)	3.54E + 02 (3.03E-02)	5.21E + 02 (3.40E-02)
6	6.43E + 02 (5.14E + 00)	4.24E + 02 (5.09E + 00)	6.31E + 02 (5.82E + 00)	4.26E + 02 (4.94E + 00)	6.33E + 02 (5.65E + 00)
7	7.01E + 02 (1.36E-01)	5.10E + 02 (1.07E-02)	7.00E + 02 (1.37E-02)	5.10E + 02 (9.30E-03)	7.00E + 02 (1.19E-02)
8	1.05E + 03 (2.85E + 01)	6.04E + 02 (2.42E + 01)	1.01E + 03 (3.34E + 01)	6.10E + 02 (2.17E + 01)	1.02E + 03 (2.99E + 01)
9	1.23E + 03 (5.20E + 01)	6.41E + 02 (4.77E + 01)	1.12E + 03 (5.66E + 01)	6.35E + 02 (4.06E + 01)	1.19E + 03 (4.81E + 01)
10	5.69E + 03 (8.31E + 02)	4.66E + 03 (7.76E + 02)	5.45E + 03 (9.27E + 02)	4.08E + 03 (9.29E + 02)	5.29E + 03 (1.11E + 03)
11	8.29E + 03 (1.34E + 03)	9.07E + 03 (1.15E + 03)	1.38E + 04 (1.33E + 03)	9.20E + 03 (1.66E + 03)	1.20E + 04 (1.92E + 03)
12	1.20E + 03 (8.43E-01)	7.29E + 02 (2.91E-01)	1.20E + 03 (3.42E-01)	7.29E + 02 (3.20E-01)	1.20E + 03 (3.75E-01)
13	1.30E + 03 (1.10E-01)	7.94E + 02 (7.90E-02)	1.30E + 03 (1.02E-01)	7.94E + 02 (7.55E-02)	1.30E + 03 (9.74E-02)
14	1.40E + 03 (1.87E-01)	9.90E + 02 (8.78E-02)	1.40E + 03 (1.04E-01)	9.90E + 02 (9.71E-02)	1.40E + 03 (1.15E-01)
15	1.56E + 03 (1.13E + 01)	1.06E + 03 (9.94E + 00)	1.54E + 03 (1.32E + 01)	1.06E + 03 (7.53E + 00)	1.54E + 03 (1.00E + 01)
16	1.62E + 03 (5.90E-01)	1.03E + 03 (3.07E-01)	1.62E + 03 (3.87E-01)	1.03E + 03 (2.95E-01)	1.62E + 03 (3.72E-01)
17	2.70E + 06 (1.43E + 06)	5.01E + 05 (2.81E + 05)	6.30E + 05 (3.76E + 05)	5.09E + 05 (2.76E + 05)	6.73E + 05 (3.70E + 05)
18	3.39E + 03 (1.31E + 03)	2.15E + 03 (8.90E + 02)	2.99E + 03 (1.03E + 03)	2.09E + 03 (9.68E + 02)	3.16E + 03 (1.12E + 03)
19	1.95E + 03 (2.86E + 01)	1.31E + 03 (2.34E + 01)	1.96E + 03 (2.65E + 01)	1.30E + 03 (2.62E + 01)	1.95E + 03 (2.96E + 01)
20	3.84E + 04 (1.51E + 04)	8.18E + 03 (3.15E + 03)	1.30E + 04 (4.28E + 03)	8.68E + 03 (3.50E + 03)	1.41E + 04 (4.75E + 03)
21	1.10E + 06 (7.26E + 05)	2.84E + 05 (1.56E + 05)	3.96E + 05 (1.96E + 05)	2.49E + 05 (1.78E + 05)	3.93E + 05 (2.23E + 05)
22	3.40E + 03 (3.17E + 02)	2.04E + 03 (2.43E + 02)	3.20E + 03 (2.77E + 02)	2.03E + 03 (3.04E + 02)	3.20E + 03 (3.47E + 02)
23	2.50E + 03 (0.00E + 00)	1.74E + 03 (0.00E + 00)	2.50E + 03 (0.00E + 00)	1.74E + 03 (0.00E + 00)	2.50E + 03 0.00E + 00
24	2.60E + 03 (0.00E + 00)	1.93E + 03 (0.00E + 00)	2.60E + 03 (0.00E + 00)	1.93E + 03 (0.00E + 00)	2.60E + 03 (0.00E + 00)
25	2.70E + 03 (0.00E + 00)	1.43E + 03 (0.00E + 00)	2.70E + 03 (0.00E + 00)	1.43E + 03 (0.00E + 00)	2.70E + 03 (0.00E + 00)
26	2.77E + 03 (4.59E + 01)	1.66E + 03 (3.60E + 01)	2.76E + 03 (4.85E + 01)	1.67E + 03 (3.60E + 01)	2.76E + 03 (4.85E + 01)

(continued)

Table 6. (continued)

	MRFO	St-OMRFO	Q-OMRFO	QR-OMRFO	SO-MRFO
Function	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)	Mean (STD)
27	2.90E + 03 (1.36E-12)	2.27E + 03 (2.62E + 02)	3.71E + 03 (2.98E + 02)	2.26E + 03 (2.32E + 02)	3.74E + 03 (2.64E + 02)
28	3.00E + 03 (1.36E-12)	3.69E + 03 (8.93E + 02)	5.15E + 03 (1.24E + 03)	3.92E + 03 (1.13E + 03)	5.10E + 03 (1.57 + 03)
29	3.14E + 03 (1.91E + 02)	3.16E + 03 (4.18E + 02)	3.35E + 03 (5.88E + 02)	4.27E + 03 (0.00E + 00)	3.10E + 03 (0.00E + 00)
30	3.43E + 04 (2.72E + 04)	1.17E + 04 (7.22E + 03)	1.49E + 04 (9.19E + 03)	1.14E + 04 (5.69E + 03)	7.56E + 03 (7.24 E + 03)
	+ /–/ =	27/3/0	14/4/12	27/3/0	16/3/11

Table 7. Results of the Friedman and Wilcoxon tests on 50D of CEC14.

	Friedman test			Wilcoxon signed rank test					
Algorithm	Mean rank	ρ	χ^2	MRFO – St-OMRFO			MRFO – Q-OMRFO		
MRFO	4.25	0.00	78.098	R^+	R^-	ρ	R^+	R^-	ρ
				3	27	0.000	5	17	0.012
St-MRFO	1.67			MRFO–QROMRFO			MRFO – S-OMRFO		
Q-OMRFO	3.77			R^+	R^-	ρ	R^+	R^-	ρ
QR-MRFO	1.70			3	27	0.000	3	18	0.010
S-OMRFO	3.62								

The time-domain analysis of the fuzzy logic controller performance on controlling the cart's position is shown in Table 10. It comprises of an analysis on the rise time, t_r settling time, t_s steady state-error, e_{SS} and percentage overshoot, $\%OS$. Noted that Q-OMRFO achieved the fastest rise time followed by MRFO, S-, QR-, and St-OMRFOs with the scores 0.981, 0.983, 0.991, 0.995, and 0.999 s, respectively. A faster response is commonly associated with a higher overshoot. This is evidenced from the result that MRFO has the highest percentage overshoot at 7.2 %, followed by Q-, St-, QR-, and S-OMRFOs with the scores 4.3 %, 3.6 %, 2.8 %, and 0.4 %, respectively. The percentage overshoot achieved by the S-OMRFO is significantly less compared to all other algorithms. A higher percentage overshoot is commonly associated with a slower settling time. This is evidenced from the result that Q-OMRFO attained the worst and slowest settling time at 2.154 s while S-OMRFO attained the fastest settling time at 1.295 s. QR- and St-OMRFOs and MRFO attained the second, third, and fourth places, respectively, on the settling time performance. Noted that on the steady-state error attainment, S-OMRFO and MRFO attained the smallest and largest errors. Based on the MRFO-IT2FLC scheme, the cart drifted away about 0.32 cm from the desired 10 cm and it drifted away about 0.004 cm based on the S-OMRFO-IT2FLC control scheme. St- and QR-OMRFO shared the same steady-state error about 0.01 cm, resulting in the second place, while Q-OMRFO attained the third place with a steady-state error about 0.038 cm from the desired location. On the overall performance of the cart response, the result shows that S-OMRFO achieved the best performance among the algorithms. It achieved the best percentage overshoot, settling time, and rise time. On the contrary, MRFO attained the worst performance among the algorithms. It attained the worst percentage overshoot and steady-state error. QR-OMRFO attained neither the best nor worst performance, Q-OMRFO attained 1-best and 1-worst performances, while St-OMRFO attained 1-worst performance giving them the second, third, and fourth places, respectively.

Figure 13 shows the output response of the pendulum's angle based on a step input function applied on the cart. The pendulum is attached on the cart's body and it has to be in a vertically upright position while in operation. In that particular condition, the corresponding angle of the pendulum is 0°.

All graphs from the figure generally show that the pendulum finally settled at the desired 0° angle. It indicates the optimized fuzzy controller had successfully stabilized the pendulum and maintained its vertical upright position. In the first

Table 8. Attainment of fitness cost, mean, and standard deviation.

Runs	MRFO	St-OMRFO	Q-OMRFO	QROMRFO	S-OMRFO
1	2.341E + 00	2.340E + 00	2.328E + 00	2.324E + 00	2.315E + 00
2	2.340E + 00	2.327E + 00	2.326E + 00	2.317E + 00	2.265E + 00
3	2.345E + 00	2.324E + 00	2.338E + 00	2.322E + 00	2.308E + 00
4	2.334E + 00	2.308E + 00	2.307E + 00	2.334E + 00	2.116E + 00
5	2.308E + 00	2.310E + 00	2.319E + 00	2.331E + 00	2.283E + 00
6	2.317E + 00	2.333E + 00	2.308E + 00	2.306E + 00	2.308E + 00
7	2.356E + 00	2.301E + 00	2.340E + 00	2.277E + 00	2.303E + 00
8	2.338E + 00	2.322E + 00	2.330E + 00	2.343E + 00	2.327E + 00
9	2.335E + 00	2.333E + 00	2.332E + 00	2.325E + 00	2.309E + 00
10	2.316E + 00	2.333E + 00	2.312E + 00	2.324E + 00	2.108E + 00
11	2.323E + 00	2.312E + 00	2.311E + 00	2.324E + 00	2.186E + 00
12	2.342E + 00	2.315E + 00	2.318E + 00	2.289E + 00	2.313E + 00
13	2.345E + 00	2.317E + 00	2.312E + 00	2.314E + 00	2.317E + 00
14	2.339E + 00	2.337E + 00	2.319E + 00	2.325E + 00	2.340E + 00
15	2.331E + 00	2.313E + 00	2.319E + 00	2.316E + 00	2.307E + 00
16	2.340E + 00	2.331E + 00	2.304E + 00	2.333E + 00	2.320E + 00
17	2.364E + 00	2.323E + 00	2.333E + 00	2.364E + 00	2.315E + 00
18	2.350E + 00	2.289E + 00	2.311E + 00	2.316E + 00	2.365E + 00
19	2.323E + 00	2.313E + 00	2.297E + 00	2.335E + 00	2.313E + 00
20	2.322E + 00	2.315E + 00	2.371E + 00	2.305E + 00	2.330E + 00
21	2.368E + 00	2.352E + 00	2.280E + 00	2.315E + 00	2.292E + 00
22	2.313E + 00	2.353E + 00	2.316E + 00	2.340E + 00	2.313E + 00
23	2.321E + 00	2.313E + 00	2.342E + 00	2.337E + 00	2.355E + 00
24	2.330E + 00	2.327E + 00	2.312E + 00	2.326E + 00	2.244E + 00
25	2.293E + 00	2.315E + 00	2.301E + 00	2.332E + 00	2.357E + 00
Mean	2.333	2.322	2.319	2.323	**2.292**
STD	0.017	0.014	0.018	0.017	0.064

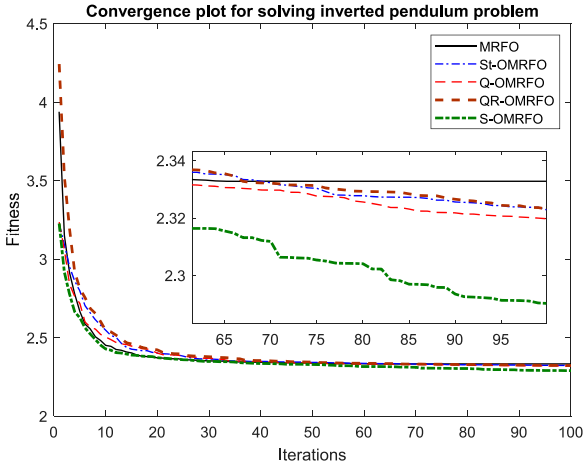

Figure 11. Convergence curves produced by the contested algorithms.

12 s, the pendulum experienced a series of significant swings. This is due to its reaction on the initial cart's movement towards the desired 10 cm location when a step input is applied. The swing continuously occurred as the cart moved iteratively in back and forth directions in order to achieve its desired 0° location. The swing significantly reduced over time before finally settling down to the final location showing that the controller had worked sufficiently well. The graphs show

Table 9. Results of the Friedman and Wilcoxon tests on inverted pendulum.

	Friedman test			Wilcoxon signed rank test					
				MRFO–St-MRFO			MRFO – Q-OMRFO		
Algorithm	Mean rank	ρ	χ^2	R^+	R^-	ρ	R^+	R^-	ρ
MRFO	4.00	0.001	17.82	256	69	0.012	271	54	0.004
St-OMRFO	3.08			MRFO – QR-OMRFO			MRFO – QR-OMRFO		
Q-OMRFO	2.64			R^+	R^-	ρ	R^+	R^-	ρ
QR-MRFO	3.08			227	98	0.083	279	46	0.002
S-OMRFO	2.20								

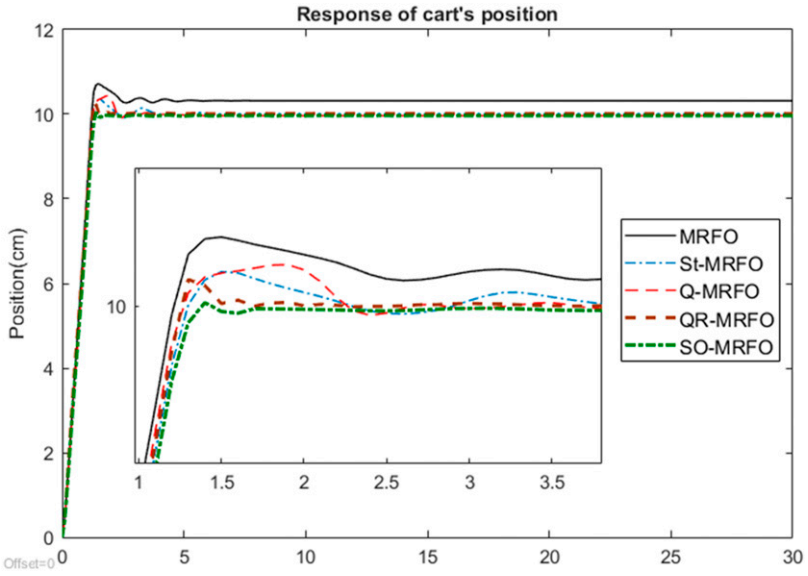

Figure 12. Response of cart's position.

that the pendulum oscillation pattern in general is almost the same. However, from the zoomed-in picture, the pendulum responses vary from one to another.

The time-domain analysis of the pendulum's response is shown in Table 11. It has an extra time-domain performance specification compared to the cart where its components include rise time, t_r, settling time, t_s, steady state-error, e_{SS}, maximum overshoot, OS_{max}, and maximum undershoot, US_{max}. Noted from the table that the smallest or the best overshoot was achieved by St-OMRFO with the score of 2.146° while the worst overshoot was attained by QR-OMRFO with the score of 2.500°. The second-best overshoot was attained by Q-OMRFO, followed by S-OMRFO and MRFO with the scores 2.303°, 2.352°, and 2.384°, respectively. Q -OMRFO achieved the best undershoot performance followed by St-OMRFO, MRFO, and S- and QR-OMRFO with the scores −1.872°, −1.888°, −2.223°, −2.252°, and −2.584°, respectively. The overshoot and undershoot are commonly in pairs as evidenced from the result that the worst performance for both components is attained by QR-OMRFO. In terms of settling time, the result shows that Q-OMRFO and QR-OMRFO achieved the fastest and slowest settling time, respectively, with the scores of 4.619 and 14.770 s. The smallest undershoot performance achieved by Q-OMRFO contributed to its achievement in settling time as these two components are related to each other. This is also evidenced from the worst performance in both settling time and undershoot components shown by QR-OMRFO. In terms of rise time, the table shows that St-OMRFO achieved the fastest time with the score of 0.249 s. As evidenced from the result, the best rise time performance is due to its largest overshoot attainment. MRFO and QR-OMRFO shared the second fastest rise time performance with the score of 0.252 s. Q- and S-OMRFO attained the third and last performances, respectively, with the scores of 0.254 and 0.261 s. For the steady-state error performance, all algorithms achieved almost zero error. Q-OMRFO led the performance with the smallest error, followed by MRFO, St-, S-, and QR-OMRFOs.

Table 10. Time-domain performance of the cart's response.

Algorithm	MRFO	St-OMRFO	Q-OMRFO	QR-OMRFO	S-OMRFO
Settling time, t_s (sec)	2.033	1.842	2.154	1.405	1.295
Rise time, t_r (sec)	0.983	0.999	0.981	0.995	0.991
Percent overshoot, $\%OS$	7.200	3.600	4.300	2.800	0.400
Steady state error, e_{SS}	0.320	0.010	0.038	0.010	0.004

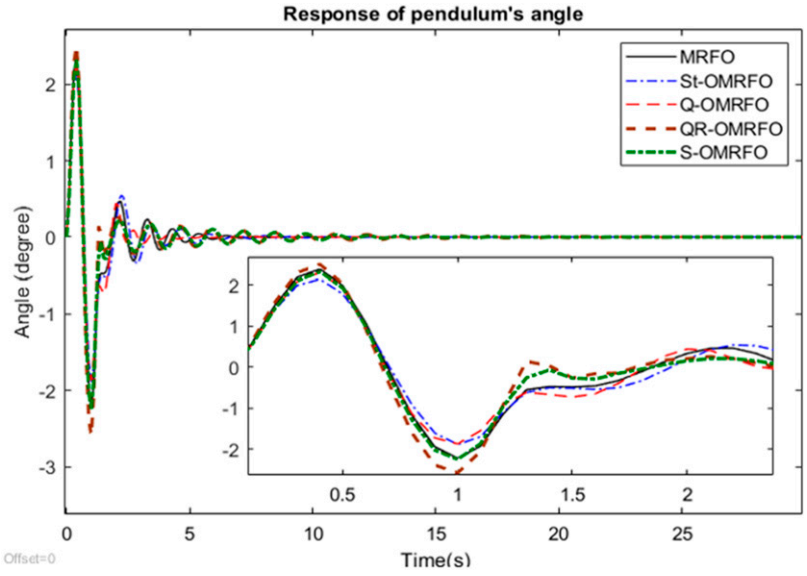

Figure 13. Response of pendulum's angle.

Table 11. Time-domain performance of the pendulum's response.

Algorithm	MRFO	St-OMRFO	Q-OMRFO	QR-OMRFO	S-OMRFO
Settling time, t_s (sec)	6.066	7.542	4.619	14.770	11.720
Rise time, t_r (sec)	0.252	0.249	0.254	0.252	0.261
Max. overshoot,	2.384	2.146	2.303	2.500	2.352
Max. undershoot,	−2.223	−1.888	−1.872	−2.584	−2.252
Steady-state error, e_{SS}	3.72E-9	4.40E-6	3.16E-10	1.30E-3	3.92E-5

For the overall performance on the application of the proposed algorithms to solve the real-world problem, it shows that Q-OMRFO achieved the best performance among other algorithms for the undershoot, settling time, and steady state-error, while St-OMRFO achieved the best performance for overshoot and rise time components. On the contrary, QR-OMRFO attained the worst performance among the algorithms for the overshoot, undershoot, settling time, and steady-state error, while S-OMRFO attained the worst performance for the rise time component. MRFO achieved neither best nor worst performances on the five components for the pendulum response. Based on these performances, the best algorithm for optimizing the pendulum's response was achieved by Q-OMRFO, followed by St-OMRFO, MRFO, and S- and QR-OMRFOs.

Conclusion

The Opposition-based Manta Ray Foraging Optimization (OMRFO) algorithm has been presented in this paper. MRFO comprises a combination of random and deterministic spiral models in its Chain, Cyclone, and Somersault phases. The

algorithm suffers insufficient exploration of search space which has led to a premature convergence and low accuracy performance. As a solution to the problem, an Opposition-based Learning (OBL) mechanism was incorporated into the original structure of the MRFO. The opposition scheme expanded the search area of every single Manta Ray agent. It offered the agents to explore the opposite region of its current location, which is unable to be reached by the conventional strategy of MRFO. Four variants of the opposition schemes had been adopted, which led to Standard-opposition (St-), Quasi-opposition (Q-), Quasi-Reflected Opposition (QR-), and Super-Opposition (S-) OMRFOs. The schemes utilized different strategies in the way they determined the opposite location of each agent. The accuracy performance of the proposed algorithms had been tested on 30 unconstrained problems adopted from the IEEE Evolutionary Computation, CEC14 test suite. A 50-dimension had been tested on each problem to comprehensively investigate the effectiveness of the proposed OMRFOs over MRFO for different problems. The algorithms also had been applied to solve a complex real-world problem in intelligent control system engineering. It was utilized to optimize parameters of 26 dimensions nonlinear fuzzy logic model to control the cart and pendulum positions of an inverted pendulum system. A total of 25 independent tests had been conducted on the inverted pendulum problem to acquire a set of data to perform a statistical analysis on the accuracy achievement. The result of the performance test of the proposed OMRFO algorithms on the 50-dimension of the CEC14 benchmark functions showed that all the proposed OMRFOs outperformed MRFO. This is evidenced from the statistical analysis on the Friedman test showing all proposed algorithms have acquired better rank. St-OMRFO achieved the best rank for the 50-dimension problems of the CEC14. The proposed OMRFOs also significantly improved the accuracy performance in searching for a theoretical optimum solution of the benchmark problems. This is evidenced from the statistical analysis on the Wilcoxon sign rank test showing the two-tailed result with 95% confidence that is lower than 5%. The statistical analysis on the result of the inverted pendulum system showed that S-OMRFO achieved the best mean accuracy. The result of the Friedman test showed that the final rank from the best to the worst orders is S-, Q-, St-, and QR-OMRFOs and MRFO. It is also evidenced from the Wilcoxon sign rank test on the result of the inverted pendulum that the two-tailed value for S-, St-, and Q-OMRFOs is less than 0.05, indicating the improvement over MRFO is significant. All the algorithms had satisfactorily controlled both cart and pendulum positions of the inverted pendulum system. The cart rapidly settled to its desired final location in less than 1 s. The pendulum oscillation due to its reaction to move the cart to the 10 cm location had been quickly attenuated and removed. Last but not least, as the proposed algorithms are not problem-specific algorithms, they might be considered as potential tools for solving other complex and nonlinear behavior of real-world problems.

Acknowledgments

The authors would like to thank the Ministry of Higher Education Malaysia for providing financial support under Fundamental Research Grant Schemes (FRGS) No. FRGS/1/2021/ICT02/UMP/03/2 (University reference RDU210116) and FRGS/1/2021/ICT02/UMP/02/2 (University reference RDU210110).

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Fundamental Research Grant Schemes (FRGS) No. FRGS/1/2021/ICT02/UMP/03/2 (University reference RDU210116) and FRGS/1/2021/ICT02/UMP/02/2 (University reference RDU210110).

ORCID iD

Nor Maniha Abdul Ghani https://orcid.org/0000-0002-4133-5558

References

1. Hussain K, Mohd Salleh MN, Cheng S, et al. Metaheuristic research: a comprehensive survey. *Artif Intell Rev* 2019; 52(4): 2191–2233.
2. Di Trapani F, Antonio Pio Sberna, Giuseppe Carlo Marano. A genetic algorithm-based framework for seismic retrofitting cost and expected annual loss optimization of non-conforming reinforced concrete frame structures. *Comput Struct* 2022; 271: 106855.
3. Kennedy J and Eberhart R. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995; 1942–1948.

4. Ahmad Nor Kasruddin Nasir, Mohd Ashraf AhmadOsman Tokhi M. Hybrid spiral-bacterial foraging algorithm for a fuzzy control design of a flexible manipulator. *J Low Freq Noise Vib Act Control* 2022 41(1): 3402–3581.
5. KhisheMosavi MR. Chimp optimization algorithm. *Expert Syst Appl* 2020; 149: 113338.
6. Kaidi W, Khishe M, and Mohammadi M. Dynamic levy flight chimp optimization. *Knowl Base Syst* 2022; 235: 107625.
7. Osaba E. A Tutorial on the design, experimentation and application of metaheuristic algorithms to real-World optimization problems. *Swarm Evol Comput* 2021; 64: 100888.
8. Zhang N, Zhao D, Shi J, et al. Characterizing and predicting bluff-body solid fuel ramjet performances via shape design and multi-objective optimization model. *Phys Fluids* 2023; 35: 125150.
9. Saffari A and Khishe M, Seyed-Hamid Zahiri. Fuzzy-ChOA: an improved chimp optimization algorithm for marine mammal classification using artificial neural network. *Analog Integr Circuits Signal Process* 2022; 111: 403–417.
10. Saffari A, Zahiri SH, and Khishe M. Fuzzy grasshopper optimization algorithm: a hybrid technique for tuning the control parameters of GOA using fuzzy system for big data sonar classification. *Iranian Journal of Electrical and Electronic Engineering* 2022; 01: 2131.
11. Saffari A, Zahiri SH, and Khishe M. Fuzzy Whale optimisation algorithm: a new hybrid approach for automatic sonar target recognition. *J Exp Theor Artif Intell* 2023; 35(2): 309–325.
12. Guo Y, Khishe M, Mohammadi M, Mojtaba Shams Nateri, et al.. Evolving deep convolutional neural networks by extreme learning machine and fuzzy Slime Mould optimizer for real-time sonar image recognition. *Int J Fuzzy Syst* 2022; 24: 1371–1389.
13. Dhiman G and Kumar V. Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. *Knowl Base Syst* 2019; 165: 169–196.
14. Mirjalili S and Lewis A. The whale optimization algorithm. *Adv Eng Software* 2016; 95: 51–67.
15. Sun Y and Chen Y. Multi-population improved whale optimization algorithm for high dimensional optimization. *Appl Soft Comput* 2021; 112: 107854.
16. Mirjalili S. The ant lion optimizer. *Adv Eng Software* 2015; 83: 80–98.
17. Yazdani M and Jolai F. Lion Optimization Algorithm (LOA): a nature-inspired metaheuristic algorithm. *J Comput Des Eng* 2016; 3(1): 24–36.
18. Balochian S and Baloochian H. Social mimic optimization algorithm and engineering applications. *Expert Syst Appl* 2019; 134: 178–191.
19. Rakhshani H and Rahati A. Snap-drift cuckoo search: a novel cuckoo search optimization algorithm. *Appl Soft Comput* 2017; 52: 771–794.
20. She B, Fournier A, Yao M, et al. A self-adaptive and gradient-based cuckoo search algorithm for global optimization. *Appl Soft Comput* 2022; 122: 108774.
21. Zhao W, Zhang Z, and Wang L. Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications. *Eng Appl Artif Intell* 2020; 87: 103300.
22. Cabanillas-Torpoco M. First description of a giant manta ray fetus Mobula birostris (Walbaum 1792) from Tumbes, Peru (Southeast Pacific). *Zootaxa* 2019; 4603(2): 397.
23. Razak AAA, Nasir ANK, Ghani NMA, et al. Non-dominated sorting manta ray foraging algorithm with an application to optimize PD control. *Recent Trends in Mechatronics Towards Industry* 2022; 40: 463–474.
24. Zhang Y. Backtracking search algorithm with specular reflection learning for global optimization. *Knowl Base Syst* 2021; 212: 106546.
25. Zhang J, Huang JX, and Hu QV. Global optimization with one-class classification-assisted selection. *Swarm Evol Comput* 2021; 60: 100801.
26. Zhu F, Wang W, and Li S. Application of improved manta ray foraging optimization algorithm in coverage optimization of wireless sensor networks. *Comput Intell Neurosci* 2022; 2022: 3082933.
27. Rizk-Allah RM, Zineldin MI, Mousa AAA, et al. Novel hybrid manta ray foraging optimizer and its application on parameters estimation of lithium-ion battery. *Int J Comput Intell Syst*; 15(1): 1–22.
28. Jusof MFM, Nasir ANK, Razak AAA, et al. Adaptive-somersault MRFO for global optimization with an application to optimize PD control. Proceedings of the 12th National Technical Seminar on Unmanned System Technology 2020: 1027–1039.
29. Liu G and An R. Applying a yin–yang perspective to the theory of paradox: a review of Chinese management. *Psychol Res Behav Manag* 2021; 14: 1591.
30. Alamri HS, Zamli KZ, and Pmt. Opposition-based learning technique for enhancing meta-heuristic performance. *IEEE Access* 2019; 7: 2925088.
31. Rahnamayan S, Wang GG, and Ventresca M. An intuitive distance-based explanation of opposition-based sampling. *Applied Soft Computing Journal* 2012; 12(9): 034.
32. Katoch S, Chauhan SS, and Kumar V. A review on genetic algorithm: past, present, and future. *Multimed Tool Appl* 2021; 80(5): 6.

33. Jusof MFM, Mohammad S, Razak AAA, et al. Hybrid manta ray foraging-particle swarm algorithm for PD control optimization of an inverted pendulum. *Recent Trends in Mechatronics Towards Industry* 2022; 40: 1–13.
34. Eltaeib T and Mahmood A. Differential evolution: a survey and analysis. *Appl Sci* 2018; 8(10): 8101945.
35. Baioletti M, di Bari G, Milani A, et al. Differential evolution for neural networks optimization. *Mathematics* 2020; 8(1): 8010069.
36. Bo Q, Cheng W, and Khishe M. Evolving chimp optimization algorithm by weighted opposition-based technique and greedy search for multimodal engineering problems. *Appl Soft Comput* 2023; 132: 109869.
37. Khishe M. Greedy opposition-based learning for chimp optimization algorithm. *Artif Intell Rev* 2023; 56: 7633–7663.
38. Yang Y, Cui Z, Wu J, et al. Fuzzy c-means clustering and opposition-based reinforcement learning for traffic congestion identification. *J Inf Comput Sci* 2012; 9(9): 2441–2450.
39. Sahba F, Tizhoosh HR, and Salama MMMA. Application of opposition-based reinforcement learning in image segmentation. Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing, Honolulu, HI, USA, 01-05 April 2007.
40. Mousavirad SJ, Schaefer G, Korovin I, et al. A region-based differential evolution algorithm incorporation opposition-based learning for optimising the learning process of multi-layer neural networks. *Lect Notes Comput Sci* 2021; 12694: 26.
41. Boubaker O. The inverted pendulum benchmark in nonlinear control theory: a survey. *Int J Adv Rob Syst* 2013; 10: 55058.
42. Huang SJ and lo Huang C. Control of an inverted pendulum using grey prediction model. *IEEE Trans Ind Appl* 2000; 36(2): 833761.
43. Prasad LB, Tyagi B, and Gupta HO. Optimal control of nonlinear inverted pendulum system using PID controller and LQR: performance analysis without and with disturbance input. *Int J Autom Comput* 2014; 11(6): 11633.
44. Zhao D, Lu Z, He Z, et al. A review of active control approaches in stabilizing combustion systems in aerospace industry. *Prog Aero Sci* 2018; 97: 35–60.
45. El-Bardini M and El-Nagar AM. Interval type-2 fuzzy PID controller for uncertain nonlinear inverted pendulum system. *ISA Trans* 2014; 53(3): 007.
46. Torshizi AD, Zarandi MHF, and Zakeri H. On type-reduction of type-2 fuzzy sets: a review. *Applied Soft Computing Journal* 2015; 27: 031.
47. Jamin F, Ghani NMA, Ibrahim Z, et al. Stabilizing control of two-wheeled wheelchair with movable payload using optimized interval type-2 fuzzy logic. *J Low Freq Noise Vib Act Control* 2022; 40(3): 1585–1606.
48. Omatu S, Kishida Y, and Yoshioka M. Neuro-control for single-input multi-output systems. *International Conference on Knowledge-Based Intelligent Electronic Systems, Proceedings, KES* 1998; 1: 725847.
49. İlgen S, Oflaz E, Gülbahçe E, et al. Modelling and control of a single-wheel inverted pendulum by using Adams and Matlab. *International Journal of Applied Mathematics, Electronics and Computers* 2016; 4: 326–328.
50. Messikh L, Guechi EH, and Blazic S. Stabilization of the cart–inverted-pendulum system using state-feedback pole-independent MPC controllers. *Sensors* 2022; 22(1): 243.
51. Jusof MFM, Mohammad S, Razak AAA, et al. Hybrid manta ray foraging-particle swarm algorithm for PD control optimization of an inverted pendulum. *Lecture Notes in Electrical Engineering* 2022; 730: 1007.
52. Razak AAA, Nasir ANK, Ghani NMA, et al. Hybrid genetic manta ray foraging optimization and its application to interval type 2 fuzzy logic control of an inverted pendulum system. *IOP Conf Ser Mater Sci Eng* 2020; 917(1): 012082.
53. Mendel JM and John RI. A fundamental decomposition of type-2 fuzzy sets. *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS* 2001; 4: 944356.
54. Nobile MS, Cazzaniga P, Besozzi D, et al. Fuzzy Self-Tuning PSO: a settings-free algorithm for global optim. *Swarm Evol Comput* 2018; 39: 001.
55. Wu D and Tan WW. A type-2 fuzzy logic controller for the liquid-level process. *Fuzzy Syst Conf* 2004; 2: 1375536.
56. Nie M and Tan WW. Towards an efficient type-reduction method for interval type-2 fuzzy logic systems. 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), Hong Kong, China, 01-06 June 2008.
57. Atacak and Bay OF. A type-2 fuzzy logic controller design for buck and boost DC-DC converters. *J Intell Manuf* 2012; 23(4): 10845.
58. Nasir ANK and Razak AAA. Opposition-based spiral dynamic algorithm with an application to optimize type-2 fuzzy control for an inverted pendulum system. *Expert Syst Appl*; 195(2022): 116661.
59. Rojas-Morales N, Riff Rojas MC, and Montero Ureta E. A survey and classification of Opposition-Based Metaheuristics. *Comput Ind Eng* 2017; 110: 028.
60. Mahdavi S, Rahnamayan S, and Deb K. Opposition based learning: a literature review. *Swarm Evol Comput* 2018; 39: 010.
61. Liang BQ and Suganthan P. *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization*. Zhengzhou: Zhengzhou University
62. Harris E, Boushey C, Bruemmer B, et al. Publishing nutrition research: a review of nonparametric methods, part 3. *J Am Diet Assoc* 2008; 108(9): 426.
63. Sheskin DJ. *Handbook of parametric and nonparametric statistical procedures*. Boca Raton, FL: CRC Press: 2020: 1928.

64. Nash S. Handbook of parametric and nonparametric statistical procedures. *Technometrics* 2001; 43(3): 629.
65. Kafadar K and Sheskin DJ. Handbook of parametric and nonparametric statistical procedures. *Am Statistician* 1997; 51(4): 2685909.
66. LaVange M and Koch GG. Rank score tests. *Circulation* 2006; 114(23): 613638.
67. Jurecková J. Regression rank-scores tests against heavy tailed alternatives. *Bernoulli* 1999; 5(4): 3318695.
68. Demšar J. Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 2006; 7: 1–30.
69. Rodriguez-Fdez, Canosa A, and Mucientes M and STAC: A web platform for the comparison of algorithms using statistical tests. 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). Istanbu, 02-05 August 2015.
70. Derrac SG, DM and Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol Comput* 2011; 1(1): 002.
71. García S, Fernández A, Luengo J, et al. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput* 2009; 13(10): 0392.
72. García S, Fernández A, Luengo J, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf Sci* 2010; 180(10): 010.
73. LaTorre A, Molina D, Osaba E, et al. A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm Evol Comput* 2021; 67: 100973.
74. Razak AAA, Nasir ANK, Ghani NMA, et al. Non-dominated sorting manta ray foraging algorithm with an application to optimize PD control. *Lecture Notes in Electrical Engineering* 2022; 730: 42.
75. Razak AAA, Nasir ANK, Rizal NAM, et al. Quasi oppositional-manta ray foraging optimization and its application to PID control of a pendulum system. *Lecture Notes in Electrical Engin* 2022; 770: 69.
76. Razak AAA, Nasir ANK, Ghani NMA, et al. Manta ray foraging optimization with quasi-reflected opposition strategy for global optimization. *Lecture Notes in Electrical Engineering* 2022; 842: 43.