Multi-Agent Finite Impulse Response Optimizer for Numerical Optimization Problems

Tasiransurini Ab Rahman

Universiti Tun Hussein Onn Malaysia <https://orcid.org/0000-0002-1246-0894>

Nor Azlina Ab. Aziz (✉ azlina.aziz@mmu.edu.my)

Multimedia University - Melaka Campus: Multimedia University <https://orcid.org/0000-0002-2119-6191>

Zuwairie Ibrahim

Universiti Malaysia Pahang

Nor Hidayati Abdul Aziz

Multimedia University - Melaka Campus: Multimedia University

Mohd Ibrahim Shapiai

Universiti Teknologi Malaysia

Jasjit S. Suri

Advance Knowledge Engineering Centre

Research article

Keywords: measurement, estimation, finite impulse response filter, local search, population-based, metaheuristic algorithm

Posted Date: December 23rd, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-129794/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Multi-Agent Finite Impulse Response Optimizer for Numerical Optimization Problems

Tasiransurini Ab Rahman ^{1,*}, Nor Azlina Ab. Aziz ^{2,*}, Zuwairie Ibrahim ³, Nor Hidayati Abdul Aziz ², Mohd Ibrahim Shapiai ⁴, and Jasjit S Suri ⁵

¹ Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400, Johor, Malaysia; surini@uthm.edu.my

² Faculty of Engineering and Technology, Multimedia University, 75450, Melaka, Malaysia; azlina.aziz@mmu.edu.my, hidayati.aziz@mmu.edu.my

³ College of Engineering, Universiti Malaysia Pahang, 26300, Pahang, Malaysia; zuwairie@ump.edu.my

⁴ Malaysia Japan International Institute of Technology, Universiti Teknologi Malaysia, 54100, Kuala Lumpur, Malaysia; md_ibrahim83@utm.my

⁵ Advance Knowledge Engineering Center, Global Biomedical Technologies, Inc., Roseville, CA, 95661, USA; jsuri@comcast.net

* Correspondence: surini@uthm.edu.my, azlina.aziz@mmu.edu.my;

Abstract: This paper investigates the potential of the ultimate iterative unbiased finite impulse response (UFIR) filter as a source of inspiration in a population-based metaheuristic algorithm. Here, a new algorithm inspired by the measurement and estimation procedures of the UFIR filter named the Multi-Agent Finite Impulse Response Optimizer (MAFIRO) for solving numerical optimization problems is proposed. MAFIRO works with a set of agents where each performs the measurement and estimation to find a solution. MAFIRO employs a random mutation of the best-so-far solution and the shrinking local neighborhood method to balance between the exploration and exploitation phases during the optimization process. Subsequently, the performance of MAFIRO is tested by solving the benchmark test suite of the IEEE Congress on Evolutionary Computation 2014. The benchmark is composed of 30 mathematical functions. The competency of MAFIRO is compared with the Particle Swarm Optimization algorithm, Genetic Algorithm, and Grey Wolf Optimizer. The results show that MAFIRO leads in 23 out of 30 functions and has the highest Friedman rank. MAFIRO performs significantly better than the other tested algorithms. Based on the findings, we show that the concept of the UFIR filter is a good inspiration for a population-based metaheuristic algorithm.

Keywords: measurement; estimation; finite impulse response filter; local search; population-based; metaheuristic algorithm

1. Introduction

Providing optimal services or products is one of the ways to attract customers in many sectors such as business, education, banking, and industry. The need to provide the optimal services or products contributes to the increment of optimization problems from time to time. The requirement to find the best either minimum or maximum result with efficient time has encouraged many researchers to explore the solutions for optimization problems.

Commonly, optimization problems can be solved efficiently by applying suitable optimization methods. From the computer science and engineering points of view, optimization methods can be defined as procedures or algorithms used to solve optimization problems. The goal of the optimization algorithms (also known as optimizers) is to find an optimal variable's value that can either minimize or maximize the objective function under a given constraint.

Over the past few decades, numerous metaheuristic algorithms have been designed to provide efficient approximate solutions for optimization problems. In describing metaheuristic algorithms, several classifications have been proposed. The classifications are dependent on the characteristics and focus of study such as memory usage versus memoryless methods, the iterative versus greedy methods, one versus various neighborhood structures, and dynamic versus static objective function (1–3). Most of existing metaheuristic algorithms in the literature are nature-inspired (4,5), which led Fister *et al.* (6) to present a new view of metaheuristic algorithms' classification based on the inspirational source. Fister *et al.* classified metaheuristic algorithms into four main

classifications: swarm intelligence-based, bio-inspired non-swarm intelligence-based, physics or chemistry-based, and others. The non-nature-inspired algorithms are classified under others category. The nature-inspired algorithms transform a specific process in nature as an effective way of solving optimization problems. On the contrary, non-nature-inspired algorithms mimic a specific process other than nature for solving the optimization problems.

Different from Fister *et al.*, Mirjalili (7–9) and Mandal (10) classified existing metaheuristic algorithms into three main classifications: evolutionary algorithms (EAs), physics-based algorithms, and swarm intelligence (SI) algorithms. EAs and SI algorithms are the two main streams of nature-inspired algorithms (11). EAs are inspired by the evolution of nature concept (12). Meanwhile, SI algorithms are inspired by the concept of the collective or swarming behavior of creatures (9). In SI algorithms, each agent performs a series of certain operations and shares their information among others (13). SI algorithms have attracted much attention from the research community, as most of the proposed SI algorithms in the literature are bio-inspired (4,5). The Grey Wolf Optimizer (GWO) (12), Salp Swarm Algorithm (SSA) (14), Grasshopper Optimization Algorithm (GOA) (15), Spotted Hyena Optimizer (SHO) (16), Pity Beetle Algorithm (PBA) (17), and Sailfish Optimizer (18) are among modern SI algorithms.

As there are no clear guidelines in classifying metaheuristic algorithms, several researchers have created an alternative classification method for categorizing metaheuristic algorithms, especially those with characteristics different from EAs and SI. Besides biology-based, chemistry-based, and physics-based, the plant intelligence-based algorithms (19) were also proposed under the classification of nature-inspired-based algorithms. Several algorithms, such as the Flower Pollination Algorithm (20), the Runner Root Algorithm (21), Rooted Tree Optimization (22), and the Path Planning Algorithm (23), are classified under the plant intelligence-based algorithms. Other classifications, such as social-based, music-based, sport-based, and mathematics-based (19,24), also exist in the literature and can be classified under the non-nature-inspired algorithms.

From another viewpoint, metaheuristic algorithms are also classified based on the number of agents used in the search strategy, including single-solution (also known as a single-agent) and population-based solution (also known as a multi-agent). The classification based on the number of agents provides a clear view of the algorithms (2) and has motivated some researchers in describing metaheuristic algorithms (4,10,25–28). Single-solution and population-based metaheuristic algorithms have both been successfully applied to many optimization problems.

Single-solution algorithms begin with a random generation of one solution, and it is improved over the optimization process. Although the implementation of single-solution algorithms is easier and their computational complexity is lower, they tend to get trapped in local optima (29). In contrast, the optimization process for population-based algorithms starts with a random generation of a set of solutions. The agents in population-based algorithms share the information among them, where the best-so-far solution is selected to improve the solution throughout various iterations. According to (30), population-based metaheuristic algorithms have a better ability of exploration and better avoidance of local optima stagnation compared to the single-solution metaheuristic algorithms.

The trends of metaheuristic algorithms in terms of the inspirational source and number of agents are reviewed in Table 1. A total number of 35 existing metaheuristic algorithms from 2017 to 2020 are randomly picked and listed.

Categories 1 to 4 are categorized based on the inspirational source (categories 1 to 3: nature-inspired-based (NI) algorithm, and category 4: non-nature-inspired-based (non-NI) algorithm), whereas categories 5 and 6 are established based on the number of agents used (category 5: single-solution algorithm and category 6: population-based algorithm).

As shown in Table 1, NI algorithms, especially those from the EAs group (category 1) and SI group (category 2), dominate the number of metaheuristic algorithms. Only nine out of the 35 listed algorithms are non-NI algorithms.

Table 1 shows that researchers are more likely to develop a population-based algorithm than a single-solution one. Only three out of 35 represent single-solution algorithms. This finding is similar to those reported by (31) and (32). In their papers, only three out of 50 algorithms (from 1975 to 2017) and only two out of 74 algorithms (from 1951 to 2017) are single-solution algorithms, respectively.

Table 1. Categories of some existing metaheuristic algorithms from 2017 to 2020
(category: 1-EAs, 2-SI, 3-other NI, 4-non-NI, 5-single-solution, 6-population-based)

Algorithm	1	2	3	4	5	6
Artificial butterfly optimization (ABO) (33)		√				√
Chemotherapy science algorithm (CSA) (4)			√			√
Electro-search (ES) (34)			√			√
Grasshopper optimization algorithm (GOA) (15)		√				√
Human mental search (HMS) (27)				√		√
Kidney-inspired algorithm (KA) (35)			√			√
Multi-objective heuristic Kalman algorithm (MOHKA) (36)						√
Salp swarm algorithm (SSA) (14)		√				√
Selfish herd optimizer (SHO) (37)		√				√
Spotted hyena optimizer (SHO) (16)		√				√
Thermal exchange optimization (TEO) (38)			√			√
Tug of war optimization (TWO) (39)				√		√
Elephant swarm water search algorithm (ESWSA) (10)		√				√
Emperor penguin optimizer (EPO) (30)		√				√
Farmland fertility (FF) (40)				√		√
Pity beetle algorithm (PBA) (17)		√				√
Queuing search (QS) (41)				√		√
Social engineering optimizer (SEO) (31)				√	√	
Single-solution simulated Kalman filter (ssSKF) (42)				√	√	
Tree growth algorithm (TGA) (32)	√					√
Volleyball premier league (VPL) (43)				√		√
Racoon optimization algorithm (44)	√		√			√
Harris hawks optimization (45)		√	√			√
Pathfinder algorithm (46)		√	√			√
Single seekers society (47)	√		√			√
Squirrel search algorithm (48)		√	√			√
Emperor penguins colony (49)		√	√			√
Sailfish optimizer (18)		√	√			√
Algorithm of the innovative gunner (50)				√	√	
Fitness Dependent Optimizer (51)		√	√			√
Black Widow Optimization Algorithm (52)	√					√
Marine Predators Algorithm (53)		√				√
Tunicate Swarm Algorithm (54)		√				√
Wingsuit Flying Search (55)				√		√
Monta Ray Foraging Optimization Algorithm (56)		√				√

Most of the algorithms listed in Table 1 fall into the population-based, swarm intelligence, and nature-inspired categories. It is worth mentioning that all EAs and SI algorithms are population-based. Both need to employ population-based aspects to implement the evolution process and swarming behavior, respectively. However, population-based algorithms are not necessarily bound to the concept of either EAs or SI in their algorithms.

A large number of population-based algorithms are indirectly related to a large number of EAs and SI algorithms, where the population-based solutions need to be employed for both EAs and SI algorithms. For example, for the Tree Growth Algorithm (TGA) (32), the founders need to develop a population-based algorithm because the inspirational source is from the evolutionary process of trees in nature. TGA is grouped under the EAs category. The same decision holds for the Salp Swarm Algorithm (SSA) (14), by which the inspirational source is from the swarming behaviors of salps. Thus, SSA is developed as a population-based algorithm. Unlike TGA, SSA is categorized under the SI category.

Therefore, we can conclude that the inspirational source plays an important role and contributes to the decision of researchers to either develop a single-solution or a population-based algorithm.

Apart from classification, the more important feature of metaheuristic algorithms is the ability to balance between the exploration and exploitation phases (57). The former is a process of investigating the search space area as wide as possible to secure the solution. During the exploration phase, a stochastic operator is needed for the search (12). In contrast, the latter is a process of narrowing the search to a specific promising area, known as a local search.

Balancing the exploration and exploitation phases is a challenging task that requires thorough strategies. Thus, many researchers attempt to develop new metaheuristic algorithms by inspiring a convincing inspirational source, that can balance both important phases for solving optimization problems.

The inspirational source can also be driven by the estimation concepts in state-space models. Infinite impulse response (IIR) filters (including the Kalman filter (KF)) and finite impulse response (FIR) filters (58) are two popular types of estimators for state estimations, especially in control engineering applications. Both use the mathematical state-space model of systems and measurements to estimate the state. FIR filters became popular and are preferred by researchers because of the robustness and stability in their structures (59,60). Variants of FIR filters have been proposed by researchers. The ultimate iterative unbiased finite impulse response (UFIR) filter is one of them.

Different from KF, a UFIR filter completely ignores the noise statistics, error covariance, and initial value to calculate the state estimate. It also provides a fast iterative Kalman-like approach in a simpler form to improve the estimation process (61,62). The advantages of the UFIR filter meet industry requirements (63). To note, UFIR is the most robust among the FIR variants as stated in (61,64,65). UFIR filters have been effectively applied in numerous engineering applications, including applications in global positioning systems (GPS)-based vehicle tracking over a wireless sensor network (WSN) (66), an electrocardiogram (ECG) data for features extraction (67,68), and state estimation of carbon monoxide concentration (69–71).

The favorable performance of UFIR filters motivates the transformation of this estimator into a new optimizer. Previously, the UFIR filter framework has been used as an inspiration for a single-based metaheuristic optimization algorithm, called the Single-Agent Finite Impulse Response Optimizer (SAFIRO), as reported in (72–75). The findings show that SAFIRO has a good performance as a single-solution metaheuristic algorithm.

Thus, this paper introduces a new population-based metaheuristic optimization algorithm inspired by the UFIR filter framework, named the Multi-Agent Finite Impulse Response Optimizer (MAFIRO). This study investigates the potential of using work procedures of the UFIR filter in a population-based metaheuristic algorithm. MAFIRO operates with a group of agents to seek an optimal or near-optimal solution in solving numerical optimization problems. MAFIRO adopts the framework of the UFIR filter by performing the measurement and estimation of the solution.

Besides, MAFIRO implements a random mutation of the best-so-far solution ($X_{bestsofar}$) and shrinking local neighborhood method in finding a solution. The exploration in MAFIRO is encouraged by a random mutation of $X_{bestsofar}$; meanwhile, the exploitation is encouraged by a shrinking local neighborhood. It is important to mention that the term “mutation” in this paper is dissimilar from the context of “mutation” in GA. In this study, “mutation” means the displacement of a new measurement that is not fully following the $X_{bestsofar}$ value.

The performances of MAFIRO are evaluated and compared with other population-based metaheuristic algorithms. The comparisons are based on the mean fitness value and statistical ranking obtained in solving the IEEE Congress on Evolutionary Computation (CEC) 2014 benchmark test suite (76). The CEC 2014 benchmark test suite contains 30 mathematical functions represent real-parameter numerical optimization problems. Three state-of-the-art population-based metaheuristic algorithms are considered when comparing the results of the proposed algorithm. The algorithms are the Particle Swarm Optimization (PSO) algorithm (77), Genetic Algorithm (GA) (78), and Grey Wolf Optimizer (GWO) (12). SAFIRO is excluded in the performance comparison, as it is classified in a different class of metaheuristic algorithms. Moreover, MAFIRO is developed to not compete with SAFIRO.

The experimental results show that the proposed MAFIRO performed significantly better than all tested algorithms by leading 23 out of 30 functions. Based on the results, we revealed that the framework of the UFIR filter is also a good inspirational source for a population-based metaheuristic algorithm.

The remainder of this paper is organized as follows: Section 2 briefly describes the KF, FIR, and UFIR filters. This is followed by the descriptions of the proposed MAFIRO and the experimental procedure for evaluating the performance of MAFIRO. Section 3 presents the results and discussions. Finally, Section 4 briefly provides the conclusion and scope for future works.

2. Material and Methods

2.1. Finite Impulse Response Filter

A state-space representation is commonly used in various engineering applications. It includes a linear discrete time-invariant (DTI) system, which can be applied to solve an estimation problem. DTI systems are generally modeled in a state-space by the state (the input) equation, as shown in (1) and the measurement (the output) equation, as shown in (2), respectively (79).

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}\mathbf{w}_n \quad (1)$$

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{v}_n \quad (2)$$

The $\mathbf{x}_n \in \mathbb{R}^R$ is a state vector that describes the system's variables of interest such as a position, velocity, or acceleration at the discrete time-index, n ; R denotes the number of states; $\mathbf{A} \in \mathbb{R}^{R \times R}$ is a state transition matrix that projects the previous state, $\mathbf{x}_{n-1}$ to the current state, $\mathbf{x}_n$; $\mathbf{y}_n \in \mathbb{R}^M$ is a measurement vector that represents a measurement observation; $\mathbf{C} \in \mathbb{R}^{M \times R}$ is a measurement transition matrix that projects the measurements onto the state vector variables; $\mathbf{w}_n \in \mathbb{R}^R$ is a process noise vector; $\mathbf{v}_n \in \mathbb{R}^M$ is a measurement noise vector; $\mathbf{B} \in \mathbb{R}^{R \times R}$ is a process noise matrix, and $\mathbf{D} \in \mathbb{R}^{M \times M}$ is a measurement noise matrix.

As mentioned in the introduction part, FIR filters and KFs are two different types of state estimators used to estimate the state variables. Figure 1 shows the KF strategy in estimating the state.

Figure 1. A KF strategy to estimate the state.

In KF, the estimation of the state is improved with the assistance of the measurement update equation and the Kalman gain matrix. To optimally estimate the state at time-index, t , KF requires the correct values of the system noise, $\mathbf{Q}_t$, and the measurement noise, $\mathbf{R}_t$, at each time-index, as well as the initial state, $\hat{\mathbf{x}}_0$, and the error covariance matrix, $\mathbf{P}_0$, at time zero (80). If the correct values of those parameters are not provided, KF might produce errors when the previous error is projected onto the next state estimation. This happens because of KF's infinite impulse response structure. In this case, KF is unable to provide an optimal state estimation anymore.

Hence, the FIR filter has been introduced by Jazwinski in 1968, as an option to KF. Unlike KF, the FIR filter has a finite impulse response structure where the state vector is estimated on a finite number of recent measurements and does not require the initial condition, error-covariance matrix, and noise statistics.

Additionally, the FIR filter is more robust and stable than the IIR filter due to its finite form, due to which the new estimation process does not rely on past estimation information. That means if there is an error in the previous estimation, the error is not projected to the next estimation process (81,82). Since the introduction, several modifications to the basic principle of the FIR filter have been carried out to improve its performance. The receding horizon FIR filter proposed by Ahn *et al.* (83), and the fast iterative form for the FIR filter by Zhao *et al.* (84,85) are among the significant modifications of FIR filters.

In 2016, Shmaliy *et al.* introduced the UFIR filter to provide a fast near-optimal estimation in a simpler form (61). This UFIR filter operates with two sets of mathematical equations: the batch form and the iterative form. The batch form is used to generate an initial value of the state estimate, whereas the iterative form is used for fast computation of state estimations.

As already mentioned, the estimation process in UFIR is implemented in a finite length according to its N . This means that the UFIR filter estimates the state based on N 's recent measurements, which start at time-index, $t=N$ until the maximum time-index, T .

As illustrated in Figure 2, the UFIR filter produces its initial state estimation through the batch form part (at point $m=t-N+1$ and point s), where all measurements are handled at one time. Thus, the initial estimation value is generated at point s by a convolution process between the gain, $\bar{H}_{m,s}$, and the measurement, $Y_{m,s}$, as shown in (3).

Figure 2. A UFIR strategy to estimate the state (by assuming $N=6$).

$$\bar{x}_s = \bar{H}_{m,s} Y_{m,s} \quad (3)$$

Matrix $\bar{H}_{m,s}$ is the gain or the coefficient of the filter impulse response represented in a state-space and can be obtained as in (4).

$$\bar{H}_{m,s} = G_s C_{m,s}^T \quad (4)$$

Vector $Y_{m,s}$ and matrix $C_{m,s}$ represent the measurement values and the measurement matrix, respectively, within the batch form points: m and s . Matrix G_s is known as the generalized noise power gain (GNPG) and can be computed as in (5). The superscript $(^T)$ denotes the transposed operation.

$$G_s = (C_{m,s}^T C_{m,s})^{-1} \quad (5)$$

The initial state estimation, $\bar{x}_s$, is then updated by an iterative form part (starting from point l until point $l=k$) to improve the estimation. The iterative state estimation, $\bar{x}_l$, can be computed as in (6) where A is a transition matrix of the iterative form; $\bar{x}_{l-1}$ is a previous state estimation; C is a measurement matrix; and K_l is a Kalman-like correction gain (also known as bias correction gain).

$$\bar{x}_l = A \bar{x}_{l-1} + K_l (y_l - C A \bar{x}_{l-1}) \quad (6)$$

The matrix K_l can be calculated as in (7), where G_l is the GNPG for the iterative part. The GNPG is a matrix used together with the optimal N to compensate for the deficiency of noise information and the estimation error, by tuning the K_l value as near as possible to the Kalman gain (80).

$$K_l = G_l C^T \quad (7)$$

The GNPG can be calculated recursively as in (8), where $\mathbf{G}_{l-1}$ is the GNPG matrix for the nearest previous point.

$$\mathbf{G}_l = [\mathbf{C}^T \mathbf{C} + (\mathbf{A} \mathbf{G}_{l-1} \mathbf{A}^T)^{-1}]^{-1} \quad (8)$$

In MAFIRO, the UFIR filter framework is used as an inspiration in the search strategy to find an optimal or near-optimal solution. The UFIR filter in the DTI system is adopted in modeling MAFIRO because the optimal solution to be estimated is time-independent. The interest in this work is a scalar value. Hence, all vectors and matrices in (3) to (8) are reduced to a scalar value with a 1×1 dimension. The state in (6) consists of only one parameter (variable) that holds an agent's estimated position in the search space. The comparisons between a UFIR filter and MAFIRO are listed in Table 2.

Table 2. The comparisons of a UFIR filter and MAFIRO

UFIR filter	MAFIRO
Works as an estimator to provide a near-optimal estimation in solving state-space estimation problems.	Works as an optimizer (by a set of agents) of metaheuristic algorithms to provide an optimal or near-optimal solution in solving optimization problems. The search strategy is inspired by the UFIR filter framework.
Usually, it is involved with the state vector and represents several parameters such as position, acceleration, and velocity.	Considers a scalar representation, which is the estimated position of the agents.
Measurement values are obtained from the sensor.	The measurement value is simulated using a random mutation of $X_{bestsofar}$ and the shrinking local neighborhood method.
The initial estimation value is computed in batch form by using the discrete convolution-based method.	The initial estimation position is generated by using a uniformly distributed random number generator in the range of [0,1].

Unlike a UFIR filter that works as an estimator to calculate a near-optimal solution for estimation problems, MAFIRO works as an optimizer to search for an optimal or near-optimal solution in solving optimization problems. In contrast to the UFIR filter that obtained measurement readings from the sensor, agents of MAFIRO simulate their measurement by using a random mutation of $X_{bestsofar}$ and the shrinking local neighborhood method.

A UFIR filter computes its initial estimation by using the discrete convolution-based in batch form part. On the other hand, MAFIRO generates its initial estimation by using a uniformly distributed random number generator in the search space. Further details about the process are explained in the next section.

2.2. The Proposed Multi-agent Finite Impulse Response Optimizer (MAFIRO)

2.2.1. Strategy of MAFIRO

An agent in any metaheuristic algorithm is responsible for finding an optimal or near-optimal solution, for the given optimization problem. The search for a solution depends on the strategy of the algorithm. This strategy differentiates between one algorithm and another, according to its inspirational source. As aforementioned, in MAFIRO, the UFIR filter framework is used as a source of inspiration for agents to find the solution. Because MAFIRO is a population-based algorithm, it involves a set of agents that work together to search for an optimal or near-optimal solution.

As shown in Figure 3, the images of several persons symbolize each agent in MAFIRO that works as an individual UFIR filter. Those agents need to perform the measurement and estimation of the position that represents a measurement and estimation of the solution, respectively. Subsequently, the scale of justice image represents an evaluation step in the algorithm where the fitness of the estimated solution will be evaluated. The

image of a person holding a flaming torch symbolizes the best agent with its $X_{bestsofar}$ value. To note, $X_{bestsofar}$ represents the best-so-far solution. Lastly, a trophy image visualizes an optimal solution or near-optimal solution achieved by the algorithm.

Figure 3. Two tasks performed by each MAFIRO agent.

In MAFIRO, a measurement of position (state measurement) is assumed as a measurement of the solution, $Y_i(t)$, whereas estimation of position (state estimation) is assumed as a solution, $X_i(t)$. Variable t is the iteration number of the search strategy to improve the solution, while subscript (i), represents the agent. The principle of MAFIRO is shown in Figure 4. In MAFIRO, there are P agents involved in seeking the solution. Variable N is the horizon length, specifying the number of measurements needed by each agent to do the estimation. Meanwhile, $\{Y_i(t - N + 1), \dots, Y_i(t)\}$ are N initial measurements that are randomly generated during the initialization stage.

Figure 4. The principle of MAFIRO.

During the measurement stage, each i^{th} agent will produce a new measurement of the solution, $Y_i(t)$, for each new iteration. This new measurement is simulated by using a random mutation of $X_{bestsofar}$ and the

shrinking local neighborhood method. Then, the value of the measurement is passed along to the estimation stage for further action.

During the estimation stage, each agent needs to undergo two stages: initial estimation and iterative estimation. The estimation of the i^{th} agent, $\mathbf{X}_i(t)$, at iteration, t is defined as (9), where x_i^d denotes the estimated solution of the i^{th} agent in the d^{th} dimension, and D indicates the maximum number of the dimension.

$$\mathbf{X}_i(t) = \{x_i^1(t), x_i^2(t), \dots, x_i^d(t), \dots, x_i^D(t)\} \quad (9)$$

for $i = 1, 2, \dots, p$

The fitness of the estimated solution, $\mathbf{X}_i(t)$, for each agent is then evaluated according to the objective function. All agents are compared with each other to determine which one has the best fitness value. Figure 5 shows how MAFIRO agents communicate with each other using global topology communication to share information about their fitness.

Figure 5. Global topology communication among MAFIRO agents.

In this stage, the agent with the best fitness of the estimated solution is identified. The fitness value from the best agent is then compared to the current fitness value of $X_{bestsofar}$. If a better solution is found, then the $X_{bestsofar}$ will be updated.

All procedures are repeated until they reach the maximum iteration. The optimal solution for MAFIRO is represented by the value of the $X_{bestsofar}$. Further details of each stage of MAFIRO are discussed in the next subsection.

2.2.2. Procedures of MAFIRO

MAFIRO consists of four major stages: initialization, measurement and estimation, fitness evaluation, and update $X_{bestsofar}$. Pseudocode 1 shows the entire procedure of MAFIRO, specifically for the minimization problem.

MAFIRO begins the optimization process at line 01, where all agents perform the initialization stage. After that, all agents perform the measurement and estimation stage from line 03 until line 15. This is followed by the fitness evaluation stage at line 16. Next, the agent that has the best fitness value is determined at line 17. Then, the fitness value from the best agent is compared to the fitness value of $X_{bestsofar}$. If a better solution is found, then the $X_{bestsofar}$ is updated at line 19.

The procedures of line 03 until line 20 repeat until the stopping condition, which is the maximum number of function evaluations ($maxFES$) is met. Once the $maxFES$ is achieved, the $X_{bestsofar}$ is returned as the optimal solution to the given optimization problem.

Pseudocode 1. Procedures of MAFIRO

Algorithm: MAFIRO for a minimization problem.

Requirement: horizon length, N , and coefficient value, β

```
01:      Initialization step
02:      While not a maximum iteration, do
03:          for each agent,  $i$ 
04:              generates a random value for each dimension of a new measurement
05:              if the dimension has a random value  $\leq 0.5$ 
06:                  assigned a new measurement as (11)
07:              else
08:                  compute a new measurement using (12) & (13)
09:              end
10:          at sub-iteration,  $k=2$ 
11:              generate a random initial estimation
12:          for  $k=3: N$ 
13:              compute an iterative estimation using (14) & (15)
14:          end
15:           $\mathbf{X}_i(t) = \bar{\mathbf{X}}_i(k)$ 
16:          Evaluate fitness for agents
17:          Determine the best agent that has the best fitness value
18:          end agent
19:          Update  $\mathbf{X}_{bestsofar}$ 
20:           $t \leftarrow t+1$ 
21:      end while
22:      Return  $\mathbf{X}_{bestsofar}$ 
```

1. Initialization Stage

As MAFIRO mimics the UFIR filter framework, the first step taken in MAFIRO is to define the value of N . MAFIRO needs N most recent measurements to start the search of a solution within the search space. The search space is the area for finding a solution within the lower limit, X_{min} , and the upper limit, X_{max} . The value of X_{min} and X_{max} are dependent on the given optimization problem.

By considering the computational time, the ideal value of N in MAFIRO is assigned as four. The first two points in the sub-iterations of MAFIRO are used to generate the initial estimation, whereas the third point is used as the beginning of the iterative part. Therefore, during the initialization stage, each MAFIRO agent begins its operation by randomly generating four initial measurements, $\mathbf{Y}_i(t-3)$, $\mathbf{Y}_i(t-2)$, $\mathbf{Y}_i(t-1)$, and $\mathbf{Y}_i(t)$, by using (10). Random values are used to intensively explore the search space in finding a solution.

$$\mathbf{Y}_i(t) = rand(U[X_{min}, X_{max}]) \quad (10)$$

Then, the fitness of these random initial values for all i^{th} agents are evaluated to determine the initial $\mathbf{X}_{bestsofar}$. For the minimization problem, the initial value that has the smallest fitness value is assigned as the initial $\mathbf{X}_{bestsofar}$. Conversely, for the maximization problem, the initial value that has the largest fitness value is assigned as the initial $\mathbf{X}_{bestsofar}$.

The number of function evaluations ($maxFES$) as a stopping condition is also defined during the initialization step, where $maxFES = \text{no. of iterations} \times \text{no. of dimensions}$.

2. Measurement and Estimation Stage

After obtaining an initial $\mathbf{X}_{bestsofar}$, all MAFIRO agents perform the measurement and estimation of the solution. In the actual operation of the UFIR filter, the measurement readings can be taken from the sensor. However, in MAFIRO, there is no mechanism to obtain the measurement readings. Therefore, the measurement solution of MAFIRO is simulated by using a random mutation of $\mathbf{X}_{bestsofar}$ and the shrinking local neighborhood method.

The mechanism of mutation is applied at the measurement stage to diversify the solution by giving a new solution for certain dimensions of the agents. This approach encourages the exploration process in MAFIRO as a way to escape the local optima's trap. On the other hand, a local neighborhood method is implemented to encourage the exploitation process, where the search area shrinks according to the exponential decay equation.

Each dimension of the MAFIRO agents is associated with a uniformly distributed random number between zero and one. Figure 6 illustrates the MAFIRO agents with three dimensions. The number of dimensions is dependent on the optimization problem, which reflects the level of its difficulty. The higher the dimension value, the more difficult it is to solve the problem.

Figure 6. The examples of agents with 3 dimensions.

In the proposed method, not all dimensions of the agents are mutated. Dimensions that have a random value of ≤ 0.5 are not involved in the mutation process. The cutoff value of 0.5 is chosen to give an equal possibility (45) that the dimension will be mutated. The measurement values for the respective dimensions are taken directly from the value of $X_{bestsofar}$, as in (11).

$$Y_i^d(t) = X_{bestsofar}_d \quad (11)$$

Meanwhile, dimensions with a random value of > 0.5 will undergo a random mutation process to produce a new measurement. The equation for this measurement is given as in (12), where the value of $X_{bestsofar}$ is added with the random value of the local neighborhood.

$$Y_i^d(t) = X_{bestsofar}_d + rand(U[-\delta, \delta]) \quad (12)$$

Here, the mutation is conducted in the shrinking local neighborhood of $X_{bestsofar}$, as depicted in Figure 7.

Figure 7. The shrinking local neighborhood strategy in MAFIRO.

The local search operation scales down the search area, where the search is centered around $X_{bestsofar}$. The radius of the local neighborhood, δ , shrinks based on (13), where t is the current iteration. Meanwhile, T is the number of the maximum estimations, and β is an adaptive coefficient value. In (13), the exponential decay equation is multiplied with half of the search space. This is to reduce the possibility of obtaining an infeasible solution.

$$\delta = e^{-\beta \times \frac{t}{T}} \times \frac{X_{max} - X_{min}}{2} \quad (13)$$

An adaptive coefficient value, β , is used to control the reduction size of the neighborhood. Figure 8 shows the plots for the exponential term of the delta, δ , with variant values of β . A higher value of β produces a larger step size and leads to a faster transition from the exploration to the exploitation phase. However, a larger step size

increases the possibility that the global optimal value will be missed. In contrast to a higher value, a lower value of β produces a smaller step size and leads to a slower transition.

Figure 8. The plot of local neighborhood radius, δ , with different adaptive coefficient β values.

Having a smaller step size reduces the possibility of missing the global optimal. As visualized in Figure 8, $\beta=1$ has a slower transition, whereas $\beta=20$ has a faster one. Meanwhile, $\beta=10$ has a moderate transition from the exploration to the exploitation phase. A faster convergence may cause a premature convergence, whereas a slower convergence may increase the computational time. Hence, for MAFIRO, the value of $\beta=10$ is implemented to give a moderate transition from the exploration phase to the exploitation phase.

After obtaining a measurement of the solution, $Y_i(t)$, at the measurement stage, all agents are moved to the estimation stage. Here, N most recent measurements are needed to improve the solution of the agents. Each agent estimates the solution in a finite length according to the value of N . In MAFIRO, each iteration, t , consists of the sub-iteration, k . The value of k must be the same as the value of N .

As illustrated in Figure 9, the estimation stage is divided into two parts: initial estimation and iterative estimation. The former is intended to randomly generate the initial estimation ($\bar{X}_i(2)$ at $k=2$), whereas the latter is meant to iteratively improve the estimation from $k=3$ until $k=N$. The sub-iteration, k , indicates the operation of both estimation parts.

Figure 9. A graphical representation of the MAFIRO operation.

In an actual UFIR filter, the initial estimation, $\bar{x}_s$, is calculated by using the convolution-based method as in (3) to (5). However, in MAFIRO, the initial estimation, $\bar{X}_i(k=2)$, is randomly generated within the [lower limit, upper limit] of the first two points in k , as shown in Pseudocode 2. A random element that is uniformly distributed in the range of $[0,1]$ is used for the stochastic nature of MAFIRO.

Pseudocode 2. Procedures of initial estimation, $\bar{X}_i(k=2)$, in MAFIRO

```
01:   if  $Y_i(t - N + 1) < Y_i(t - N + 2)$ 
02:        $\bar{X}_i(k=2) = rand(U[Y_i(t - N + 1), Y_i(t - N + 2)])$ 
03:   else
04:        $\bar{X}_i(k=2) = rand(U[Y_i(t - N + 2), Y_i(t - N + 1)])$ 
05:   end
```

The iterative estimation, $\bar{x}_{l,i}$, for a UFIR filter in (6) is transformed to (14) for MAFIRO. As explained earlier, in this study, the state vector is reduced to a scalar representing only one variable, which is the estimated position. Hence, the state transition matrix, A , in (6) and (8) is equivalent to 1. The measurement transition matrix, C , from (4) until (8) is set as 1, indicating that the measurement and the state estimate have the same scale, as in (62). Thus, the computation of the estimated solution, $\bar{X}_i(k)$, can be simplified as in (14).

$$\bar{X}_i(k) = \bar{X}_i(k-1) + K(k) (Y_i(t - N + k) - \bar{X}_i(k-1)) \quad (14)$$

As depicted in Figure 9, the initial estimated value, $\bar{X}_i(k=2)$, is iteratively improved during the iterative part by using (14) from $k=3$ until $k=N$. The $\bar{X}_i(k-1)$ represents the estimated value for the most recent sub-iteration point. As C is assumed to be equal to 1, the computation of GNPG in (8) can be simplified to $1/k$. Thus, the Kalman-like gain value in (7) is equal to the GNPG value in MAFIRO, as shown in (15).

$$K(k) = \frac{1}{k} \quad (15)$$

The formula $K(k)(Y_i(t - N + k) - \bar{X}_i(k-1))$ is a corrective element that will improve the estimation of the solution. $K(k)$ serves as a weighted average between the current measurement, $Y_i(t - N + k)$, with the previously estimated value, $\bar{X}_i(k-1)$, to compute the current estimated value, $\bar{X}_i(k)$, in the sub-iteration, k . The value in $K(k)$ decreases as the sub-iteration, k , increases. This condition helps to improve the estimation of the solution. The sub-iteration operation ends when it reaches $k=N$.

Then, the final estimation value of k , which is $\bar{X}_i(k)$, is assigned as the estimated value of t , where $X_i(t) = \bar{X}_i(k)$. The estimated solutions of $X_i(t)$ represent the candidate solution by each agent for that corresponding iteration. These estimated positions are then passed along to the fitness evaluation stage for further action.

3. Evaluation Stage

Each agent undergoes a fitness evaluation stage so that the quality or the fitness level of its estimated solution, $X_i(t)$, obtained from the estimation stage, can be assessed. In each iteration, every MAFIRO agent undergoes the fitness evaluation once.

The fitness level is measured according to a fitness function, also known as an objective function. An objective function is a criterion or mathematical modeling with respect to an optimization problem that needs to be solved by the algorithm.

4. Update of The Best-so-far Solution

All agents communicate with each other (Figure 5) to identify which agent has the best fitness value. The agent with the best fitness value for the corresponding iteration is assigned as the best agent. The fitness from the best agent is then compared to the fitness of $X_{bestsofar}$.

The value of $X_{bestsofar}$ will be updated if a better solution is found. For the minimization problem, $X_{bestsofar}$ is updated when the fitness of the best agent is smaller than the fitness of $X_{bestsofar}$. On the contrary, for the maximization problem, $X_{bestsofar}$ is updated when the fitness of the best agent is larger than the fitness of $X_{bestsofar}$.

The search strategy (measurement and estimation) stage, fitness evaluation stage, and $X_{bestsofar}$ updates are repeated for the second iteration until reach the stopping condition. These stages need to be repeated to improve the estimated solution and to avoid the local optima trap.

MAFIRO brings the $X_{bestsofar}$ value to the next iteration. In the following iteration, the $X_{bestsofar}$ is updated if a better fitness is found. Once the maximum iteration is reached, the $X_{bestsofar}$ returns as the optimal or near-optimal solution for the given optimization problem.

2.2.3. MAFIRO versus SAFIRO

This subsection compares the procedures between the newly proposed MAFIRO and the previously proposed SAFIRO. The UFIR filter framework is used as an inspirational source in the development of both MAFIRO and SAFIRO. Table 3 shows a comparison of MAFIRO and SAFIRO. In finding the solution for the optimization problems, both MAFIRO and SAFIRO undergo four stages: the initialization of the solution, the measurement and estimation of the solution, a fitness evaluation, and the best-so-far update.

However, MAFIRO differs from SAFIRO in terms of the number of agents used in finding the optimal solution. Instead of using one agent, MAFIRO uses a set of agents. During the initialization step, the agent of SAFIRO and each agent of MAFIRO randomly produces N initial measurements to start the optimization process. In producing a new measurement, both algorithms apply a random mutation of the best-so-far solution and the shrinking local neighborhood method for the dimension that has a random value > 0.5 .

The exploration is encouraged in both algorithms by a random mutation of the best-so-far solution, whereas the exploitation is encouraged by the shrinking local neighborhood method. In improving the solution during the estimation step, MAFIRO and SAFIRO perform two stages of estimations: the initial estimation and the iterative estimation. However, the equations for the iterative estimations for MAFIRO and SAFIRO are different due to the number of agent/s used.

Every agent in MAFIRO and SAFIRO performs the measurement, estimation, and fitness evaluation once for each iteration. In SAFIRO, for each iteration, an update of the best-so-far solution is done if the fitness of the current solution is better than the fitness of the best-so-far solution. Meanwhile, in MAFIRO, an update of the best-so-far solution is done if the fitness of the best agent in the current population is better than the fitness of the best-so-far solution. Both MAFIRO and SAFIRO use little memory, as these algorithms need to memorize only the value of the best-so-far solution.

Table 3. The comparisons of MAFIRO and SAFIRO

	MAFIRO	SAFIRO
No. of agents	A set of agents	One agent
Initialization	$Y_i(t) = \text{rand}(U[X_{min}, X_{max}])$	$Y(t) = \text{rand}(U[X_{min}, X_{max}])$
Measurement and estimation	<p>The number of measurement and estimation steps = number of agents.</p> <p>Each agent undergoes the measurement and estimation step once per iteration.</p> <p>$Y_i^d(t) = X_{bestsofar_d}$ or</p> <p>$Y_i^d(t) = X_{bestsofar_d} + \text{rand}(U[-\delta, \delta])$</p> <p>$\bar{X}_i(k) = \bar{X}_i(k-1) + K(k)(Y_i(t-N+k) - \bar{X}_i(k-1))$</p>	<p>The agent undergoes the measurement and estimation stage once per iteration.</p> <p>$Y_d(t) = X_{bestsofar_d}$ or</p> <p>$Y_d(t) = X_{bestsofar_d} + \text{rand}(U[-\delta, \delta])$</p> <p>$\bar{X}(k) = \bar{X}(k-1) + K(k)(Y(t-N+k) - \bar{X}(k-1))$</p>
Fitness evaluation	<p>The number of fitness evaluations = number of agents.</p> <p>Each agent does the fitness evaluation once in one iteration.</p>	The agent does the fitness evaluation once per iteration.
$X_{bestsofar}$ update	<p>$X_{bestsofar}$ is updated once per iteration.</p> <p>In each iteration, $X_{bestsofar}$ will be updated if the fitness of the best agent is better than the fitness of $X_{bestsofar}$.</p>	<p>The agent updates $X_{bestsofar}$ once per iteration.</p> <p>In each iteration, $X_{bestsofar}$ will be updated if the fitness of the current solution is better than the fitness of $X_{bestsofar}$.</p>

2.3. Experiment

2.3.1 CEC 2014 Benchmark Test Suite of Single Objective Real-Parameter Numerical Optimization

The proposed MAFIRO is successfully developed using MATLAB software. The competency of MAFIRO is tested by solving 30 single-objective mathematical test functions available in the CEC 2014 benchmark test suite (76). The CEC 2014 benchmark test suite, which contains the real-parameter minimization optimization problems, has been used by several researchers for various algorithms, including the Manta Ray Foraging Optimization (MRFO) algorithm (56), the Squirrel Search Algorithm (SSA) (48), the Pity Beetle Algorithm (PBA) (17), the Queuing Search (QS) algorithm (41), the Biology Migration Algorithm (BMA) (86), and the Sine Optimization Algorithm (SOA) (87).

Table 4. The CEC 2014 benchmark test suite (source: (76))

Types	No.	Functions	Ideal Fitness
Unimodal functions	1	Rotated High Conditioned Elliptic function	100
	2	Rotated Bent Cigar function	200
	3	Rotated Discus function	300
Simple multimodal functions	4	Shifted and Rotated Rosenbrock's function	400
	5	Shifted and Rotated Ackley's function	500
	6	Shifted and Rotated Weierstrass function	600
	7	Shifted and Rotated Griewank's function	700
	8	Shifted Rastrigin's function	800
	9	Shifted and Rotated Rastrigin's function	900
	10	Shifted Schwefel's function	1000
	11	Shifted and Rotated Schwefel's function	1100
	12	Shifted and Rotated Katsura function	1200
	13	Shifted and Rotated HappyCat function	1300
	14	Shifted and Rotated HGBat function	1400
	15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's function	1500
	16	Shifted and Rotated Expanded Scaffer's F6 function	1600
Hybrid functions	17	Hybrid function 1 (N=3)	1700
	18	Hybrid function 2 (N=3)	1800
	19	Hybrid function 3 (N=4)	1900
	20	Hybrid function 4 (N=4)	2000
	21	Hybrid function 5 (N=5)	2100
	22	Hybrid function 6 (N=5)	2200
Composition functions	23	Composition function 1 (N=5)	2300
	24	Composition function 2 (N=3)	2400
	25	Composition function 3 (N=3)	2500
	26	Composition function 4 (N=5)	2600
	27	Composition function 5 (N=5)	2700
	28	Composition function 6 (N=5)	2800
	29	Composition function 7 (N=3)	2900
	30	Composition function 8 (N=3)	3000

The CEC 2014 benchmark test suite consists of four different groups: unimodal functions, simple multimodal functions, hybrid functions, and composition functions. These groups represent the real optimization problems. All functions are minimization problems and are designed as black-box problems.

In the CEC 2014 benchmark test suite, the solutions for the given optimization problems are represented in their fitness form, meaning that each function has its optimal or ideal fitness value (Table IV). Further details about these functions can be read in (76). The MATLAB code for the CEC 2014 benchmark test suite can be obtained at http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2014.

2.3.2 Benchmarking against existing metaheuristic algorithms

The competency of MAFIRO is benchmarked against three population-based metaheuristic algorithms found in the literature. The GA and PSO algorithm are selected in this benchmarking experiment, as both are very prominent and established algorithms for the evolutionary computation category and swarm intelligence category, respectively. The codes for the GA and PSO algorithm are taken from the Practical Genetic Algorithms (88) and the website of the CEC 2014 (89), respectively.

Apart from that, the GWO is selected to relatively represent modern metaheuristic algorithms. GWO is among the most popular modern metaheuristic algorithms, with more than 4,200 paper citations. The code for GWO is taken from its author's website. As mentioned in the introduction part, SAFIRO is not involved in the comparison because it is in a different group of metaheuristic algorithms.

To ensure a fair comparison of algorithm performance, all tested algorithms are set to the same platform and experimental parameters (Table 4). All values listed in Table 5 are in accordance with the parameter setting in the CEC 2014 benchmark test suite (76). To provide a fair evaluation for each comparison, the complexity of the problem is set to 50 dimensions. Meanwhile, the *maxFES* is set to 500,000 (*maxFES* = 10,000 iterations $\times$ dimension, *D* (76)). The number of agents used in this experiment is 50.

Table 5. The experimental parameters

Experimental parameters	Values
Number of agents	50
Number of dimensions	50
Number of maximum iterations	10000
Number of runs	51
Search space	[-100,100]

The evaluation is based on the average performance over 51 run times on each test problem. The search space in the range of [-100,100] is used for all functions. The results obtained in this experiment are not normally distributed. Thus, the Friedman test, due to its suitability for non-parametric tests, is conducted to compare the results for all tested algorithms.

Theoretically, the Friedman test defines the null hypothesis as all performances of the algorithms being equivalent to one another, with no significant differences (90). The performances of MAFIRO, PSO algorithm, GA, and GWO are ranked statistically based on their mean fitness. The significant differences between these algorithms are also observed

The post hoc analysis is then carried out using Holm's method to analyze significant differences in the algorithms' performances that were previously detected by the Friedman test. This is done as recommended by Derrac *et al.* (91). Holm's method rejects the null hypothesis if the statistical value is smaller than the *p*-value.

Normally, the value chosen for the tolerance level is either 0.01, 0.05, or 0.1. The larger the value, the easier it is to claim the significant difference. In contrast, the smaller the value, the more rigid it is to identify the significant difference (90). Therefore, in this work, $\alpha=0.05$ is chosen to balance the opportunity to obtain a significant difference with a rigid procedure in detecting the significant difference.

In this test, the KEEL software tool is applied as a platform for performing both the Friedman and Holm post hoc tests. The KEEL software can be downloaded through <http://www.keel.es>.

3. Results and Discussions

3.1. Statistical Analysis

The mean fitness and mean error of MAFIRO, PSO algorithm, GA, and GWO in solving 30 single-objective test functions available in the CEC 2014 benchmark test suite are tabulated in Table 6 to 9. The results in bold font indicate the best results among the algorithms tested for each function (Fn). Overall, MAFIRO shows excellent results by leading 23 out of the 30 functions. Further explanations of the results are given in the following sub-sections.

1. Unimodal Functions (Fn1 to Fn3)

The first group in the CEC 2014 benchmark test suite is provided to test the unimodal optimization problems. Function 1 (Fn1) to function 3 (Fn3) are used to test the algorithms' abilities to solve rotation problems. Fn1 is difficult to solve because the function is related to the *Rotated High Conditioned Elliptic*, which involves a quadratic ill-conditioned property (76). The ideal fitness for Fn1 is 100. Meanwhile, Fn2 is a *Rotated Bent Cigar* function with an ideal fitness of 200. Meanwhile, Fn3 is a *Rotated Discus* function with an ideal fitness of 300.

Overall, MAFIRO provides the best results compared to other algorithms for all unimodal functions. Table 6 shows that MAFIRO is capable of solving rotation optimization problems by achieving the fitness values of $2.13\text{E}+06$ and 7673.24 for Fn1 and Fn2, respectively. MAFIRO managed to solve Fn3 with the optimal solution of 300. As the unimodal functions are related to exploitation benchmarking (12), the results show that MAFIRO performs well in the exploitation process, thus contributing to the success of solving the unimodal functions.

Table 6. The mean fitness and mean error values of MAFIRO, PSO algorithm, GA, and GWO for unimodal functions (numbers in bold represent the best fitness)

Fn	MAFIRO		PSO		GA		GWO	
	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error
1	2.13E+06	2.13E+06	2.02E+08	2.02E+08	1.29E+08	1.29E+08	8.87E+07	8.87E+07
2	7673.24	7473.24	3.84E+08	3.84E+08	4.20E+09	4.20E+09	7.50E+09	7.50E+09
3	300.00	0.00	3.76E+04	3.73E+04	3.57E+04	3.54E+04	5.58E+04	5.55E+04

2. Simple Multimodal Functions (Fn4 to Fn16)

The second group in the CEC 2014 benchmark test suite is provided to test the simple multimodal optimization problems. Simple multimodal functions consist of 13 optimization problems. Most of the functions are involved in shifting and rotation problems, which are related to exploration benchmarking (12). As shown in Table 7, MAFIRO performs well for simple multimodal functions by leading 9 out of 13 functions.

The results clearly indicate that besides being great at exploitation processes, MAFIRO is also good in exploration by providing the best solutions compared to others for Fn4 (Shifted and Rotated Rosenbrock's function), Fn5 (Shifted and Rotated Ackley's function), F6 (Shifted and Rotated Weierstrass function), Fn7 (Shifted and Rotated Griewank's function), Fn11 (Shifted and Rotated Schwefel's function), Fn12 (Shifted and Rotated Katsuura function), Fn13 (Shifted and Rotated HappyCat function), Fn14 (Shifted and Rotated HGBat function), and Fn15 (Shifted and Rotated Expanded Griewank's plus Rosenbrock's function) with fitness values of 501.61, 520.00, 618.51, 700.01, 6301.56, 1200.15, 1300.63, 1400.50, and 1510.86, respectively. The ideal fitness for those functions is 400, 500, 600, 700, 1100, 1200, 1300, 1400, and 1500, respectively.

For Fn16 (Shifted and Rotated Expanded Scaffer's F6 Function), although MAFIRO achieved the second-best solution behind GWO, the result is very near to the ideal fitness of 1600. In this function, GWO managed to get a near-optimal solution with a 19.90 mean error. In the same manner, despite attaining the second-best solution behind PSO, MAFIRO still provides a near-optimal solution for Fn9 (Shifted and Rotated Rastrigin's function) with a 1096.38 fitness value. The ideal fitness for this function is 900. Apart from that, MAFIRO obtained the third-best solution behind PSO and GA for Fn10 (Shifted Schwefel's function). For Fn8 (Shifted Rastrigin's function), MAFIRO acquired the last-best solution. However, the 197.24 mean error is still considered small. The behavior of algorithms that are able to outperform MAFIRO in solving simple multimodal functions is observed in the sub-section of convergence behavior analysis.

Table 7. The mean fitness and mean error values of MAFIRO, PSO algorithm, GA, and GWO for simple multimodal functions (numbers in bold represent the best fitness)

Fn	MAFIRO		PSO		GA		GWO	
	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error
4	501.61	101.61	1923.34	1523.34	1222.90	822.90	1188.10	788.10
5	520.00	20.00	521.08	21.08	520.77	20.77	521.13	21.13
6	618.51	18.51	635.14	35.14	647.09	47.09	630.20	30.20
7	700.01	0.01	704.34	4.34	745.83	45.83	769.57	69.57
8	997.24	197.24	877.93	77.93	926.83	126.83	995.47	195.47
9	1096.38	196.38	1077.83	177.83	1254.10	354.10	1099.10	199.10
10	5935.75	4935.75	1936.31	936.31	3786.80	2786.80	6422.00	5422.00
11	6301.56	5201.56	1.35E+04	1.24E+04	1.01E+04	8998.00	6815.00	5715.00
12	1200.15	0.15	1203.01	3.01	1201.20	1.20	1202.40	2.40
13	1300.63	0.63	1300.65	0.65	1300.80	0.80	1300.70	0.70
14	1400.50	0.50	1400.56	0.56	1408.10	8.10	1416.10	16.10
15	1510.86	10.86	1545.62	45.62	2665.60	1165.60	2715.70	1215.70
16	1620.65	20.65	1622.09	22.09	1621.20	21.20	1619.90	19.90

3. Hybrid Functions (Fn17 to Fn22)

The third group in the CEC 2014 benchmark test suite is provided to test the hybrid optimization problems. Functions 17 to 22 are provided for testing hybrid optimization problems. In hybrid functions, the variables are randomly distributed into several subcomponents, where various subcomponents use different basic functions (76). The hybrid function consists of a combination of several multimodal functions (Fn19, Fn21, and Fn22). It also can be a combination of unimodal functions with simple multimodal functions (Fn17, Fn18, and Fn20). Therefore, the functions are more difficult to handle. Table 8 shows that MAFIRO performs very well in handling hybrid functions. MAFIRO achieved the best solution compared to other algorithms in all hybrid functions.

Table 8. The mean fitness and mean error values of MAFIRO, PSO algorithm, GA, and GWO for hybrid functions (numbers in bold represent the best fitness)

Fn	MAFIRO		PSO		GA		GWO	
	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error
17	1.17E+05	1.15E+05	1.64E+07	1.64E+07	2.50E+07	2.50E+07	4.27E+06	4.27E+06
18	4119.19	2319.19	1.16E+06	1.16E+06	1.99E+07	1.99E+07	4.26E+07	4.26E+07
19	1923.03	23.03	1972.31	72.31	1997.30	97.30	1986.80	86.80
20	2490.75	490.75	1.62E+04	1.42E+04	5.30E+04	5.10E+04	1.70E+04	1.50E+04
21	1.18E+05	1.16E+05	1.49E+06	1.49E+06	1.13E+07	1.13E+07	2.43E+06	2.42E+06
22	2994.67	794.67	4130.12	1930.12	3809.80	1609.80	3000.20	800.20

4. Composition Functions (Fn23 to Fn30)

The fourth group in the CEC 2014 benchmark test suite is provided to test the composition optimization problems. This group of composition functions comprises eight functions, which have many local optima. Therefore, the capability of both exploration and exploitation in the algorithm can be benchmarked together through composition functions (12).

Referring to Table 9, MAFIRO shows a good balance between exploration and exploitation by leading in five out of six composition functions, which are Fn23, Fn25, Fn27, Fn29, and Fn30. MAFIRO delivered

competitive outcomes by achieving the second-best result for Fn28 behind GWO. For Fn24, MAFIRO obtained the third-best solution. Meanwhile, for Fn26, MAFIRO's result was not as good as those of the other three tested algorithms. Function 26 is a combination of one unimodal function (Fn1) and four simple multimodal functions: Fn6 (Rotated Weierstrass Function F6), Fn7 (Rotated Griewank's Function F7), Fn11 (Rotated Schwefel's Function F11), and Fn13 (Rotated HappyCat Function F13). The behaviors of MAFIRO and the other algorithms are observed in the sub-section of convergence behavior analysis.

Table 9. The mean fitness and mean error values of MAFIRO, PSO algorithm, GA, and GWO for composition functions (numbers in bold represent the best fitness)

Fn	MAFIRO		PSO		GA		GWO	
	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error	Mean fitness	Mean error
23	2646.20	346.20	2685.12	385.12	2696.60	396.60	2741.90	441.90
24	2678.55	278.55	2674.05	274.05	2716.40	316.40	2600.00	200.00
25	2713.11	213.11	2737.57	237.57	2742.90	242.90	2724.90	224.90
26	2782.80	182.80	2702.09	102.09	2700.80	100.80	2780.60	180.60
27	3564.83	864.83	4195.82	1495.82	4248.60	1548.60	3777.40	1077.40
28	5296.96	2496.96	1.25E+04	9735.36	6748.20	3948.20	4939.20	2139.20
29	3.04E+04	2.75E+04	3.26E+06	3.26E+06	2.42E+06	2.42E+06	5.41E+06	5.41E+06
30	4.14E+04	3.84E+04	6.02E+05	5.99E+05	1.26E+05	1.23E+05	1.43E+05	1.40E+05

3.2. Friedman Test and Holm Post Hoc Test

The Friedman test is then employed as a statistical analysis tool to rank the performance of MAFIRO against the other three tested algorithms. The rankings are compiled according to the average fitness value for all 30 benchmark functions, over the 51 runs. The average rank is computed for each algorithm, with lower values indicating better algorithm performance. As can be seen in Table 10, MAFIRO is ranked first, followed by PSO, GWO, and GA. Friedman's statistic is performed while considering the reduction performance distributed according to a chi-square value of 28.52 with three degrees of freedom (DOF).

Table 10. The average ranking of the algorithms

Algorithm	Ranking
MAFIRO	1.4333
GA	3.0333
PSO	2.6667
GWO	2.8667

In this experiment, the Friedman test detects significant differences between the algorithms. Therefore, the null hypothesis was rejected and further analysis (applying the Holm post hoc test) (91) was carried out to identify better-performing algorithms. The results of the Holm post hoc test (with significant level, $\alpha=0.05$) are recorded in Table 11. The Holm post hoc test detected significant differences between the performance of MAFIRO, PSO, GWO, and GA. The hypotheses with an unadjusted p -value ≤ 0.016667 are rejected by Holm's procedure. In this test, we showed that the proposed MAFIRO performed significantly better than the other three tested algorithms.

Table 11. The Holm post hoc result for $\alpha=0.05$

Algorithms	p	Holm
MAFIRO vs GA	0.000002	0.008333
MAFIRO vs GWO	0.000017	0.010000
MAFIRO vs PSO	0.000216	0.012500
GA vs PSO	0.271332	0.016667
PSO vs GWO	0.548506	0.025000
GA vs GWO	0.617075	0.050000

3.3. Boxplot

Next, the boxplots are generated to observe the stability of the results in solving the CEC 2014 benchmark test suite. The boxplots in Figure 10(a) to (d) describe the solution (fitness value) at the last iteration, over 51 runs. Only Fn3, Fn6, Fn19, and Fn23 are selected to demonstrate the results here due to space constraints. (The boxplots for Fn1 until Fn30 can be seen in Appendix A). As all benchmark functions are minimization problems, a lower position denotes a higher-quality solution. The size of the boxplot reflects its variance in which a small size indicates a stable performance (90).

Figure 10. The boxplot of MAFIRO, PSO, GA, and GWO for (a) Fn3 (unimodal function); (b) Fn6 (simple multimodal function); (c) Fn19 (hybrid function); (d) Fn23 (composition function).

As shown in Figure 10(a) to (d), generally, the data are not normally distributed where the median does not lie in the middle of the box. Therefore, the non-parametric statistical test has been applied to analyze the data. Figure 10(a) to (d) show a better quality of MAFIRO results than the other algorithms. For those selected functions, MAFIRO displays the lowest position, which is the nearest to the ideal fitness value compared to the other algorithms. The smallest size of the boxplot indicates that MAFIRO has small variances of mean, median, minimum, and maximum values. This condition implies the best performance of MAFIRO in solving those functions.

3.4. Convergence Curve Analysis

The capability of MAFIRO and other algorithms to achieve optimal or near-optimal solutions can be seen by generating convergence curve graphs. Each convergence curve shows the mean fitness against 10,000 iterations over 51 runs.

Statistically, MAFIRO performs very well in solving the CEC 2014 benchmark test suite by leading in 23 out of 30 functions. Among these 30 functions, Fn3, Fn6, Fn19, and Fn23 were selected to represent convergence curves of unimodal, simple multimodal, hybrid, and composition functions here, respectively. The other convergence curves can be shown in Appendix B.

1) Convergence Curve

Referring to Figure 11(a) to (d), an obvious change at the beginning of the search can be seen for each algorithm where the graphs drastically decline and then gradually flatten at some iterations until the iteration ends. This is due to the transition step from the exploration to the exploitation phase in the optimization process, as stated by Berg in (92).

Figure 11. The convergence curve comparison for (a) Fn3 (unimodal function); (b) Fn6 (simple multimodal function); (c) Fn19 (hybrid function); (d) Fn23 (composition function).

As visualized in Figure 11(a) to (d), MAFIRO can obtain the best solution with the fewest number of fitness evaluations. Influenced by the approaches of the random mutation and shrinking local neighborhood, MAFIRO shows a good balance between the exploration and exploitation phase, especially in handling unimodal functions (Figure 11(a)). When solving Fn3, MAFIRO managed to reach the optimal solution, with a fitness value of 300.

The convergence graph in Figure 11(b) shows that MAFIRO has a great performance compared to others in solving Fn6 with the fastest convergence while remaining nearest to the ideal fitness value. In Figure 11(c) and (d), although the functions are involved with hybrid and composition functions, respectively, all the algorithms managed to converge to the near-optimal solutions. Both figures show the excellent performances of MAFIRO, which obtains the best solution compared to the others.

Figure 12(a) to (c) show the convergence graphs where PSO, GA, and GWO reveal better performances than MAFIRO, respectively. The convergence graphs in Figure 12(a) show the capability of all algorithms in escaping the local optima when solving Fn10. As can be seen clearly, MAFIRO and PSO have tried many times to escape from the local optima before gradually converging to provide a near-optimal solution for this function. These conditions happen due to the nature of the *Schwefel* function, which consists of many local optima, as can be seen in Figure 5 in (76). The longer period of the exploration phase has benefitted the PSO to give the best solution compared to others for this function. In this function, MAFIRO obtained the third-best solution behind PSO and GA. Figure 12(b) and (c) show the ability of GA and GWO to reach a better solution than MAFIRO when solving Fn26 and Fn28, respectively. The ability to exploit more search areas has given an advantage to GA and GWO for finding a better solution. In those functions, MAFIRO yields the last-best solution and the second-best solution, respectively.

Figure 12. The convergence curve comparison for (a) Fn10 (unimodal function); (b) Fn26 (composition function); (c) Fn28 (composition function).

Subsequently, the behavior of MAFIRO's search agent is further observed through its trajectory, search history, and fitness trend. To perform the observations, the proposed MAFIRO is tested on the CEC 2014 benchmark test suite with a complexity of two-dimensional (2-D) optimization problems.

A single-agent with 100 iterations is set where only unimodal, simple multimodal, and composition (except for Fn29 and Fn30) functions are involved in these experiments. The hybrid functions are excluded because they are not specified in the 2-D domain. Function 3, Fn13, and Fn23 are selected to visualize the behavior of a MAFIRO agent in finding the best near-optimal solution within the search space area throughout the optimization process.

2) Trajectory of A MAFIRO Agent

The strategy of a MAFIRO agent in seeking the solution is specifically observed where two graphs are shown for each function. These two graphs present the first and second dimensions of the agent. In these graphs, the solutions of a MAFIRO agent are compared to the best-so-far-solution, $X_{bestsofar}$.

As can be seen in Figure 13(a) to (f), for both dimensions, the MAFIRO agent finds the solutions by roaming the search space at the beginning of an iteration and tries to narrow the search nearer to the best-so-far-solution after a certain iteration. The random mutation and shrinking local search neighborhood approach (employed during the measurement stage) allow a MAFIRO agent to get closer to the best-so-far solution and find the final solution before reaching the maximum number of iterations.

Figure 13. The trajectory of a MAFIRO agent for (a) a one-dimensional Fn3 (unimodal function); (b) a two-dimensional Fn3 (unimodal function); (c) a one-dimensional Fn13 (simple multimodal function); (d) a two-dimensional Fn13 (simple multimodal function); (e) a one-dimensional Fn23 (composition function); (f) a two-dimensional Fn23 (composition function).

3) Search History of A MAFIRO Agent

The transition of a MAFIRO agent for exploring and exploiting the search space can also be viewed by tracing its search history. Figure 14 (a) to (c) depict the search history of MAFIRO agent for Fn3, Fn13, and Fn23, respectively. The contour map for a 2-D function is used to mark the changes in a MAFIRO agent throughout the search process. The position of a MAFIRO agent is denoted by a star symbol, representing its estimated solution throughout the iterations. Meanwhile, the circle shape in each figure indicates the position of the final solution. From Figure 14 (a) to (c), we can clearly see that a MAFIRO agent has adequate exploration space before focusing on the promising areas for the exploitation process, which is around the best-so-far solution. Towards the end of the search process, the agent is able to find the final solution for the given optimization problems.

Figure 14. The search history of a MAFIRO agent for (a) Fn3 (unimodal function); (b) Fn13 (simple multimodal function); (c) Fn23 (composition function).

4) Fitness Trend of A MAFIRO Agent

The solution obtained by a MAFIRO agent against the best-so-far-solution, $X_{bestsofar}$, is shown in Figure 15(a) to (c). The obvious changes at the beginning of an iteration are due to the stochastic element applied during the measurement stage. The correction element of the measured solution and the Kalman-like gain during the estimation stage contribute to the improvement of MAFIRO's estimated solution. The stability of the graph towards the end of an iteration indicates that the MAFIRO agent can obtain a better solution before completing the search process (reaching the maximum iteration). In general, MAFIRO agents undergo ample exploration and

exploitation processes and successfully provide the optimal or near-optimal solution for unimodal, simple multimodal, and composition optimization problems.

The findings described above indicate that the overall result of MAFIRO is very impressive. MAFIRO demonstrates a competitive and satisfying performance in solving unimodal, simple multimodal, hybrid, and composition functions. MAFIRO exhibits a high competency to reach the optimal or near-optimal solution by leading in 23 out of the 30 functions in solving the CEC 2014 benchmark test suite. These results show that the proposed MAFIRO manages to iteratively estimate the solution for numerous optimization problems (unimodal, multimodal, hybrid, and composition functions).

Figure 15. The fitness trend of a MAFIRO agent for (a) Fn3 (unimodal function); (b) Fn13 (simple multimodal function); (c) Fn23 (composition function).

4. Conclusion and Future Works

A new population-based metaheuristic optimization algorithm named the Multi-agent Finite Impulse Response Optimizer (MAFIRO) is introduced in this paper. In finding the optimal or near-optimal solution, MAFIRO has been inspired by the estimation equations of the UFIR filter framework. MAFIRO uses a random mutation of $X_{bestsofar}$ and the shrinking local neighborhood method to improve the solutions. The CEC 2014 benchmark test suite, which contains real-parameter minimization optimization problems, was applied to provide benchmark functions for evaluating the performance of MAFIRO. Then, MAFIRO's performance was statistically compared to some well-known metaheuristic algorithms (i.e., the PSO algorithm, GA, and GWO).

MAFIRO can produce optimal or near-optimal solutions and significantly outperformed the other tested algorithms in the Friedman test and Holm post hoc test. Thus, we can conclude that besides performing well as a

single-solution metaheuristic algorithm, the concept of the UFIR filter is also a good inspirational source for a population-based metaheuristic algorithm.

With these excellent results, the proposed MAFIRO provides a convincing option for other researchers to apply this algorithm in solving any optimization problems in numerous fields. Additionally, MAFIRO can be a good platform for other fundamental researchers to modify, enhance, or hybridize it with other algorithms for better performances in handling optimization problems.

Further investigations and strategies for an escape from local optima will be considered for the improvement of MAFIRO. Besides, an asynchronous update mechanism for the algorithm's iteration strategy will be explored. Additionally, the proposed MAFIRO will be applied to solve engineering optimization problems such as for the maximization of wireless sensor network (WSN) coverage and in solving printed circuit board (PCB) routing problems.

References

1. Birattari M, Paquete L, Stutzle T, Varrentrapp K. Classification of Metaheuristics and Design of Experiments for the Analysis of Components [Internet]. Darmstadt, Germany; 2001. Available from: <https://eden.dei.uc.pt/~paquete/papers/AIDA-01-05.pdf>
2. Blum C, Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput Surv* [Internet]. 2003;35(3):268–308. Available from: https://www.iiia.csic.es/~christian.blum/downloads/blum_rol_i_2003.pdf
3. Talbi EG. Metaheuristics from Design to Implementation [Internet]. Hoboken, New Jersey: John Wiley & Sons, Incorporated; 2009. Available from: <https://www.wiley.com/en-us/Metaheuristics%3A+From+Design+to+Implementation+-p-9780470278581>
4. Salmani MH, Eshghi K. A Metaheuristic Algorithm Based on Chemotherapy Science : CSA. *J Optim*. 2017;2017.
5. Bakhshipour M, Ghadi MJ, Namdari F. Swarm Robotics Search & Rescue: A Novel Artificial Intelligence-Inspired Optimization Approach. *Appl Soft Comput J* [Internet]. 2017;57:708–26. Available from: <http://dx.doi.org/10.1016/j.asoc.2017.02.028>
6. Fister Jr. I, Yang X-S, Fister I, Brest J, Fister D. A Brief Review of Nature-Inspired Algorithms for Optimization. *Electrotech Rev* [Internet]. 2013;80(3):1–7. Available from: <http://arxiv.org/abs/1307.4186>
7. Mirjalili S. The Ant Lion Optimizer. *Adv Eng Softw* [Internet]. 2015;83:80–98. Available from: <http://dx.doi.org/10.1016/j.advengsoft.2015.01.010>
8. Mirjalili S. Moth-flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm. *Knowledge-Based Syst* [Internet]. 2015;89:228–49. Available from: <http://dx.doi.org/10.1016/j.knosys.2015.07.006>
9. Mirjalili S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowledge-Based Syst*. 2016;96:120–33.
10. Mandal S. Elephant Swarm Water Search Algorithm for Global Optimization. *Sadhana* [Internet]. 2018;43(1):1–21. Available from: [10.1007/s12046-017-0780-z](https://doi.org/10.1007/s12046-017-0780-z)
11. Askarzadeh A. Bird Mating Optimizer: An Optimization Algorithm Inspired by Bird Mating Strategies. *Commun Nonlinear Sci Numer Simul* [Internet]. 2014;19(4):1213–28. Available from: <http://dx.doi.org/10.1016/j.cnsns.2013.08.027>
12. Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. *Adv Eng Softw* [Internet]. 2014;69:46–61. Available from: <http://dx.doi.org/10.1016/j.advengsoft.2013.12.007>
13. Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: A Gravitational Search Algorithm. *Inf Sci (Ny)*. 2009;179(13):2232–48.
14. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp Swarm Algorithm: A Bio-inspired Optimizer for Engineering Design Problems. *Adv Eng Softw*. 2017;1–29.
15. Saremi S, Mirjalili S, Lewis A. Grasshopper Optimisation Algorithm: Theory and Application. *Adv Eng Softw*. 2017;105:30–47.
16. Dhiman G, Kumar V. Spotted Hyena Optimizer: A Novel Bio-inspired Based Metaheuristic Technique for Engineering Applications. *Adv Eng Softw* [Internet]. 2017;114:48–70. Available from: <https://doi.org/10.1016/j.advengsoft.2017.05.014>
17. Kallioras NA, Lagaros ND, Avtzis DN. Pity Beetle Algorithm – A New Metaheuristic Inspired by the Behavior of Bark Beetles. *Adv Eng Softw* [Internet]. 2018;121(April):147–66. Available from: <https://doi.org/10.1016/j.advengsoft.2018.04.007>
18. Shadravan S, Naji HR, Bardsiri VK. The Sailfish Optimizer: A Novel Nature-inspired Metaheuristic Algorithm for Solving Constrained Engineering Optimization Problems. *Eng Appl Artif Intell* [Internet]. 2019;80(January):20–34. Available from: <https://doi.org/10.1016/j.engappai.2019.01.001>
19. Akyol S, Alatas B. Plant Intelligence Based Metaheuristic Optimization Algorithms. *Artif Intell Rev*. 2016;
20. Yang X-S. Flower Pollination Algorithm for Global Optimization. In: Durand-Lose J, Jonoska N, editors. *Unconventional Computation and Natural Computation UCNC 2012 Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg; 2012. p. 240–9.
21. Merrikh-Bayat F. The Runner-root Algorithm: A Metaheuristic for Solving Unimodal and Multimodal Optimization Problems Inspired by Runners and Roots of Plants in Nature. *Appl Soft Comput* [Internet]. 2015;33:292–303. Available from: <http://dx.doi.org/10.1016/j.asoc.2015.04.048>

22. Labbi Y, Attous D Ben, Gabbar HA, Mahdad B, Zidan A. A New Rooted Tree Optimization Algorithm for Economic Dispatch with Valve-point Effect. *Int J Electr Power Energy Syst* [Internet]. 2016;79:298–311. Available from: <http://dx.doi.org/10.1016/j.ijepes.2016.01.028>
23. Zhou Y, Wang Y, Chen X, Zhang L, Wu K. A Novel Path Planning Algorithm Based on Plant Growth Mechanism. *Soft Comput*. 2017;21(2):435–45.
24. Alatas B. Sports Inspired Computational Intelligence Algorithms for Global Optimization. *Artif Intell Rev* [Internet]. 2017;1–49. Available from: <https://link.springer.com/content/pdf/10.1007%2Fs10462-017-9587-x.pdf>
25. Beheshti Z, Shamsuddin SM. A Review of Population-based Meta-Heuristic Algorithms. *Int J Adv Soft Comput Appl* [Internet]. 2013;5(1):1–35. Available from: <https://www.academia.edu/3589014>
26. Boussaid I, Lepagnot J, Siarry P. A Survey on Optimization Metaheuristics. *Inf Sci (Ny)*. 2013;237:82–117.
27. Mousavirad SJ, Ebrahimpour-Komleh H. Human Mental Search: A New Population-based Metaheuristic Optimization Algorithm. *Appl Intell*. 2017;47(3):850–87.
28. Dogan B, Olmez T. A New Metaheuristic for Numerical Function Optimization: Vortex Search Algorithm. *Inf Sci (Ny)*. 2015;293:125–45.
29. Javidy B, Hatamlou A, Mirjalili S. Ions Motion Algorithm for Solving Optimization Problems. *Appl Soft Comput* [Internet]. 2015;32:72–9. Available from: <http://dx.doi.org/10.1016/j.asoc.2015.03.035>
30. Dhiman G, Kumar V. Emperor Penguin Optimizer: A Bio-inspired Algorithm for Engineering Problems. *Knowledge-Based Syst* [Internet]. 2018;159(October 2017):20–50. Available from: <https://doi.org/10.1016/j.knosys.2018.06.001>
31. Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R. The Social Engineering Optimizer (SEO). *Eng Appl Artif Intell* [Internet]. 2018;72(January):267–93. Available from: <https://doi.org/10.1016/j.engappai.2018.04.009>
32. Cheraghalipour A, Hajiaghaei-Keshteli M, Paydar MM. Tree Growth Algorithm (TGA): A Novel Approach for Solving Optimization Problems. *Eng Appl Artif Intell* [Internet]. 2018;72(April):393–414. Available from: <https://doi.org/10.1016/j.engappai.2018.04.021>
33. Qi X, Zhu Y, Zhang H. A New Meta-heuristic Butterfly-Inspired Algorithm. *J Comput Sci* [Internet]. 2017;23:226–39. Available from: <https://doi.org/10.1016/j.jocs.2017.06.003>
34. Tabari A, Ahmad A. A New Optimization Method: Electro-Search Algorithm. *Comput Chem Eng* [Internet]. 2017;103:1–11. Available from: <http://dx.doi.org/10.1016/j.compchemeng.2017.01.046>
35. Jaddi NS, Alvankarian J, Abdullah S. Kidney-inspired algorithm for optimization problems. *Commun Nonlinear Sci Numer Simul* [Internet]. 2017;42:358–69. Available from: <http://dx.doi.org/10.1016/j.cnsns.2016.06.006>
36. Hultmann Ayala H V., dos Santos Coelho L, Reynoso-Meza G. Heuristic Kalman Algorithm for Multiobjective Optimization. In: *International Federation of Automatic Control (IFAC)* [Internet]. Elsevier B.V.; 2017. p. 4460–5. Available from: <https://doi.org/10.1016/j.ifacol.2017.08.374>
37. Fausto F, Cuevas E, Valdivia A, González A. A Global Optimization Algorithm Inspired in the Behavior of Selfish Herds. *BioSystems* [Internet]. 2017;160:39–55. Available from: <http://dx.doi.org/10.1016/j.biosystems.2017.07.010>
38. Kaveh A, Dadras A. A Novel Meta-heuristic Optimization Algorithm: Thermal Exchange Optimization. *Adv Eng Softw* [Internet]. 2017;110:69–84. Available from: <http://dx.doi.org/10.1016/j.advengsoft.2017.03.014>
39. Kaveh A, Zolghadr A. A Novel Meta-heuristic Algorithm: Tug of War. *Int J Optim Civ Eng* [Internet]. 2017;6(4):469–92. Available from: http://ijocce.iust.ac.ir/browse.php?a_id=265&sid=1&slc_lang=en
40. Shayanfar H, Gharehchopogh FS. Farmland Fertility: A New Metaheuristic Algorithm for Solving Continuous Optimization Problems. *Appl Soft Comput* [Internet]. 2018;71:728–46. Available from: <https://doi.org/10.1016/j.asoc.2018.07.033>
41. Zhang J, Xiao M, Gao L, Pan Q. Queuing Search Algorithm: A Novel Metaheuristic Algorithm for Solving Engineering Optimization Problems. *Appl Math Model* [Internet]. 2018;63:464–90. Available from: <https://doi.org/10.1016/j.apm.2018.06.036>
42. Abdul Aziz NH, Ibrahim Z, Ab Aziz NA, Mohamad MS, Watada J. Single-solution Simulated Kalman Filter Algorithm for Global Optimisation Problems. *Sadhana*. 2018;123(4):2333–5.
43. Moghdani R, Salimifard K. Volleyball Premier League Algorithm. *Appl Soft Comput J* [Internet]. 2018;64:161–85. Available from: <http://dx.doi.org/10.1016/j.asoc.2017.11.043>
44. Koohi SZ, Abdul Hamid NA, Othman M, Ibragimov G. Raccoon Optimization Algorithm. *IEEE Access*. 2019;7:5383–99.
45. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris Hawks Optimization: Algorithm and Applications. *Futur Gener Comput Syst* [Internet]. 2019; Available from: https://www.sciencedirect.com/science/article/pii/S0167739X18313530?dgcid=rss_sd_all
46. Yapici H, Cetinkaya N. A New Meta-heuristic Optimizer: Pathfinder Algorithm. *Appl Soft Comput* [Internet]. 2019; Available from: <https://www.sciencedirect.com/science/article/pii/S1568494619301309>
47. Baykasoğlu A, Hamzadayi A, Akpınar S. Single Seekers Society (SSS): Bringing Together Heuristic Optimization Algorithms for Solving Complex Problems. *Knowledge-Based Syst* [Internet]. 2019;165:53–76. Available from: <https://doi.org/10.1016/j.knosys.2018.11.016>
48. Jain M, Singh V, Rani A. A Novel Nature-inspired Algorithm for Optimization: Squirrel Search Algorithm. *Swarm Evol Comput* [Internet]. 2019;44(June 2017):148–75. Available from: <https://doi.org/10.1016/j.swevo.2018.02.013>
49. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S. Emperor Penguins Colony: A New Metaheuristic Algorithm for Optimization. *Evol Intell* [Internet]. 2019; Available from: <http://link.springer.com/10.1007/s12065-019-00212-x>
50. Pijarski P, Kacejko P. A New Metaheuristic Optimization Method: the Algorithm of the Innovative Gunner (AIG). *Eng Optim*. 2019;0273.
51. Abdullah JM, Rashid TA. Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. *IEEE*

- Access [Internet]. 2019;7:43473–86. Available from: <https://ieeexplore.ieee.org/document/8672851/>
52. Hayyolalam V, Asghar A, Kazem P. Black Widow Optimization Algorithm : A novel meta-heuristic approach for solving engineering optimization problems ☆. *Eng Appl Artif Intell* [Internet]. 2020;87:952–1976. Available from: <https://doi.org/10.1016/j.engappai.2019.103249>
53. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH. Marine Predators Algorithm : A nature-inspired metaheuristic. *Expert Syst Appl* [Internet]. 2020;152:957–4174. Available from: <https://doi.org/10.1016/j.eswa.2020.113377>
54. Kaur S, Awasthi LK, Sangal AL, Dhiman G. Tunicate Swarm Algorithm : A new bio-inspired based metaheuristic paradigm for global optimization ☆. *Eng Appl Artif Intell* [Internet]. 2020;90:952–1976. Available from: <https://doi.org/10.1016/j.engappai.2020.103541>
55. Covic N, Lacevic B. Wingsuit Flying Search — A Novel Global Optimization Algorithm. *IEEE Access*. 2020;8:53883–900.
56. Zhao W, Zhang Z, Wang L. Manta ray foraging optimization : An effective bio-inspired optimizer for engineering applications. *Eng Appl Artif Intell* [Internet]. 2020;87(103300). Available from: <https://doi.org/10.1016/j.engappai.2019.103300>
57. Lin L, Gen M. Auto-tuning strategy for evolutionary algorithms: Balancing between exploration and exploitation. *Soft Comput*. 2009;13(2):157–68.
58. Kwon WH, Kim PS, Han SH. A Receding Horizon Unbiased FIR Filter for Discrete-time State Space Models. *Automatica*. 2002;38(3):545–51.
59. Ahn CK, Zhao S, Shmaliy YS, Sakthivel R. A Revisit to Strictly Passive FIR Filtering. *IEEE Trans Circuits Syst II Express Br*. 2017;PP(99).
60. Shmaliy YS, Simon D. Iterative Unbiased FIR State Estimation: A Review of Algorithms. *EURASIP J Adv Signal Process* [Internet]. 2013;2013(1):113. Available from: <http://asp.eurasipjournals.com/content/2013/1/113>
61. Shmaliy YS, Khan S, Zhao S. Ultimate Iterative UFIR Filtering Algorithm. *Measurement*. 2016;92:236–42.
62. Shmaliy YS, Khan SH, Zhao S, Ibarra Manzano O. General Unbiased FIR Filter with Applications to GPS-Based Steering of Oscillator Frequency. *IEEE Trans Control Syst Technol*. 2017;25(3):1141–8.
63. Zhao S, Shmaliy YS, Ki Ahn C, Liu F. Self-Tuning Unbiased Finite Impulse Response Filtering Algorithm for Processes With Unknown Measurement Noise Covariance. *IEEE Trans Control Syst Technol*. 2020;(Early Acc):1–8.
64. Uribe-murcia KJ, Shmaliy YS, Ki Ahn C, Zhao S. Unbiased FIR Filtering for Time-Stamped Discretely Delayed and Missing Data. *IEEE Trans Automat Contr*. 2019;65(5):2155–62.
65. Ahn CK, Shmaliy YS. New Receding Horizon FIR Estimator for Blind Smart Sensing of Velocity via Position Measurements. *IEEE Trans Circuits Syst II*. 2018;65(1):135–9.
66. Uribe-Murcia K, Shmaliy YS, Andrade-lucio JA. UFIR Filtering for GPS-Based Tracking over WSNs with Delayed and Missing Data. *J Electr Comput Eng*. 2018;2018.
67. Lastre-Dominguez C, Shmaliy YS, Ibarra-Manzano O, Vazquez-Olguin M, Morales-Mendoza LJ. Unbiased FIR Denoising of ECG Data for Features Extraction. In: 2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2017). 2017.
68. Lastre-dom C, Shmaliy YS, Ibarra-manzano O, Vazquez-olguin M, Munoz-minjarez J. ECG Signals Denoising in State Space using UFIR Filtering for Features Extraction. In: 2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City. 2018.
69. Vazquez-Olguin M, Semenovic Shmaliy Y, Ibarra-Manzano O, Lastre-Dominguez, Carlos Morales-Mendoza LJ. Design of an Unbiased Finite Impulse Response Filter for a Smart Sensor to Estimate State of CO Concentration. In: 2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2017). 2017.
70. Shmaliy YS, Ki Ahn C. Smart Sensors Using UFIR Filtering. *IEEE Sens J*. 2017;(January).
71. Olguin VM, Shmaliy YS, Ki Ahn C, Ibarra Manzano OG. Blind Robust Estimation With Missing Data for Smart Sensors Using UFIR Filtering. *IEEE Sens J*. 2017;17(6):1819–27.
72. Ab Rahman T, Ibrahim Z, Ab Aziz NA, Zhao S, Abdul Aziz NH. Single-Agent Finite Impulse Response Optimizer for Numerical Optimization Problems. *IEEE Access*. 2018;6.
73. Ab Rahman T, Ibrahim Z, Ab. Aziz NA, Abdul Aziz NH, Khairi Mohammed S, Muhammad B, et al. A Study on the Effect of Local Neighbourhood Parameter towards the Performance of SAFIRO. *Int J Eng Technol* [Internet]. 2018;7(4.27):30–7. Available from: <https://www.sciencepubco.com/index.php/ijet/article/view/22476>
74. Ab Rahman T, Ibrahim Z, Ab. Aziz NA, Abdul Aziz NH, Mohamad MS, Shapiai MI. Evaluation of Different Horizon Lengths in Single-agent Finite Impulse Response Optimizer. In: 2019 International Conference on Computer and Information Sciences (ICCIS) [Internet]. IEEE; 2019. Available from: <https://ieeexplore.ieee.org/document/8716455>
75. Ab Rahman T, Ab. Aziz NA, Abdul Aziz NH. Single-agent Finite Impulse Response Optimizer vs Simulated Kalman Filter Optimizer. *Mekatronika*. 2019;01(02):15–22.
76. Liang JJ, Qu BY, Suganthan PN. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization [Internet]. Technical Report 201311. 2013. Available from: <http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/SharedDocuments/Forms/AllItems.aspx?RootFolder=%2Fepnsugan%2FPublicSite%2FSharedDocuments%2FCEC-2014&FolderCTID=&View=%7BD4F31868-97D8-4779-AE49-9CEC4DC3F310%7D>
77. Eberhart R, Kennedy J. A New Optimizer Using Particle Swarm Theory. In: Sixth International Symposium on Micro Machine and Human Science. 1995. p. 39–43.
78. Holland JH. Genetic Algorithms. *Sci Am* [Internet]. 1992;267(1):66–72. Available from: <http://www.nature.com/doi/10.1038/scientificamerican0792-66>
79. Shmaliy YS, Liu F, Zhao S, Khan SH. Unbiased, Optimal, and in-between: The trade-off in Discrete Finite Impulse Response Filtering. *Inst Eng Technol Signal Process*. 2016;10(4):325–34.

80. Shmaliy YS, Zhao S, Ki Ahn C. Unbiased Finite Impulse Response Filtering. IEEE Control Syst Mag. 2017;(October).
81. Ling K V., Lim KW. Receding Horizon Recursive State Estimation. IEEE Trans Automat Contr. 1999;44(9):1750–3.
82. Zhao S, Ahn CK, Shmaliy YS, Shi P, Agarwal RK. An Iterative Filter with Finite Measurements for Suddenly Maneuvering Targets. J Guid Control Dyn. 2017;40(9):2316–22.
83. Ahn CK. A New Solution to the Induced $I \infty$ Finite Impulse Response Filtering Problem based on Two Matrix Inequalities. Int J Control [Internet]. 2014;87(2):404–9. Available from: <http://www.tandfonline.com/doi/abs/10.1080/00207179.2013.836284>
84. Zhao S, Shmaliy YS, Liu F. Fast Kalman-Like Optimal Unbiased FIR Filtering with Applications. IEEE Trans Signal Process. 2016;64(9):2284–97.
85. Zhao S, Shmaliy YS, Liu F. Fast Computation of Discrete Optimal FIR Estimates in White Gaussian Noise. IEEE Signal Process Lett. 2015;22(6):718–22.
86. Zhang Q, Wang R, Yang J, Lewis A, Chiclana F, Yang S. Biology Migration Algorithm: A New Nature-inspired Heuristic Methodology for Global Optimization. Soft Comput. 2018;1–26.
87. Meshkat M, Parhizgar M. Sine Optimization Algorithm (SOA): A Novel Optimization Algorithm by Change Update Position Strategy of Search Agent in Sine Cosine Algorithm. Proc - 3rd Iran Conf Signal Process Intell Syst ICSPIS 2017. 2018;2017–Decem:11–6.
88. Haupt RL, Haupt SE. Practical Genetic Algorithms (Second Edition) [Internet]. John Wiley & Sons, Inc., Hoboken, New Jersey; 2014. Available from: <https://onlinelibrary.wiley.com/doi/book/10.1002/0471671746>
89. Suganthan PN. Particle Swarm Optimization Algorithm MATLAB code [Internet]. Technical Report, Nanyang Technological University; 2014. Available from: <http://www.ntu.edu.sg/home/epnsugan/>
90. Ab. Aziz NA, Mubin M, Ibrahim Z, Nawawi SW. Statistical Analysis for Swarm Intelligence - Simplified. Int J Futur Comput Commun. 2015;4(3):193–7.
91. Derrac J, García S, Molina D, 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 [Internet]. 2011;1(1):3–18. Available from: <http://dx.doi.org/10.1016/j.swevo.2011.02.002>
92. Van Den Bergh F, Engelbrecht AP. A Study of Particle Swarm Optimization Particle Trajectories. Inf Sci (Ny). 2006;176(8):937–71.

List of Abbreviations

CEC	Congress on evolutionary computation
DTI	Discrete time-invariant
DOF	Degree of freedom
EAs	Evolutionary algorithms
FES	Function evaluation
FIR	Finite impulse response
Fn	Function
GA	Genetic algorithm
GWO	Grey wolf optimizer
MAFIRO	Multi-agent finite impulse response optimizer
<i>maxFES</i>	Maximum function evaluation
NI	Nature-inspired
PSO	Particle swarm optimization
SI	Swarm-inspired
UFIR	Ultimate iterative unbiased finite impulse response

Declarations

Availability of data and materials

The source of dataset used in this work has been mentioned in subsection 2.3.1.

Competing Interest

The authors declare no competing interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Funding

This work was supported by the FUNDAMENTAL RESEARCH GRANT SCHEME (FRGS) awarded by the Ministry of Higher Education of Malaysia (MOHE) to Multimedia University, grant number FRGS/1/2019/ICT02/MMU/02/15.

Authors' Contributions

Tasiransurini Ab Rahman: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Resources, Data Curation, Writing-Original Draft, Visualization, Project Administration **Nor Azlina Ab. Aziz:** Conceptualization, Methodology, Software, Data Curation, Validation, Writing-Review and Editing, Supervision, Funding Acquisition **Zuwairie Ibrahim:** Conceptualization, Validation, Supervision **Nor Hidayati Abdul Aziz:** Conceptualization **Mohd Ibrahim Shapiai:** Writing-Review and Editing **Jasjit S Suri:** Writing-Review and Editing

Acknowledgment

The first author would like to thank Prof. Shunyi Zhao for teaching her the concept of the UFIR filter. Besides, authors would like to thank the anonymous reviewers for their constructive comments, as well as to MOHE, Universiti Tun Hussein Onn Malaysia, Multimedia University, Universiti Malaysia Pahang, and Universiti Teknologi Malaysia for their logistics support.

Authors' Information

Tasiransurini Ab Rahman holds a certificate and diploma in electronic engineering from Polytechnic of Sultan Haji Ahmad Shah, Pahang, Malaysia. She obtained her B.Eng. (Hons) in electrical engineering from Kolej Universiti Tun Hussein Onn, Johor, Malaysia, M.Eng. degree in electrical engineering (communication and computer network) from Universiti Kebangsaan Malaysia, Bangi, Malaysia, and her Ph.D. degree in electronic engineering from Universiti Malaysia Pahang, Malaysia. She is currently a senior lecturer in the Department of Electronic Engineering, Faculty of Electric and Electronic Engineering, Universiti Tun Hussein Onn Malaysia. She is one of the inventors of a single-agent finite

impulse response optimizer (SAFIRO), which is an optimization algorithm based on the iterative unbiased finite impulse response (UFIR) filter framework. Her current research interests include computational intelligence and its application in engineering.

Nor Azlina Ab. Aziz graduated with a B.Eng. (Hons) in electronics, majoring in computers and a M.Eng.Sc from Multimedia University, Malaysia and a Ph.D. degree from University of Malaya, Malaysia. Both her doctoral and master research works were in the field of computational intelligence. She is currently a lecturer in the Faculty of Engineering & Technology, Multimedia University, Melaka, Malaysia. She is also the chair for Multimedia University's Engineering Computational Intelligence Special Interest Group. She is one of the inventors of SAFIRO and the simulated Kalman filter (SKF) algorithm, which is an optimization algorithm based on the Kalman filter framework. Her research interests include the fundamental aspects and applications of computational intelligence in engineering.

Zuwairie Ibrahim received his B.Eng. in electrical engineering and an M.Eng. degree in image processing from Universiti Teknologi Malaysia, Johor, Malaysia, in 2000 and 2003, respectively. In 2006, he received his Ph.D. degree in DNA computing from Meiji University, Tokyo, Japan. From 2002 to 2008, he was a lecturer at Universiti Teknologi Malaysia, Johor, Malaysia. He was then promoted to senior lecturer in 2008 and served Universiti Teknologi Malaysia until 2012. In 2012, he transferred to Universiti Malaysia Pahang, Pahang, Malaysia, to work as an associate professor. He is one of the inventors of the SKF algorithm and SAFIRO. His research interests include the fundamental and applications of

computational intelligence, specifically, particle swarm optimization, ant colony optimization, gravitational search algorithm, and black hole algorithms.

Nor Hidayati Abdul Aziz (M'06 –SM'10) received the B.Eng. (Hons) in electronics engineering, majoring in computers from Multimedia University, a M.Eng. degree in electrical (electronics & telecommunications) from Universiti Teknologi Malaysia, and a Ph.D. degree in electronic engineering from Universiti Malaysia Pahang, Malaysia. From 2002 to 2006, she is a former engineer of Telekom Malaysia Berhad, Malaysia. Currently, she is a lecturer in the Faculty of Engineering & Technology, Multimedia University, Melaka, Malaysia. She is one of the inventors of the SKF algorithm. Her research

interests include fundamental and applications of computational intelligence.

Mohd Ibrahim Shapiai is a senior lecturer at Universiti Teknologi Malaysia. He received his M.Eng. from the University of York, UK in 2007 and his Ph.D. from Universiti Teknologi Malaysia in the area of machine learning in 2013. During July and August 2015, he was a visiting researcher at the Department of System Design Engineering, Keio University, Japan, for his collaborative research work with Associate Professor Yasue Mitsukura. He is also the visiting researcher at the Faculty of Design, Kyushu University, Japan, for his collaborative research work with Associate Professor Gerard Remijn in March 2017. He is now working in the area of brain-computer interface (BCI) from the perspective of deep learning. His research interests include the enhancement of spatial filtering for BCI, swarm optimization, structure learning, and advancement of deep learning to deal with the complex output. He has conducted several deep learning training studies in Kuala Lumpur since 2017. He was been invited as a speaker at HPC, Grid, Cloud & Identity (HGCI) Summit 2017 for his work on “Artificial Intelligence for Image and Signal Processing in Biomedical Applications,” which explores the advancement of deep learning for brain-computer interface technology.

Jasjit S. Suri, Ph.D., MBA, is an innovator, visionary, scientist, and an internationally known world leader in biomedical engineering and its management. Dr. Suri received the (i) Director General’s Gold medal in 1980 and is a fellow of (ii) American Institute of Medical and Biological Engineering, awarded by National Academy of Sciences, Washington, DC (2004), (iii) American Institute of Ultrasound in Medicine (2019), (iv) Asia Pacific Vascular Society (2020), and (v) recipient of the Life Time Achievement Award from Marquis (2018). He is currently the chairman of AtheroPoint, Roseville, CA, USA, which is dedicated to imaging technologies for cardiovascular and stroke. He has nearly ~17,000 citations, has co-authored 50 books, and has an H-index of 62.

Appendix A

Figure A1. The boxplots of MAFIRO, PSO, GA, and GWO for Fn1 until Fn3 (unimodal functions)

Figure A2. The boxplots of MAFIRO, PSO, GA, and GWO for Fn4 until Fn16 (simple multimodal functions)

Figure A3. The boxplots of MAFIRO, PSO, GA, and GWO for Fn17 until Fn22 (hybrid functions)

Figure A4. The boxplots of MAFIRO, PSO, GA, and GWO for Fn23 until Fn30 (composition functions)

Appendix B

Figure B1. The convergence curves of MAFIRO, PSO, GA, and GWO for Fn1 until Fn3 (unimodal functions)

Figure B2. The convergence curves of MAFIRO, PSO, GA, and GWO for Fn4 until Fn16 (simple multimodal functions)

Figure B3. The convergence curves of MAFIRO, PSO, GA, and GWO for Fn17 until Fn22 (hybrid functions)

Figure B4. The convergence curves of MAFIRO, PSO, GA, and GWO for Fn23 until Fn30 (composition functions)

Figures

Kalman Filter Strategy

Figure 1

A KF strategy to estimate the state.

UFIR Filter Strategy

Figure 2

A UFIR strategy to estimate the state (by assuming $N=6$).

Figure 3

Two tasks performed by each MAFIRO agent.

Figure 4

The principle of MAFIRO.

Figure 5

Global topology communication among MAFIRO agents.

Figure 6

The examples of agents with 3 dimensions.

Figure 7

The shrinking local neighborhood strategy in MAFIRO.

Figure 8

The plot of local neighborhood radius, δ , with different adaptive coefficient β values.

Figure 9

A graphical representation of the MAFIRO operation.

Figure 10

The boxplot of MAFIRO, PSO, GA, and GWO for (a) Fn3 (unimodal function); (b) Fn6 (simple multimodal function); (c) Fn19 (hybrid function); (d) Fn23 (composition function).

(a)

(b)

(c)

(d)

Figure 11

The convergence curve comparison for (a) Fn3 (unimodal function); (b) Fn6 (simple multimodal function); (c) Fn19 (hybrid function); (d) Fn23 (composition function).

(a)

(b)

(c)

Figure 12

The convergence curve comparison for (a) Fn10 (unimodal function); (b) Fn26 (composition function); (c) Fn28 (composition function).

(a)

(b)

(c)

(d)

(e)

(f)

Figure 13

The trajectory of a MAFIRO agent for (a) a one-dimensional Fn3 (unimodal function); (b) a two-dimensional Fn3 (unimodal function); (c) a one-dimensional Fn13 (simple multimodal function); (d) a two-dimensional Fn13 (simple multimodal function); (e) a one-dimensional Fn23 (composition function); (f) a two-dimensional Fn23 (composition function).

Figure 14

The search history of a MAFIRO agent for (a) Fn3 (unimodal function); (b) Fn13 (simple multimodal function); (c) Fn23 (composition function).

(a)

(b)

(c)

Figure 15

The fitness trend of a MAFIRO agent for (a) Fn3 (unimodal function); (b) Fn13 (simple multimodal function); (c) Fn23 (composition function).