IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2017, accepted November 1, 2017, date of publication January 23, 2018, date of current version March 13, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2777894

Single-Agent Finite Impulse Response Optimizer for Numerical Optimization Problems

TASIRANSURINI AB RAHMAN[1,2], ZUWAIRIE IBRAHIM[1], NOR AZLINA AB. AZIZ[3], SHUNYI ZHAO[4,5], (Member, IEEE), AND NOR HIDAYATI ABDUL AZIZ[3], (Senior Member, IEEE)

[1]Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, Pekan 26600, Malaysia
[2]Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat 86400, Malaysia
[3]Faculty of Engineering and Technology, Multimedia University, Melaka 75450, Malaysia
[4]Jiangnan University, Wuxi 214122, China
[5]University of Alberta, Edmonton, AB T6G 2R3, Canada

Corresponding author: Zuwairie Ibrahim (e-mail: zuwairie@ump.edu.my)

ABSTRACT This paper introduces a new single-agent metaheuristic optimization algorithm, named single-agent finite impulse response optimizer (SAFIRO). This proposed algorithm is inspired by the estimation ability of the ultimate iterative unbiased finite impulse response (UFIR) filter. The UFIR filter is one of the variants of the finite impulse response (FIR) filter, whereby in state space models, the FIR filter can be used as an option other than the Kalman filter (KF) for state estimation. Unlike the KF, the UFIR filter does not require any noise covariance, error covariance, and initial condition to calculate the state estimate. The UFIR filter also provides an iterative Kalman-like form to improve the estimation process. In the SAFIRO algorithm, the agent works as an individual UFIR to find an optimal or a near-optimal solution, where the agent needs to perform two main tasks; measurement and estimation. The performance of the SAFIRO algorithm is evaluated using the CEC 2014 Benchmark Test Suite for single-objective optimization and statistically compared with the several well-known metaheuristic optimization algorithms, such as Particle Swarm Optimization algorithm, Genetic Algorithm, and Grey Wolf Optimization algorithm. The experimental results show that the proposed SAFIRO algorithm is able to converge to the optimal and the near-optimal solutions, and significantly outperform all the aforementioned state-of-the-art metaheuristic algorithms.

INDEX TERMS Optimization, metaheuristics, single-agent, FIR, local search neighbourhood.

I. INTRODUCTION

Optimization is seen in many fields such as engineering, social science, economics, and business. It is a process of achieving an optimal solution to the problem. The optimal solution can be either a minimum or a maximum solution. In general, optimization methods can be divided into exact methods and approximate methods [1]. The exact methods may not be suitable for some complex optimization problems. Thus, approximate methods are the other option to solve these problems. Approximation algorithms and heuristic algorithms are subcomponents of approximate methods. Heuristic algorithms can be further classified into two classes: problem-specific heuristics and metaheuristics [1]. Problem-specific heuristics are problems-dependent algorithms whereas metaheuristics are more general algorithms and can be used to solve various types of optimization problems with minimum modification.

Metaheuristic algorithms have gained huge popularity and attracted researcher's attention because of its flexibility and ability in solving large scale and variety of optimization problems [2], [3]. These algorithms have iterative and stochastic behaviour. Some literature classified the metaheuristic algorithms into two categories based on the number of agents: single-agent and multi-agent [4]–[6].

Single-agent metaheuristics are more exploitation oriented which intensify the search in local regions, whereas multi-agent metaheuristics are more exploration oriented which allow diversification in the whole search space [5].

In single-agent (also known as trajectory-based) metaheuristics, the search for an optimal or near-optimal solution starts with a single initial solution by an agent. Then, this agent moves away from the initial and creates a search path in the search space [5]. A single solution is updated iteratively until a stopping condition is met. The prominent examples of

single-agent metaheuristic algorithms are shown in Table 1. Tabu Search (TS) algorithm [7], [8] introduced by Glover in 1986 depends on searching neighbouring solutions and local memory. Vortex Search (VS) algorithm [9] and Mean-Variance Mapping Optimization (MVMO) algorithm [10] are examples of modern single-agent metaheuristic algorithms. VS algorithm developed by Dogan and Olmez in 2013 is inspired by the vortex flow of stirred fluids where the agent uses an adaptive step size adjustment scheme as its search behaviour.

Different from single-agent, multi-agent metaheuristics employ a set of agents to search an optimal or a near-optimal solution at each stage [5]. Basically, the process in multi-agent algorithms starts with the initialization of the population. Then, the solutions of this initial population are evaluated. After that, the new population of candidate solutions are iteratively generated to replace the current population where these solutions will also be evaluated before the search process begins. The iteration stops once the stopping criterion is satisfied [1]. A few of familiar multi-agent metaheuristic algorithms are as listed in Table 1. Particle Swarm Optimizer (PSO) algorithm [11], [12] pioneered by Eberhart and Kennedy in 1995 mimics the behaviour of organisms, such as bird flocking. Genetic Algorithm (GA) [13], [14] developed by Holland in 1975, is inspired by the biological evolution theory, which consists of selection, crossover, and mutation steps. Grey Wolf Optimizer (GWO) [15] created by Mirjalili *et al.* in 2013 is inspired by the leadership hierarchy and hunting style of grey wolves.

TABLE 1. Metaheuristic classification and examples.

Classification	Algorithm
Single-agent	TS, Simulated Annealing (SA) [24], [25], Variable Neighborhood Search (VNS) [26], [27], VS, and MVMO.
Multi-agent	Evolutionary Algorithms (EAs) [28], GA, PSO, Ant Colony Optimization (ACO) [29], [30], Artificial Bee Colony (ABC) [31], [32], and GWO.
Bio-inspired SI based	PSO, ACO, ABC, Firefly Algorithm (FA) [33], and GOA.
Bio-inspired non-SI based	GA and KA.
Physics or chemistry-based	Gravitational Search Algorithm (GSA) [34], Black Hole (BH) [35], EFO, and IGMM.
Non-nature inspired based	SA, TS, VNS, MVMO, and VS.

From another perspective, Fister *et al.* [16] classified metaheuristic algorithms into four source of inspiration which are bio-inspired swarm intelligence (SI) based, bio-inspired non-SI based, physics or chemistry-based, and non-nature inspired based. Examples of the algorithms for these categories are shown in Table 1. Most of the proposed metaheuristic algorithms in the literature are nature-inspired [6], [17]. Grasshopper Optimization Algorithm (GOA) [18] and Kidney Algorithm (KA) [19] are among new algorithms that are categorized as bio-inspired algorithm as they are inspired by the behaviour of grasshopper and kidney process in the human body, respectively. Electromagnetic Field Optimization (EFO) algorithm [20] and Ideal Gas Molecular Movement (IGMM) algorithm [21] on the other hand are categorized as physics or chemistry-inspired algorithms because they are inspired by the behaviour of electromagnets attraction-repulsion force and movement-collision of gas molecules, respectively.

The non-nature inspired algorithms are not very popular and lacking in number compared to nature-inspired algorithms. Hence, it is always interesting and beneficial to discover the source of inspiration by looking away from nature. As stated by Wolpert and Macready in [22], there is no optimization algorithm that is better than other algorithms in solving all optimization problems. Thus, there is room for exploring and developing new and effective optimization algorithms for solving several types of problems. However, the challenge in this field is how to get a good source of inspiration from the existing knowledge either to improve the existing algorithms or develop a new algorithm [20].

The aim of applying metaheuristic algorithm is to estimate a near-optimal solution for an optimization problem. In some cases, the metaheuristic algorithm is capable to estimate an optimal solution. This estimation is done by an agent during solution search process. In metaheuristic algorithm, the agent is a problem solver or also known as an optimizer that is responsible to find and estimate an optimal or a near-optimal solution. This condition is identical to the concept of estimation in state space model, whereby the estimator is used to optimally estimate the state. Hence, the inspirational source can also be triggered by the concept of estimation in state space model. In state space models, there are two types of estimators for state estimation: infinite impulse response (IIR) filter including Kalman filter (KF) and finite impulse response (FIR) filter [23]. Both are used as state estimator and employ the mathematical state space model of the system and state measurements to estimate the state.

KF estimator is able to give an optimal estimation in an ideal situation where the noise covariance, the error covariance and the initial value are known and need to be traced starting from the zero-time index. Without this information, KF may produce error where this error will project to the next time index of estimation process due to its infinite impulse response structure. If this happens, KF will be unable to give the best estimation for this particular situation. As an option, FIR filter is used to estimate the state with its finite impulse response structure and is capable to give an acceptable of a near-optimal estimation for the state. Although FIR filter can only provide a near-optimal and not an optimal estimation, this filter is preferred compared to KF due to its robustness and stability in its structures [36], [37]. FIR structural stability attracts researchers to explore, modify and improve its structures over time. Since its introduction, many FIR variants have emerged. One of these variants is the ultimate iterative unbiased finite impulse response (UFIR) filter. This UFIR filter has advantages over its previous basic structure. Besides completely ignores noise statistics, error covariance

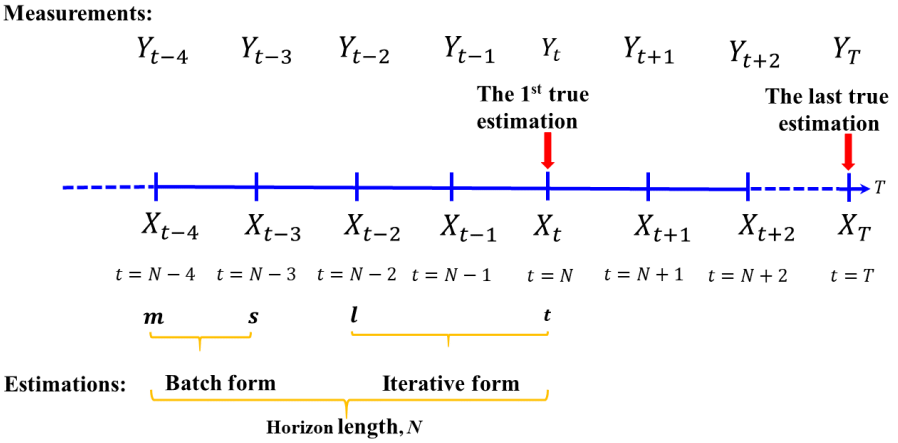


FIGURE 1. The estimation process of UFIR filter for the first true estimation (by assuming $N = 5$).

and initial value to calculate the state estimate, UFIR filter also comes with fast iterative Kalman-like in a simpler form to improve the estimation process which can universally be used for systems with or without the control inputs [38], [39].

Therefore, in this paper, a new metaheuristic optimization algorithm inspired by the estimation ability of UFIR filter, called Single-agent Finite Impulse Response Optimizer (SAFIRO) is proposed. The SAFIRO algorithm works with only one agent to find the best solution in solving a numerical optimization problem. Experimental results show that the proposed SAFIRO algorithm managed to significantly outperform PSO algorithm, GA algorithm, and GWO algorithm.

This paper is arranged as follows: after the introduction, section II introduces the FIR and UFIR filter, whereas the proposed SAFIRO algorithm is presented in section III. Next, section IV explains the experimental procedure done to evaluate the performance of SAFIRO, followed by results and discussion in section V. Section VI concludes the paper and finally, section V exposes the future works.

II. FINITE IMPULSE RESPONSE FILTER

Estimation problems for numerous engineering application can be represented in state space with general Discrete Time-Invariant (DTI) linear model [40], as shown in (1) and (2)

$$\boldsymbol{x}_n = \boldsymbol{A}\boldsymbol{x}_{n-1} + \boldsymbol{B}\boldsymbol{w}_n \tag{1}$$

$$\boldsymbol{y}_n = \boldsymbol{C}\boldsymbol{x}_n + \boldsymbol{D}\boldsymbol{v}_n \tag{2}$$

where n is the discrete time index; J is the number of the state; $\boldsymbol{x}_n$ is the $J \times 1$ system state vector (representing the system's variables of interest such as position, velocity or acceleration); $\boldsymbol{y}_n$ is the $M \times 1$ measurement vector (represents measurement observation); $\boldsymbol{A}$ is the $J \times J$ state transition matrix (that projects the previous state, $\boldsymbol{x}_{n-1}$ to the present state, $\boldsymbol{x}_n$); $\boldsymbol{C}$ is the $M \times J$ measurement transition matrix (that projects the measurements onto the state vector variables); $\boldsymbol{w}_n$ is a $J \times 1$ system noise vector; $\boldsymbol{v}_n$ is a $M \times 1$ measurement noise vector; $\boldsymbol{B}$ is the $J \times J$ process noise matrix; and $\boldsymbol{D}$ is the $M \times M$ measurement noise matrix.

As aforementioned, FIR filter and KF are generally considered as two different types of state estimators. FIR filter is proposed by Jazwinski in 1968 as another option to KF whereby FIR filter has finite impulse response structure, to provide better robustness and stability. The idea in FIR filter is to estimate the state vector based on a finite number of recent measurement [41], [42]. In contrast to KF, FIR filter does not require the initial condition and noise statistics. Unlike KF that projects an estimation from one point to another, FIR filter uses a finite number of inputs on the most recent time interval called as the horizon length, N. The accuracy of UFIR depends on the average of N, which must be optimal [43].

FIR filter has been improved significantly from its basic principle. For example, a receding horizon FIR filter was composed by Ahn [44] and fast iterative forms for FIR filter was introduced by Zhao *et al.* [45], [46]. Recently, Shmaliy *et al.* have developed the UFIR filter to provide a fast near-optimal estimation in a simple form [38]. This filter works with two sets of mathematical equations. The first set of equations is known as batch form equations whereas the second set of equations is known as iterative form equations. The former is defined to generate the initial value of the state estimate whereas the latter is applied for fast computation of the state estimation. UFIR is claimed in [38] and [43] as the most robust among the FIR variants.

The estimation process in UFIR is executed in a finite length according to its N. As illustrated in Fig. 1, the range of state estimations of UFIR filter started from the time index, $t = N$ until the maximum time index, T. To get the first state estimation value (known as a true estimate) at $t = N$, UFIR needs N most recent measurements (which starts from point m until t).

The estimation process in UFIR begins with the generation of the initial state that is available from the batch form (at point $m = t - N + 1$ and $s = m + 1$) and iteratively updates the state estimation value (from point $l = s + 1$ until $l = t$).

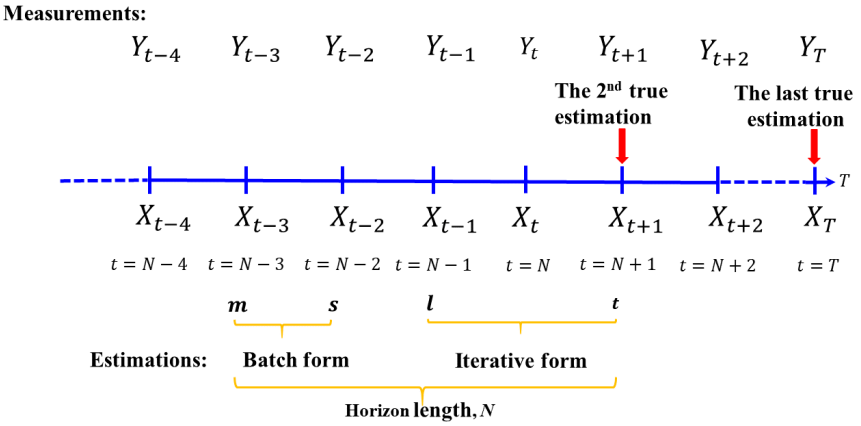


FIGURE 2. The estimation process of UFIR filter for the second true estimation (by assuming $N = 5$).

The batch form can be computed in the discrete convolution-based applied to measurement as shown in (3). In state space, $\overline{\boldsymbol{H}}_{m,s}$ represents gain or coefficient of the filter impulse response and can be calculated as in (4). The $\boldsymbol{Y}_{m,s}$ represents the measurement value of point m and s, whereas $\boldsymbol{C}_{m,s}$ is the measurement matrix from m to s. UFIR handles all the measurements within the batch form points at a time. Thus, the initial value is generated at point s by a convolution process between the gain, $\overline{\boldsymbol{H}}_{m,s}$ and the measurement, $\boldsymbol{Y}_{m,s}$.

$$\bar{\boldsymbol{x}}_s = \overline{\boldsymbol{H}}_{m,s}\boldsymbol{Y}_{m,s} \tag{3}$$

$$\overline{\boldsymbol{H}}_{m,s} = (\boldsymbol{C}_{m,s}^T\boldsymbol{C}_{m,s})^{-1}\boldsymbol{C}_{m,s}^T \tag{4}$$

On the other hand, the iterative form can be calculated as shown in (5). $\bar{\boldsymbol{x}}_l$ represents the estimated state vector at present point, whereas $\bar{\boldsymbol{x}}_{l-1}$ is the estimated state vector at the nearest previous point. $\boldsymbol{y}_l$ represents the value of measurement at present point and $\boldsymbol{C}$ is a measurement matrix. $\boldsymbol{K}_l$ is called as Kalman-like correction gain and can be computed as in (6). K_{l-1} is the gain for the nearest previous point, whereas $\boldsymbol{A}$ is a transition matrix of the iterative form.

As the number of iterative points is increased, the state estimation value is improved by this Kalman-like gain.

$$\bar{\boldsymbol{x}}_l = \boldsymbol{A}\bar{\boldsymbol{x}}_{l-1} + \boldsymbol{K}_l\boldsymbol{C}^T(\boldsymbol{y}_l - \boldsymbol{C}\boldsymbol{A}\bar{\boldsymbol{x}}_{l-1}) \tag{5}$$

$$\boldsymbol{K}_l = [\boldsymbol{C}^T\boldsymbol{C} + (\boldsymbol{A}\boldsymbol{K}_{l-1}\boldsymbol{A}^T)^{-1}]^{-1} \tag{6}$$

The true state estimate is taken when an iterative variable, l reaches the present time index, t. The same procedures are repeated to get the second true estimate at $t = N + 1$ as illustrated in Fig. 2.

The similar procedures are repeated until the last true estimation is reached at $t = T$, indicating that all the horizon points have been estimated. As a matter of fact, there is no relationship between the first, second until the last true estimation because the estimation process is done in a finite form (based on the horizon length). This makes the FIR filter more robust compared with IIR filter whereby in the case that there is an error in previous FIR estimation, the error is not projected to the next FIR estimation process. In addition, the noise statistics are not required by this procedure in performing the estimation process.

TABLE 2. Analogy of UFIR filter with SAFIRO algorithm.

UFIR filter	SAFIRO algorithm
Act as an estimator to provide a near-optimal estimation for the state.	Act as an optimizer (an agent) of metaheuristic algorithm to find an optimal or a near-optimal solution for an optimization problem. The search strategy is inspired by the UFIR's framework.
Several state space models involved which represent several parameters such as position, acceleration, and velocity.	Only one state space is considered, representing the position that holds the estimated solution of a problem.
Measurements value are taken from the sensor.	Measurements are simulated from random mutation of *X_best_so_far* and local neighbourhood method.
The initial estimation value is generated from the batch form which is obtained from discrete convolution-based.	The initial estimation value is generated by using a uniformly distributed random generator in the range of [0,1].

Table 2 shows the analogy between the UFIR filter with the SAFIRO algorithm. In SAFIRO algorithm, UFIR filter's framework serves as an inspiration for the optimizer. Specifically, the static model of DTI FIR is employed in modelling the SAFIRO algorithm because the optimal solution to be estimated is time independent. Therefore, the state vector in (5) becomes a scalar which consists of only one variable that holds an agent's estimated position in the search space.

Different from UFIR filter that estimates the state using the measurements which can be obtained from the sensor, the agent of SAFIRO estimates an optimal or a near-optimal solution by using simulated measurements from random mutation and local neighbourhood method. The initial estimation value for UFIR filter is generated from the batch form whereas the initial value in SAFIRO algorithm is generated randomly in the search space. As a result, UFIR filter gives a near-optimal estimation for the state, whereas SAFIRO

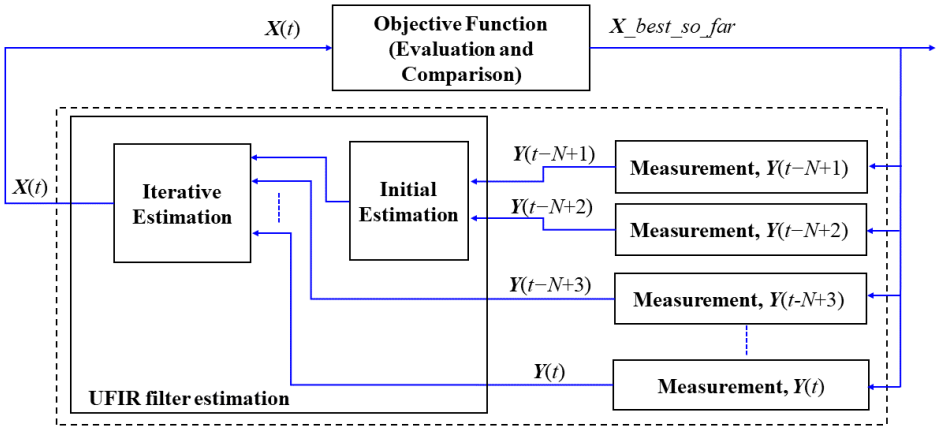


FIGURE 3. The principle of SAFIRO algorithm.

provides an estimation to an optimal or a near-optimal solution. Details of the process are explained in the following section III.

III. SINGLE-AGENT FINITE IMPULSE RESPONSE OPTIMIZER ALGORITHM

A. STRATEGY OF SAFIRO ALGORITHM

An agent plays an important role to find an optimal solution in solving the optimization problem. The agent in SAFIRO works as an individual UFIR to optimally improve its estimation position. This estimation position represents a solution for the given optimization problem. In order to find an optimal solution, SAFIRO agent needs to perform two main tasks: measurement and estimation. Figure 3 shows how the UFIR works as an optimizer for iteration, t.

$\boldsymbol{X}\_best\_so\_far$ holds the value of the best-so-far solution. $\boldsymbol{Y}(t-N+1), \ldots, \boldsymbol{Y}(t)$ are the measured positions of an agent that represents the measured solution for the optimization problem. The measurement position is simulated in SAFIRO by using a random mutation of $\boldsymbol{X}\_best\_so\_far$ together with a local neighbourhood method. A uniformly distributed random number in the range of [0, 1] is applied in this work. A new measurement solution, $\boldsymbol{Y}(t)$ is produced for each new iteration.

The estimation phase in SAFIRO is divided into two stages: initial estimation and iterative estimation. $\boldsymbol{X}(t)$ is the estimated position that represents the solution of the given optimization problem for each iteration, t. Quality of the $\boldsymbol{X}(t)$ is then evaluated using the objective function and compared to $\boldsymbol{X}\_best\_so\_far$. If $\boldsymbol{X}(t)$ is found to be a better solution than $\boldsymbol{X}\_best\_so\_far$, then $\boldsymbol{X}\_best\_so\_far$ will be updated with $\boldsymbol{X}(t)$ value. The process is repeated until the maximum iteration is reached. Details of each phase of SAFIRO algorithm are discussed in the following subsection.

B. PROCEDURE OF SAFIRO ALGORITHM

The overall process of SAFIRO algorithm is divided into five main phases: initialization, measurement, estimation, fitness evaluation, and update $\boldsymbol{X}\_best\_so\_far$, as depicted in Fig. 4.

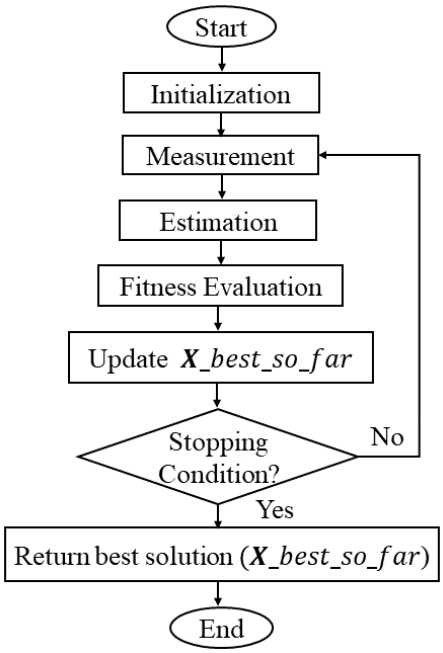


FIGURE 4. The flowchart of SAFIRO algorithm.

1) INITIALIZATION PHASE

In the initialization phase (at iteration, $t = 0$), the parameter of N is defined. Like the FIR filter, SAFIRO needs N most recent measurements to begin the estimation step. Further explanation about parameter tuning of N is explained at result and discussion part. Based on the experiment that has been carried out, the optimal value of N for SAFIRO is equal to 4. Therefore, four random initial values are generated for $\boldsymbol{Y}(0)$, $\boldsymbol{Y}(t-1)$, $\boldsymbol{Y}(t-2)$ and $\boldsymbol{Y}(t-3)$. Equation (7) shows the equation to generate the random initial value for $\boldsymbol{Y}(0)$. The same equation is used to generate $\boldsymbol{Y}(t-1)$, $\boldsymbol{Y}(t-2)$ and

$\boldsymbol{Y}(t-3)$. X_{min} is the lower limit whereas X_{max} is the upper limit of the search space.

$$\boldsymbol{Y}(0) = rand\ (U\ [X_{min}, X_{max}]) \tag{7}$$

The goal of using random values is to efficiently explore the search space to find a near-optimal solution. Then, the fitness of these random initial values is evaluated to determine the initial $\boldsymbol{X}\_best\_so\_far(0)$. For a minimization problem, the initial measurement which has the smallest fitness value is assigned as $\boldsymbol{X}\_best\_so\_far$, whereas for a maximization problem, the initial measurement which has the largest fitness value is assigned as $\boldsymbol{X}\_best\_so\_far$.

Besides the initial measurements, the maximum number of iterations, T is also defined at this phase.

2) MEASUREMENT PHASE

As mentioned in section II, in a real operation of FIR filter, the measurement readings can be taken from the sensor. But in SAFIRO algorithm, measurement is simulated using $\boldsymbol{X}\_best\_so\_far$. The measurements are simulated by using random mutation of $\boldsymbol{X}\_best\_so\_far$ and local neighbourhood method. In the proposed method, each dimension of the problem to be optimized is associated with a random value ranging from 0 to 1. The dimensions that have a random value greater than 0.5 are selected to be mutated to produce a new candidate solution. The mutation is conducted in a local neighbourhood of $\boldsymbol{X}\_best\_so\_far$.

The pseudocode for measurement phase is given as in Pseudocode 1, whereas the equation for measurement is given in (8).

Pseudocode 1 Measurement

```
for each dimension, d
  if the dimension of an agent is selected for mutation
      compute measurement value using (8).
  else
      compute measurement value using (10).
  end
end
```

Here, a shrinking local neighbourhood method is used. This concept is visualized in Fig. 5.

The local search process scales down the search area where the search is centred around $\boldsymbol{X}\_best\_so\_far$. Equation (9) gives the radius of the local neighbourhood, δ, where, t is the number of current iteration; T is the number of maximum iteration; X_{max} is the upper limit of search space and β is the adaptive coefficient value. Adaptive coefficient value is employed to control how fast the size of the neighborhood will be reduced. In this paper, the $\beta = 10$ is selected for SAFIRO algorithm as recommended by [47]. Figure 6 shows the plots for exponential term of the delta, δ with different values of β. The bigger value of β leads to a faster transition from exploration to exploitation.

The remaining dimensions that are not selected for mutation process, will hold the value of the $\boldsymbol{X}\_best\_so\_far$ as their

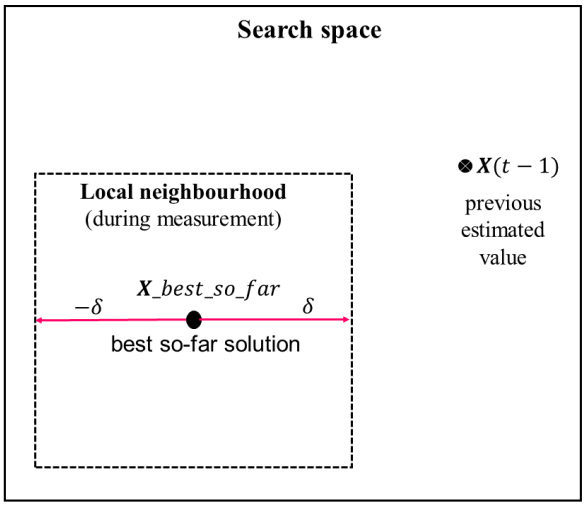


FIGURE 5. A local neighbourhood's strategy in SAFIRO algorithm.

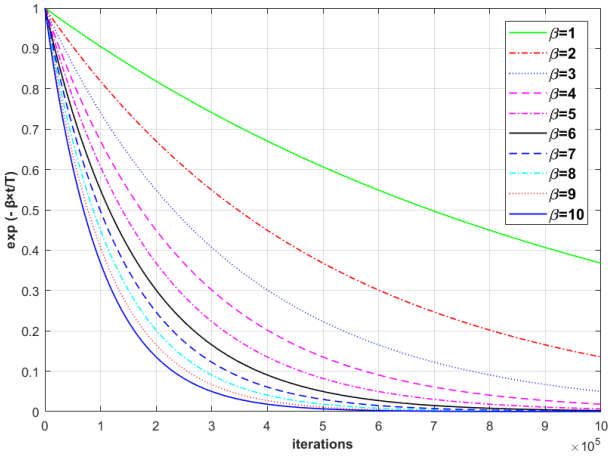


FIGURE 6. The plot of δ with different β values.

measurement value, as given in (10).

$$Y_d\ (t) = \boldsymbol{X}\_best\_so\_far_d(t-1) + rand(\mathrm{U}\ [-\delta, \delta]) \tag{8}$$

$$\delta = e^{-\beta \times \frac{t}{T}} \times \frac{X_{max} - X_{min}}{2} \tag{9}$$

$$Y_d\ (t) = \boldsymbol{X}\_best\_so\_far_d(t-1) \tag{10}$$

3) ESTIMATION PHASE

The next phase is the estimation phase which updates the position of an agent. The agent estimates the solution in a finite length according to N.

As illustrated in Fig. 7, the agent estimates the position for each iteration, t in a finite length according to N. In each t, the estimation is done by sub-iteration, k.

Unlike the real UFIR filter that computes an initial estimation value by using the convolution-based method, the initial estimation value, $\hat{\boldsymbol{X}}(k = 2)$ in SAFIRO algorithm is generated randomly between [lower limit, upper limit] of the first two points in sub-iteration, as shown in Pseudocode 2.

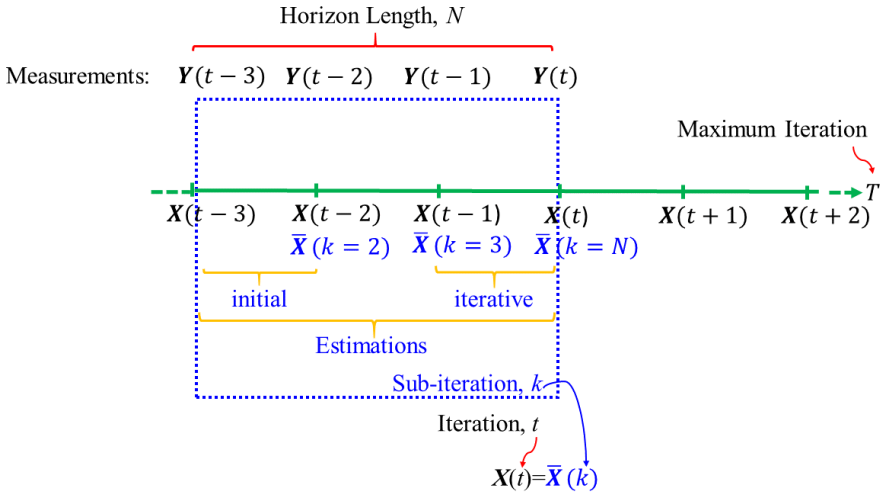


FIGURE 7. A graphical view of the operation in SAFIRO algorithm (by assuming $N = 4$).

Pseudocode 2 Generation of Initial Estimation, $\bar{\boldsymbol{X}}(2)$ at $k = 2$

```
if   Y(t − N + 1) < Y(t − N + 2)
     X̄(2) = rand(U[Y(t − N + 1), Y(t − N + 2)])
else
     X̄(2) = rand(U[Y(t − N + 2), Y(t − N + 1)])
```

Then, the solution of this initial estimation is improved iteratively by (11), started at $k = 3$ until $k = N$. The improvement is influenced by the measurement value, $\boldsymbol{Y}(t - N + k)$ and the Kalman-like gain, $K(k)$. $\bar{\boldsymbol{X}}(k)$ is the estimated solution for present point, whereas $\bar{\boldsymbol{X}}(k - 1)$ is the most recent sub-iteration point. As the state vector is reduced to a scalar that contains only one variable (the estimated position value), therefore the state transition matrix, $\boldsymbol{A}$, in (5) and (6) becomes 1. The measurement transition matrix, $\boldsymbol{C}$ in (5) and (6) is assigned to be 1, indicates the same scale of measurement and scale of state estimate, as in [39]. Therefore, the equation of $\bar{\boldsymbol{X}}(k)$ can be simplified as shown in (11). $(\boldsymbol{Y}(t - N + k) - \boldsymbol{X}(k - 1))$ element represents a correction based on the $\boldsymbol{Y}(t-N+k)$ measurement while the Kalman-like gain, $K(k)$ adjusts the result. As stated in [48], the total value of gain along the horizon length is equal to 1, and the value is distributed uniformly. Thus, in SAFIRO, the value of gain for each sub-iteration point is equal to $\frac{1}{k}$ as in (12). As the sub-iteration, k increases, the value of $K(k)$ gets smaller and the estimation improved.

$$\bar{\boldsymbol{X}}(k) = \bar{\boldsymbol{X}}(k - 1) + K(k)(Y(t - N + k) - \bar{\boldsymbol{X}}(k-1)) \quad (11)$$

$$K(k) = \frac{1}{k} \quad (12)$$

The sub-iteration stage is stopped when $k = N$. Then, the final value of k is assigned as the estimation of iteration $(X(t) = \bar{\boldsymbol{X}}(k))$. The estimated value of $\boldsymbol{X}(t)$ represents the updated solution by an agent for that particular iteration.

4) FITNESS EVALUATION AND *X_best_so_far* UPDATE

The evaluation step evaluates the fitness level of the agent. The fitness level is measured according to the objective function (or also known as the fitness function). The objective function is the function of the given optimization problem that needs to be solved by the optimization algorithm.

The fitness of the estimated solution, $X(t)$ is compared to the fitness of *X_best_so_far* whereby *X_best_so_far* will be updated if a better solution is found. For minimization problem, *X_best_so_far* is updated when $\text{fit}(\boldsymbol{X}(t)) < \text{fit}(\boldsymbol{X}\_best\_so\_far(t))$ whereas for maximization problem, *X_best_so_far* is updated when $\text{fit}(\boldsymbol{X}(t)) > \text{fit}(\boldsymbol{X}\_best\_so\_far(t))$.

The process of measurement and estimation are repeated until the maximum iteration,T is reached. Once the maximum iteration is reached, the *X_best_so_far* is return as the solution to the given problem.

The whole procedure of SAFIRO algorithm for minimization problem is given in Pseudocode 3.

IV. EXPERIMENTAL SETUP

The SAFIRO algorithm is developed from scratch by using MATLAB software. The performance of the proposed algorithm is evaluated by solving a set of problems on the CEC 2014 Benchmark Test Suite [49] which contains 30 single objective test functions that represent real optimization problems. These test functions are divided into four different groups: unimodal functions, simple multimodal functions, hybrid functions and composition functions, as can be seen in Table 3. All the functions are minimization problems and treated as black-box problems. The details of these functions are specified in [49]. The MATLAB codes for CEC 2014 benchmark suite can be downloaded from http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2014.

Pseudocode 3 Procedure in SAFIRO Algorithm

Algorithm: SAFIRO algorithm for a minimization problem.
Requirement: horizon length, N
01: Initialization phase
02: **while** not maximum iteration **do**
03: Procedure **MEASUREMENT**
04: Pseudocode 1
05: end Procedure
06: Procedure **ESTIMATION**
07: at sub-iteration, $k = 2$
08: Pseudocode 2
09: **for** $k = 3$: N
10: Iteration of estimation: as in (11) and (12)
11: **end**
12: $\boldsymbol{X}(t) \leftarrow \bar{\boldsymbol{X}}(k)$
13: end Procedure
14: Evaluate fitness for agent
15: $t \leftarrow t + 1$
16: end **while**
17: Return $\boldsymbol{X}\_best\_so\_far$

The parameter that needs to be set during initialization phase in SAFIRO is the N. The optimal value of N is equal to 4. The initial measurement for an agent is set to a random value following the search space $[-100, 100]$. For benchmarking of the results, three metaheuristic algorithms were applied.

GA and PSO algorithms were chosen as both are very well-known and established algorithms in the evolutionary computation category and swarm intelligence category, respectively. Additionally, GWO algorithm is selected to represent modern metaheuristic algorithms. The original paper of GWO algorithms has more than 780 citations since it is introduced in 2013.

Regardless the number of agent, fair comparison of algorithm performance can be done by setting the same number of fitness evaluation. In this comparison, the number of fitness evaluation for all algorithms is set to 500,000 while the problem dimension is set to 50. The stopping condition is set to be the maximum number of iterations for all algorithms. The evaluation is based on the average performance over 51 run times on each test problem.

The Friedman test for multiple algorithms comparison is then conducted to compare the results for all algorithms. Friedman test is chosen because it is suitable for non-parametric test, as the solutions yielded in this experiment is not normally distributed. Friedman test defines the null hypothesis as all tested algorithms are equal one to another, with no significant differences [50]. In Friedman test, the performances of all four algorithms were ranked statistically based on their mean fitness. After that, the significant differences are observed.

The post hoc analysis using Holm's method is applied as recommended by Derrac *et al.* [51], in order to characterize the significant differences (between the algorithms' performance), as detected by the Friedman test. Holm's method rejects the null hypothesis if the statistical value is smaller than the unadjusted p-value. The value of the significant level, α is selected to indicate the significant difference of the algorithms. Usually, the preferred value is either 0.01, 0.05 or 0.1. The smaller the value, the more rigid to determine the significant difference. In contrast, the larger the value,

TABLE 3. The CEC 2014 Benchmark Test Suite. (source: [49].

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 easier statistically to claim the significant difference [50]. Therefore, $\alpha = 0.05$ is chosen to balance the opportunity to get the significant difference with the rigid procedure in detecting the significant difference.

In this work, KEEL Software Tool is used as a platform to execute both Friedman and Holm Post Hoc tests. The KEEL Software can be downloaded from http://www.keel.es.

V. RESULTS AND DISCUSSION

In this section, the results of the proposed SAFIRO algorithm are discussed and analyzed.

A. PARAMETER TUNING

As previously stated, N is the parameter needs to be determined in SAFIRO. Simulation experiments have been performed to identify the optimal value of N. The value of $N = 4$ until $N = 10$ are selected to be set in SAFIRO. With these values, SAFIRO is tested to solve the optimization problems in the CEC 2014 Benchmark Test Suite. The $N = 4$ is chosen as the minimum value because the first two points of the horizon length are used to randomly generate the initial estimates, whereas the third point is used as a starting point of the iterative estimation part. Thus, practically, the minimum point for the horizon length is equal to 4.

The Friedman test is then applied to rank the performance, followed by the Holm Post Hoc test to determine which N is better in solving the optimization problems of CEC 2014. As shown in Table 4 and Table 5, SAFIRO has an equivalent performance for $N = 4$ and $N = 5$. Both are significantly better than $N = 6$ until $N = 10$. Hence, even though $N = 5$ has a higher ranking, $N = 4$ is chosen to minimize the computational time. Further explanation of Friedman test and Holm Post Hoc test are explained in the statistical analysis section.

TABLE 4. Average ranking of N.

N	Ranking
4	2.9667
5	2.75
6	3.8333
7	3.95
8	4.55
9	4.8833
10	5.0667

B. STATISTICAL ANALYSIS

The performance of SAFIRO algorithm is evaluated by solving 30 benchmark functions as available in CEC 2014 Benchmark Test Suite. The mean and standard deviation values of SAFIRO, PSO, GA, and GWO for each benchmark function are recorded in Table 6. Result in bold font represents the best fitness or solution for each function (Fn).

TABLE 5. Holm post HOC result of N value for $\alpha = 0.05$

N	p	Holm
5 vs 10	0.000033	0.002381
5 vs 9	0.000131	0.0025
4 vs 10	0.000167	0.002632
4 vs 9	0.00059	0.002778
5 vs 8	0.00125	0.002941
4 vs 8	0.00453	0.003125
6 vs 10	0.027024	0.003333
5 vs 7	0.031444	0.003571
7 vs 10	0.045284	0.003846
5 vs 6	0.052107	0.004167
6 vs 9	0.05977	0.004545
4 vs 7	0.077907	0.005
7 vs 9	0.094264	0.005556
4 vs 6	0.120233	0.00625
6 vs 8	0.198837	0.007143
7 vs 8	0.282059	0.008333
8 vs 10	0.354289	0.01
8 vs 9	0.550097	0.0125
4 vs 5	0.697684	0.016667
9 vs 10	0.742392	0.025
6 vs 7	0.834319	0.05

1) UNIMODAL FUNCTIONS (Fn1 TO Fn3)

As can be seen in Table 6, SAFIRO clearly shows a superior performance by producing the best solution compared to other algorithms in all unimodal functions. Unimodal functions are related to the rotation problems. According to Mirjalili in [15], unimodal functions are correlated with exploitation benchmarking. Thus, the results for unimodal functions indicate that SAFIRO has an excellent performance in exploiting the optimal solution especially in Fn3 (Rotated Discus function), where SAFIRO was able to converge to the ideal fitness of 300 (optimal solution). SAFIRO also managed to obtain the best solution in solving Fn1 (Rotated High Conditioned Elliptic function) and Fn2 (Rotated Bent Cigar function) although the problems are difficult to be solved, specifically for Fn1 because it involved a quadratic ill-conditioned property [49].

2) SIMPLE MULTIMODAL FUNCTIONS (Fn4 TO Fn16)

SAFIRO algorithm also shows excellent performance in solving simple multimodal functions by leading in 7 out of 13 functions. Most of the simple multimodal functions are related to shifting and rotation problems. These functions are suitable for exploration benchmarking of an algorithm [15]. Hence, these results proved that apart from being great in exploitation, SAFIRO is also very good in exploration especially in Fn7 (Shifted and Rotated Griewank's

TABLE 6. The mean fitness and standard deviation values obtained by SAFIRO, PSO, GA and GWO algorithms. Numbers in bold indicate the best fitness.

Function	Fn	SAFIRO		PSO		GA		GWO	
		Mean	Std Dev.	Mean	Std Dev.	Mean	Std Dev.	Mean	Std Dev.
Unimodal	1	**8.0E+05**	3.1E+05	7.0E+07	5.4E+07	1.21E+08	4.0E+07	8.9E+07	4.7E+07
	2	**7695.6**	7359.5	4.3E+07	1.8E+08	5.17E+09	1.7E+09	7.5E+09	4.1E+09
	3	**300.0**	**0.0**	1.3E+04	7356.5	3.30E+04	1.4E+04	5.6E+04	1.3E+04
Simple multimodal	4	**488.7**	34.0	1345.1	784.6	1244.4	191.0	1188.1	366.1
	5	**520.0**	0.00002	521.1	0.1	520.76	0.1	521.1	0.04
	6	**619.0**	4.2	633.2	7.1	646.1	3.7	630.2	4.0
	7	**700.010**	**0.0**	700.012	0.0	755.45	15.5	769.6	40.9
	8	994.3	43.5	**873.2**	16.6	933.03	17.8	995.5	37.4
	9	1095.9	42.9	**1080.4**	31.9	1241.2	36.5	1099.1	30.5
	10	5785.2	767.3	**1891.4**	271.4	4109.9	437.1	6422.0	747.6
	11	**6462.4**	916.3	1.3E+04	1885.9	9678	743.8	6815.0	1263.2
	12	**1200.1**	0.1	1202.6	0.5	1201.2	0.2	1202.4	1.5
	13	1300.6	0.1	**1300.57**	0.1	1300.8	0.1	1300.7	0.2
	14	1400.5	0.4	**1400.4**	0.1	1411.8	7.2	1416.1	12.3
	15	**1511.2**	2.4	1535.6	14.3	2844.6	2007.1	2715.7	1685.1
	16	1620.6	0.9	1621.8	0.6	1620.9	0.4	**1619.9**	1.0
Hybrid	17	**4.6E+04**	2.1E+04	4.8E+06	4.4E+06	1.21E+07	5.1E+06	4.3E+06	3.4E+06
	18	**3980.8**	1517.3	4.6E+04	1.2E+05	2.61E+06	2.1E+06	4.3E+07	8.6E+07
	19	**1920.6**	2.8	1956.5	29.2	1988.1	18.3	1986.8	26.7
	20	**2476.1**	101.4	1.0E+04	5735.7	3.17E+04	1.0E+04	1.7E+04	7988.7
	21	**6.1E+04**	3.7E+04	1.1E+06	1.7E+06	6.72E+06	3.7E+06	2.4E+06	2.3E+06
	22	**2920.9**	273.8	3502.9	410.6	3390	298.5	3000.2	280.5
Composition	23	**2645.0**	0.3	2663.9	6.2	2683.6	17.6	2741.9	53.9
	24	2678.7	3.4	2672.8	6.5	2716.7	7.2	**2600.0**	0.0005
	25	**2712.5**	3.1	2732.0	5.3	2739.2	7.4	2724.9	11.0
	26	2778.7	40.9	**2700.5**	0.1	2700.8	0.1	2780.6	39.8
	27	**3519.2**	96.8	3949.9	191.4	4212.6	100.1	3777.4	103.4
	28	5252.5	760.8	1.1E+04	1707.7	6238	757.6	**4939.2**	606.3
	29	2.9E+04	1.7E+04	**7369.4**	1.4E+04	1.51E+06	8.4E+05	5.4E+06	9.6E+06
	30	**3.9E+04**	7567.9	2.3E+05	2.0E+05	1.12E+05	4.5E+04	1.4E+05	1.1E+05

function), Fn12 (Shifted and Rotated Katsuura function), and Fn15 (Shifted and Rotated Expanded Griewank's plus Rosenbrock's function), where SAFIRO successfully acquired the value that is very near to the ideal fitness of 700, 1200, and 1500, respectively. Although SAFIRO ranked second behind PSO for Fn13 (Shifted and Rotated HappyCat function) and Fn14 (Shifted and Rotated HGBat function), SAFIRO managed to provide a very near-optimal solution. The same happens to Fn16 (Shifted and Rotated Expanded Scaffer's F6 function). Despite ranked third behind GWO and PSO, SAFIRO was able to obtain a very near-optimal solution for this function.

3) HYBRID FUNCTIONS (Fn17 TO Fn22)

In hybrid functions, the variables are divided into several subcomponents randomly, where different basic functions are used for different subcomponents [49]. The hybrid function is a combination of several multimodal functions (Fn19, Fn21, and Fn22), or it can be a combination of unimodal functions with simple multimodal functions (Fn17, Fn18, and Fn20). This makes the function more complicated to be solved. The readings in Table 6 show that SAFIRO has the capability to solve hybrid functions and obtain the best solution compared to other algorithms in all hybrid functions.

4) COMPOSITE FUNCTIONS (Fn23 TO Fn30)

SAFIRO is able to provide very competitive results by leading on half of the composite functions: Fn23, Fn25, Fn27, and Fn30. In composite functions, the capability of exploration and exploitation of an algorithm can be benchmarked concurrently due to many local optima contained the test functions [15]. Therefore, with these results, SAFIRO algorithm demonstrates a good balance between exploration and exploitation. For Fn24 and Fn26, even though SAFIRO did not rank first, but the solutions for both functions are close to the near-optimal solutions. The Fn24 is a combination of three simple multimodal functions which are Fn9 (Schwefel's function F10'), Fn10 (Rotated Rastrigin's function F9'), and Fn14 (Rotated HGBat function F14'). The Fn26 is a combination of one unimodal function which is Fn1 and four simple multimodal functions which are Fn6 (Rotated Weierstrass function F6'), Fn7 (Rotated Griewank's function F7'), Fn11 (Rotated Schwefel's function F11'), and Fn13 (Rotated HappyCat function F13').

Additionally, SAFIRO also shows a small standard deviation value in most of the functions. Its standard deviations values for Fn3 and Fn7 are equal to 0. These indicate that SAFIRO is capable to achieve the ideal fitness for 51 runs for both functions.

Next, the Friedman test is carried out as a statistical analysis tool to rank the performance of SAFIRO algorithm against the other three algorithms. The algorithm is ranked based on the mean fitness value over 51 runs for all 30 benchmark functions. The average rank is calculated for each algorithm where the lower rank value signifies a better algorithm performance.

According to Table 7, SAFIRO is ranked first, followed by PSO, GWO, and GA. The Friedman statistic is performed considering reduction performance distributed according to chi-square value of 31.05 with 3 degrees of freedom.

TABLE 7. Average ranking of the algorithms.

Algorithm	Ranking
SAFIRO	1.4833
PSO	2.3833
GWO	2.9333
GA	3.2

Based on the Friedman test performed, significant differences are detected between the algorithms. Hence, the null hypothesis is rejected and further analysis with a Post Hoc test [51] (using Holm's method) is done to determine whether the SAFIRO algorithm is better than the other algorithm or vice versa.

The result of the Holm Post Hoc test with significant level, $\alpha = 0.05$ is tabulated in Table 8.

In this case, Holm's procedure rejects those hypotheses that have p-value smaller than 0.025. According to Table VIII, the significant differences exist between the performances of SAFIRO, PSO, GWO, and GA. Thus, the null hypotheses are rejected. The proposed SAFIRO algorithm is performed significantly better than the other three algorithms with an unadjusted p-value ≤ 0.025.

TABLE 8. Holm post HOC result for $\alpha = 0.05$

Algorithms	z	p	Holm
SAFIRO vs PSO	2.7	0.006934	0.0125
SAFIRO vs GWO	4.35	0.000014	0.01
SAFIRO vs GA	5.15	0	0.008333

C. CONVERGENCE BEHAVIOR ANALYSIS

The graph of convergence curve is generated to observe the ability of SAFIRO and other algorithms to reach to an optimal or a near-optimal solution throughout the optimization process. In this case, each convergence curve shows the mean fitness of the best solution against 10,000 iterations over 51 runs.

Statistically, SAFIRO demonstrates a very good result by leading in 20 out of 30 functions in solving the CEC 2014 Benchmark Test Suite. Among the functions that SAFIRO ranked first, Fn3, Fn7, Fn19, and Fn23 functions were selected to visualize the convergence curve of unimodal, simple multimodal, hybrid and composition functions, respectively.

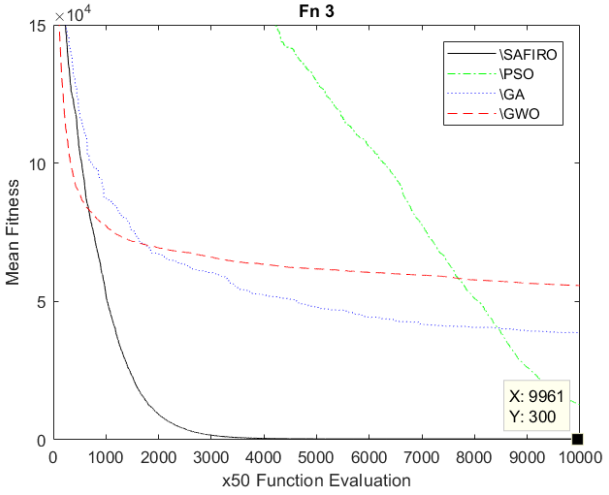


FIGURE 8. Convergence curves comparison for unimodal function (Fn3).

1) CONVERGENCE CURVE

Based on the graphs pattern in Fig. 8 to Fig. 11, it can be shown that there are drastic changes at the beginning of the search where the graphs decreased gradually over the course of the iterations. These conditions reflect the exploration and exploitation phases that occurred throughout the optimization process, as stated by Berg in [52]. Then, the graphs plateau after certain iterations until the end. These indicate

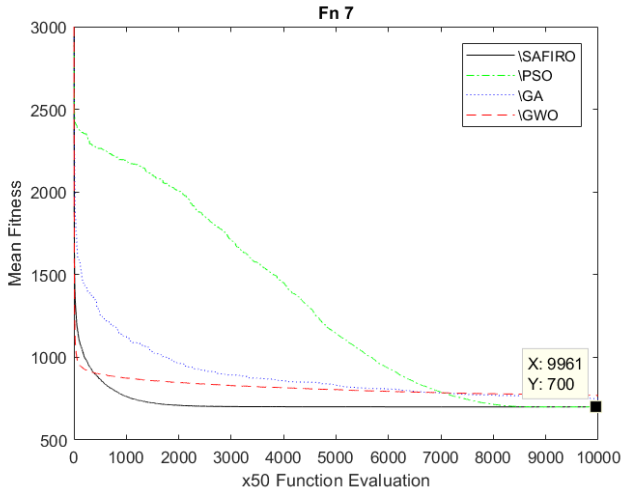


FIGURE 9. Convergence curves comparison for simple multimodal function (Fn7).

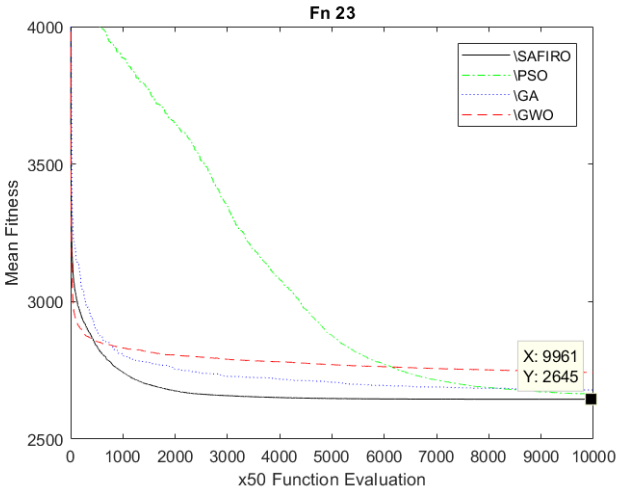


FIGURE 11. Convergence curves comparison for composition function (Fn23).

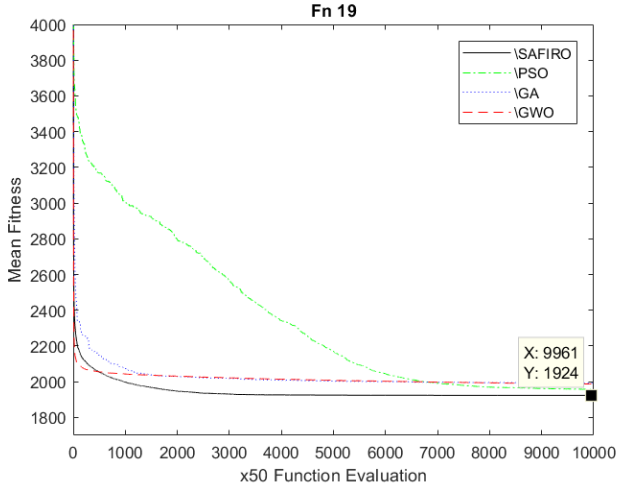


FIGURE 10. Convergence curves comparison for hybrid function (Fn19).

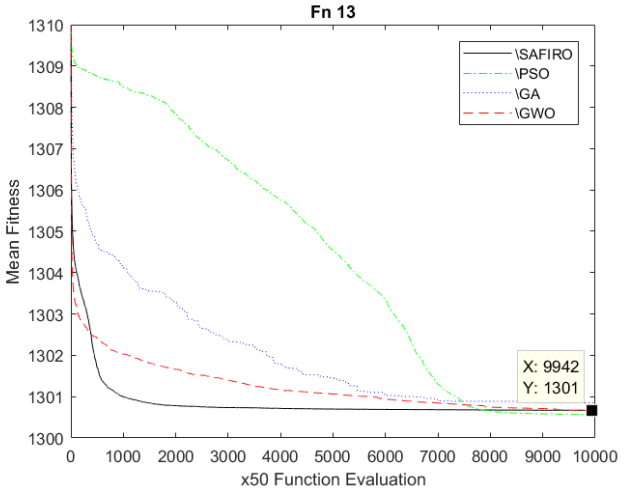


FIGURE 12. Convergence curves comparison for simple multimodal function (Fn13).

that the search agent moves from the exploration process to the exploitation process before stopping the process to find a better solution when it reaches the maximum iteration.

The mean fitness of SAFIRO (towards the end of iterations) for Fn3, Fn7, Fn19, and Fn23 are about 300, 700, 1924 and 2645, respectively. These values are almost the same as the mean fitness values tabulated in Table 6. As can be seen in Fig. 8 to Fig.11, SAFIRO is able to find good solution within lesser number of fitness evaluation.

In Fig. 9, although both SAFIRO and PSO are able to reach the ideal fitness value (optimal solution) for Fn7, SAFIRO needs fewer number of fitness evaluation compared to PSO.

The mutation and shrinking local search neighbourhood method allow the search agent to converge, exploit and look for a good solution within a subarea of the search space. However, the fast convergence of SAFIRO is also the cause for its second and third rank performance in the 10 other test functions. This can be observed in Fig. 12 and Fig. 13, which are the convergence curves for Fn13 and Fn29. The fast convergence caused SAFIRO's agent to be trapped in local optima which prevents it to escape and look for a better solution. On the other hand, PSO which shows gradual convergence is able to find a better solution due to its ability to explore more search area.

To specifically observe the behaviour of SAFIRO's agent over the course of iterations, the trajectory, search history, and fitness trend of the search agent are presented. For this aim, SAFIRO algorithm is evaluated on the two-dimensional version of CEC 2014 Benchmark Test Suite, with 100 iterations only. All hybrid functions and Fn29 and Fn30 of composition functions are excluded in this experiment due to these functions are not specified in a two-dimensional domain. The selected functions: Fn3, Fn7, Fn23 are captured graphically which represent unimodal, simple multimodal and composition function, respectively.

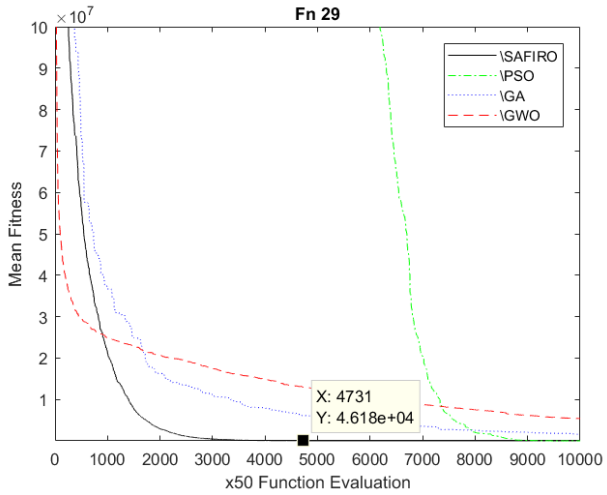


FIGURE 13. Convergence curves comparison for composition function (Fn29).

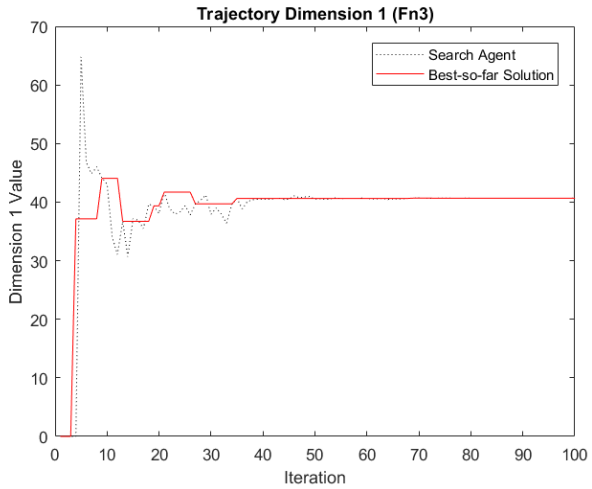


FIGURE 14. The trajectory of SAFIRO's agent for one-dimensional of unimodal function (Fn3) during the optimization process.

2) TRAJECTORY OF SAFIRO AGENT

The trajectory of the search agent for one and two-dimensional solution are plotted to see the movement of SAFIRO's agent during the iterations. Two graphs are produced for each dimension to observe the solution of search agent against the best-so-far solution (*X_best_so_far*) throughout the process. It can be observed in Fig. 14 to Fig. 19 that SAFIRO confirms the exploration process during the first half of the optimization process with obvious changes of the agent's movement, whereas the slow changes during the following iterations ensure the exploitation process. The plateau graph towards the end of the iteration shows that SAFIRO's agent managed to find the near-optimal solution before reaching the maximum iteration.

3) SEARCH HISTORY OF SAFIRO AGENT

Next, the search history of the agent is also traced to observe the capability of SAFIRO's agent in exploring and exploiting the search space, in finding the best solution. Mobilities of SAFIRO's agent are marked and plotted on the contour map for two-dimensional functions of the selected

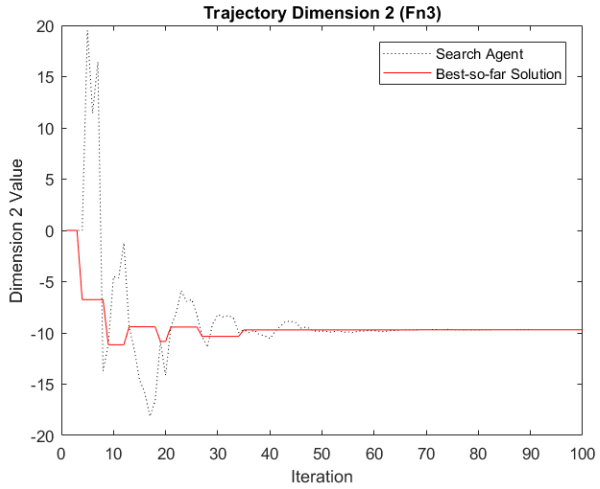


FIGURE 15. The trajectory of SAFIRO's agent for two-dimensional of unimodal function (Fn3) during the optimization process.

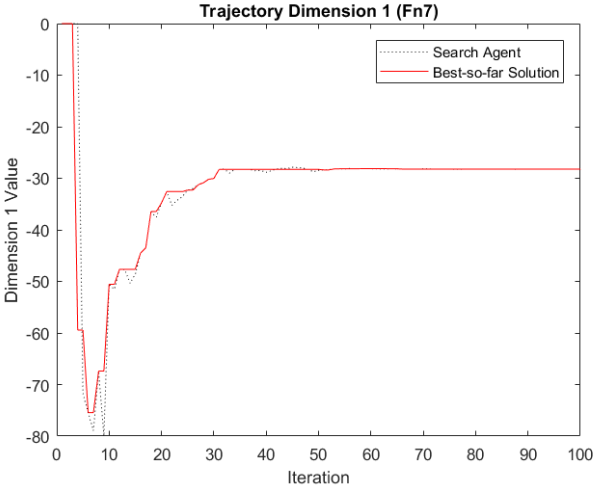


FIGURE 16. The trajectory of SAFIRO's agent for one-dimensional of multimodal function (Fn7) during the optimization process.

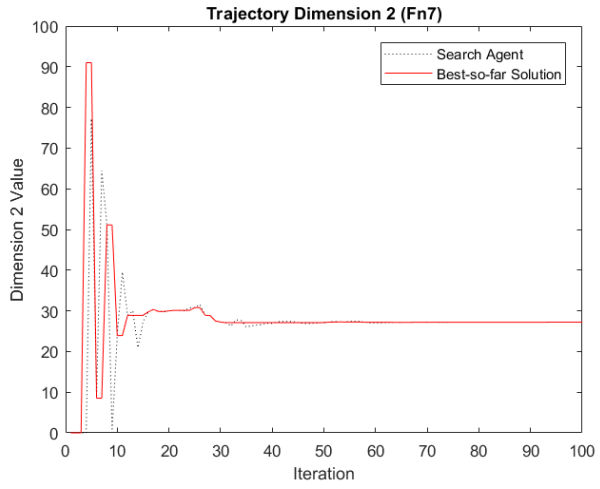


FIGURE 17. The trajectory of SAFIRO's agent for two-dimensional of multimodal function (Fn7) during the optimization process.

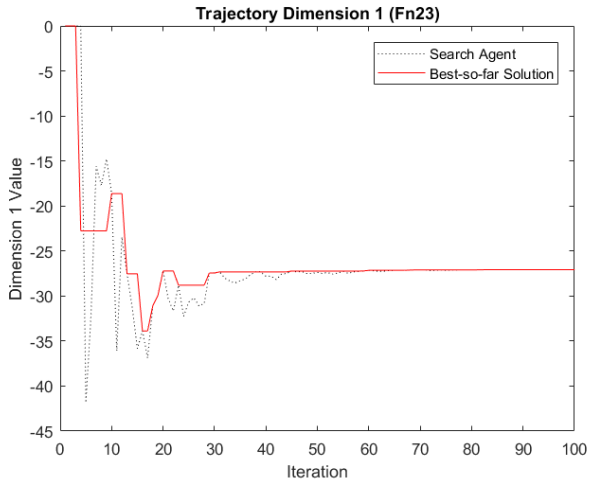


FIGURE 18. The trajectory of SAFIRO's agent for one-dimensional of composition function (Fn23) during the optimization process.

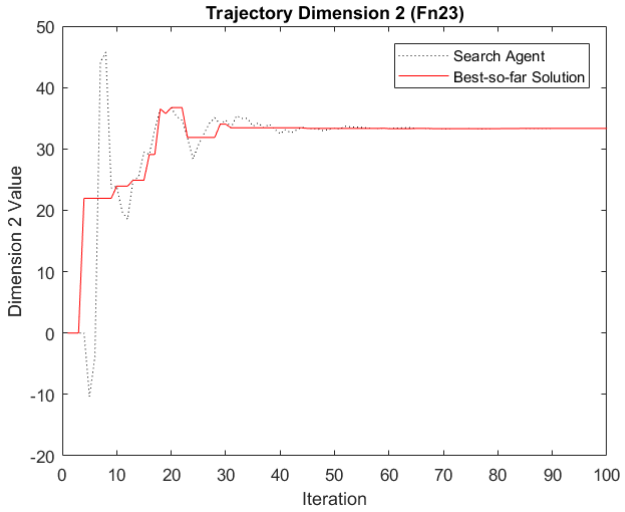


FIGURE 19. The trajectory of SAFIRO's agent for two-dimensional of composition function (Fn23) during the optimization process.

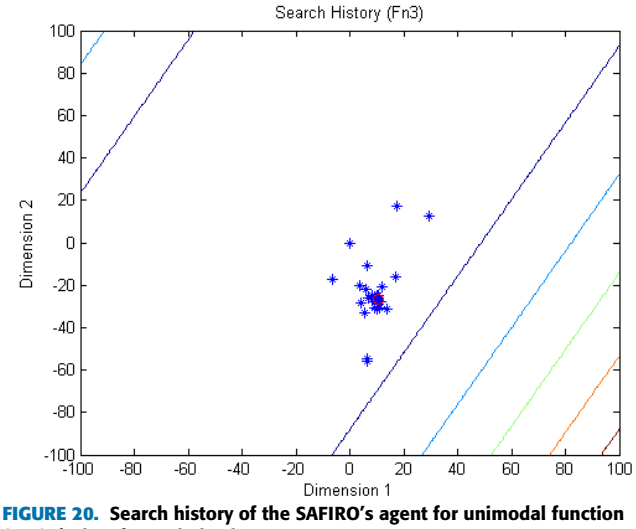


FIGURE 20. Search history of the SAFIRO's agent for unimodal function (Fn3) during the optimization process.

unimodal, multimodal and composition problems as illustrated in Fig. 20 to Fig. 22. The star symbol indicates the locations visited by the agent which represent the solution for

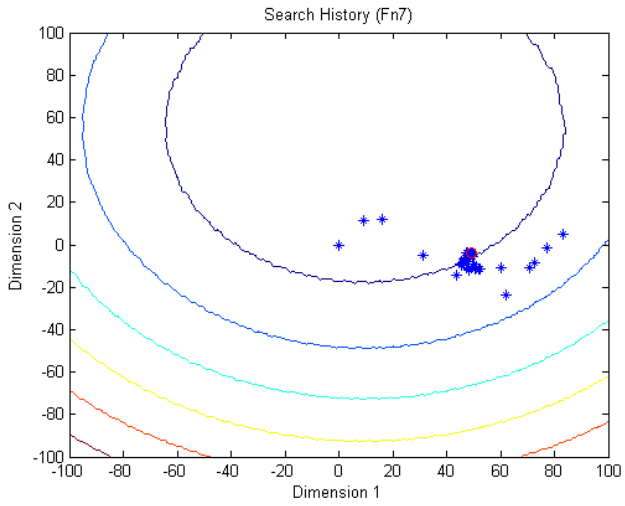


FIGURE 21. Search history of the SAFIRO's agent for multimodal function (Fn7) during the optimization process.

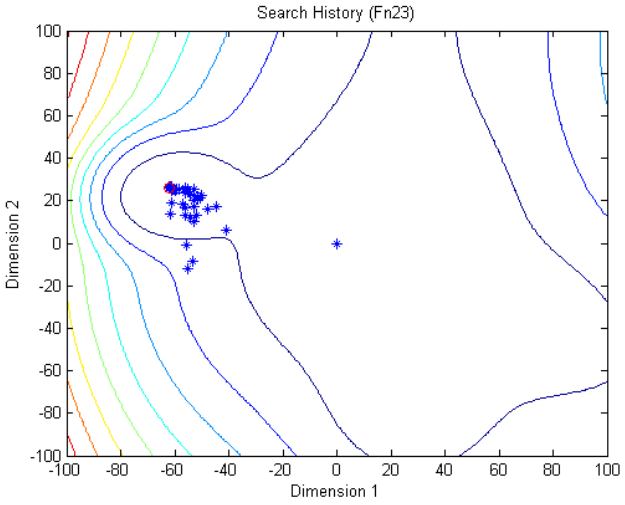


FIGURE 22. Search history of the SAFIRO's agent for composition function (Fn23) during the optimization process.

each iteration. The circle in each figure represents the final best solution (*X_best_so_far*). It is shown that SAFIRO's agent is able to adequately explore promising areas of the search space and then exploit around the best solution. As the iteration increases, the agent moves towards the best-so-far solution. This is due to the decrease in the radius of the local neighbourhood, δ as stated in (9).

4) FITNESS TREND OF SAFIRO AGENT

Lastly, the fitness trend is generated to observe the pattern of the agent's solution against the best solution-so-far (*X_best_so_far*) over 1 run for 100 iterations. Based on Fig. 23 to Fig. 25, it can be concluded that SAFIRO's agent went through sufficient exploration at the beginning of the search and decline slowly to have a better exploitation around the improved best-so-far solution for unimodal, simple multimodal and composition functions. This pattern ensures that

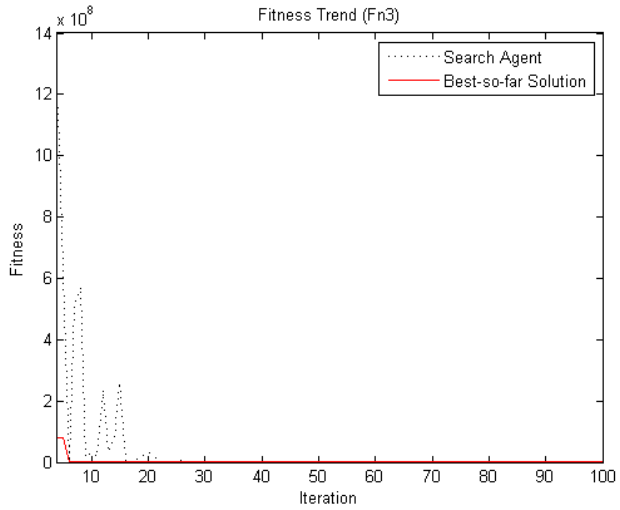


FIGURE 23. Fitness trend of the SAFIRO's agent for unimodal function (Fn3) during the optimization process.

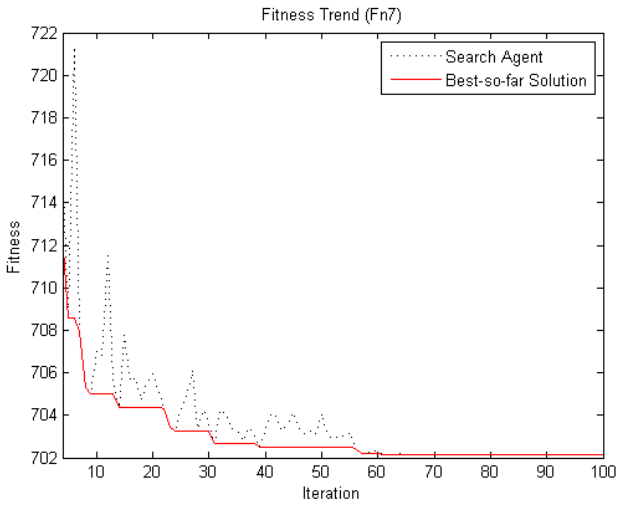


FIGURE 24. Fitness trend of the SAFIRO's agent for multimodal function (Fn7) during the optimization process.

SAFIRO algorithm eventually converges to the near-optimal solution in the search area.

Overall, SAFIRO performs very well in solving unimodal and hybrid functions and demonstrates a highly competitive yet promising performance in solving simple multimodal and composition functions. SAFIRO shows a good ability to reach the optimal and near-optimal solution with a better number of best mean fitness value compared to other algorithms. These results clearly proved that the SAFIRO algorithm managed to iteratively estimate the optimal and near-optimal solution for varieties of optimization problems (unimodal, multimodal, hybrid and composition functions) with different complexities and able to provide great solutions by leading 20 out of 30 functions in solving the CEC 2014 Benchmark Test Suite. Despite working with a single-agent, SAFIRO is capable to outperform

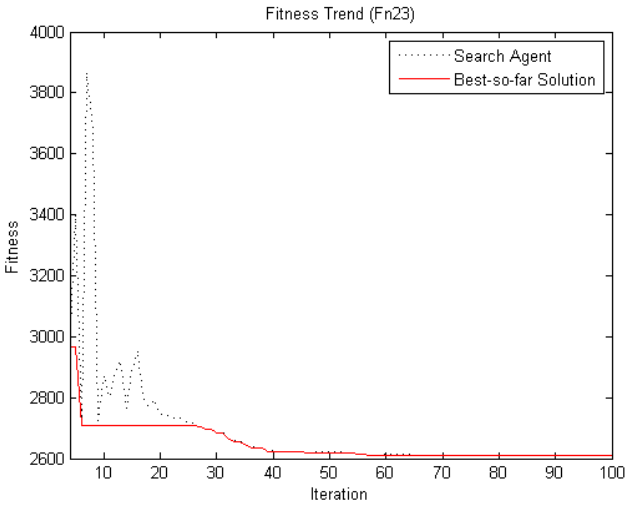


FIGURE 25. Fitness trend of the SAFIRO's agent for composition function (Fn23) during the optimization process.

population-based algorithms under the same number of function evaluations.

VI. CONCLUSION

This paper introduces a new single-agent metaheuristic optimization algorithm inspired by the estimation competency of FIR filter, named Single-agent FIR Optimizer (SAFIRO) algorithm. SAFIRO algorithm aims to find an estimate of an optimal or a near-optimal solution for an optimization problem. The proposed algorithm employed the two steps in UFIR filter procedure: measurement and estimation. Besides UFIR framework, SAFIRO uses a random mutation of $X\_best\_so\_far$ with local neighbourhood method to improve its estimation of an optimal or a near-optimal solution. In order to evaluate the performance of SAFIRO algorithm, the CEC 2014 Benchmark Test Suite has been applied. Experimental results indicate that SAFIRO is capable to converge to an optimal and a near-optimal solution and significantly outperforms well-known algorithms such as PSO, GA, and GWO in solving the CEC 2014 benchmark problems.

It is important to notice, the proposed SAFIRO algorithm contributes to new knowledge and provides a platform for other researchers to explore, modify or hybrid this algorithm with other algorithms to produce a better metaheuristic algorithm in solving optimization problems. The SAFIRO algorithm can also be used by other researchers to solve optimization problems in various fields.

VII. FUTURE WORK

For future work, the strategy to escape from local optima will be investigated and incorporated into SAFIRO. Other than that, SAFIRO's algorithm applicability will be tested to solve real engineering optimization problems such as tuning parameters in proportional-integral-derivative (PID) controllers and solving printed circuit board (PCB) routing problem.

ACKNOWLEDGEMENT

This research was financially supported by Fundamental Research Grant Scheme (FRGS) awarded by the Ministry of Education (MOE) to Multimedia University (FRGS/1/2015/TK04/MMU/03/02). The authors would like to thank anonymous reviewers for their constructive comments, and Universiti Tun Hussein Onn Malaysia, Ministry of Higher Education of Malaysia (MOHE), Multimedia University, and Universiti Malaysia Pahang for their logistics support.

REFERENCES

[1] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: Wiley, 2009.

[2] X.-S. Yang, Ed., *Recent Advances in Swarm Intelligence and Evolutionary Computation*. Cham, Switzerland: Springer, 2015.

[3] H. Yi, Q. Duan, and T. W. Liao, "Three improved hybrid metaheuristic algorithms for engineering design optimization," *Appl. Soft Comput. J.*, vol. 13, no. 5, pp. 2433–2444, 2013.

[4] Z. Beheshti and S. M. Shamsuddin, "A review of population-based meta-heuristic algorithm," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.

[5] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.

[6] M. H. Salmani and K. Eshghi, "A metaheuristic algorithm based on chemotherapy science: CSA," *J. Optim.*, vol. 2017, Feb. 2017, Art. no. 3082024.

[7] F. Glover and M. Laguna, *Tabu Search-Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves, Ed. Oxford, U.K.: Blackwell Scientific Publications, 1993, pp. 70–150.

[8] A. Acan and A. Ünveren, "A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem," *Appl. Soft Comput. J.*, vol. 36, pp. 185–203, Nov. 2015.

[9] B. Dogan and T. Olmez, "A new metaheuristic for numerical function optimization: Vortex search algorithm," *Inf. Sci.*, vol. 293, pp. 125–145, Feb. 2015.

[10] J. L. Rueda and I. Erlich, "MVMO for bound constrained single-objective computationally expensive numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, May 2015, pp. 1011–1017.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. 1995, pp. 1942–1948.

[12] F. Javidrad and M. Nazari, "A new hybrid particle swarm and simulated annealing stochastic optimization method," *Appl. Soft Comput.*, vol. 60, pp. 634–654, Nov. 2017.

[13] D. Cheong, Y. M. Kim, H. W. Byun, K. J. Oh, and T. Y. Kim, "Using genetic algorithm to support clustering-based portfolio optimization by investor information," *Appl. Soft Comput.*, vol. 61, pp. 593–602, Dec. 2017.

[14] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[16] I. Fister, Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *Electrotech. Rev.*, vol. 80, no. 3, pp. 1–7, 2013.

[17] M. Bakhshipour, M. J. Ghadi, and F. Namdari, "Swarm robotics search & rescue: A novel artificial intelligence-inspired optimization approach," *Appl. Soft Comput. J.*, vol. 57, pp. 708–726, Aug. 2017.

[18] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[19] N. S. Jaddi, J. Alvankarian, and S. Abdullah, "Kidney-inspired algorithm for optimization problems," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 42, pp. 358–369, Jan. 2017.

[20] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti, and D. N. A. Jawawi, "Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm," *Swarm Evol. Comput.*, vol. 26, pp. 8–22, Feb. 2016.

[21] H. Varaee and M. R. Ghasemi, "Engineering optimization based on ideal gas molecular movement algorithm," *Eng. Comput.*, vol. 33, no. 1, pp. 71–93, 2016.

[22] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[23] W. H. Kwon, P. S. Kim, and S. H. Han, "A receding horizon unbiased FIR filter for discrete-time state space models," *Automatica*, vol. 38, no. 3, pp. 545–551, Mar. 2002.

[24] V. F. Yu, A. A. N. P. Redi, Y. A. Hidayat, and O. J. Wibowo, "A simulated annealing heuristic for the hybrid vehicle routing problem," *Appl. Soft Comput.*, vol. 53, pp. 119–132, Apr. 2017.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[26] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.

[27] J. Sánchez-Oro, M. Sevaux, A. Rossi, R. Martí, and A. Duarte, "Improving the performance of embedded systems with variable neighborhood search," *Appl. Soft Comput.*, vol. 53, pp. 217–226, Apr. 2017.

[28] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—A comparative case study," in *Parallel Problem Solving From Nature* (Lecture Notes in Computer Science), vol. 1498. Berlin, Geramny: Springer, 1998, pp. 292–301.

[29] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, nos. 2–3, pp. 243–278, 2005.

[30] R.-H. Huang and T.-H. Yu, "An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting," *Appl. Soft Comput.*, vol. 57, pp. 642–656, Aug. 2017.

[31] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing*. Berlin, Geramny: Springer, 2007, pp. 789–798.

[32] X. Li and G. Yang, "Artificial bee colony algorithm with memory," *Appl. Soft Comput.*, vol. 41, pp. 362–372, Apr. 2016.

[33] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*. Berlin, Geramny: Springer, 2009, pp. 169–178.

[34] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.

[35] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2012.

[36] C. K. Ahn, S. Zhao, Y. S. Shmaliy, and R. Sakthivel, "A revisit to strictly passive FIR filtering," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, to be published.

[37] Y. S. Shmaliy and D. Simon, "Iterative unbiased FIR state estimation: A review of algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2013, no. 1, p. 113, 2013.

[38] Y. S. Shmaliy, S. Khan, and S. Zhao, "Ultimate iterative UFIR filtering algorithm," *Measurement*, vol. 92, pp. 236–242, Oct. 2016.

[39] Y. S. Shmaliy, S. H. Khan, S. Zhao, and O. I. Manzano, "General unbiased FIR filter with applications to GPS-based steering of oscillator frequency," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 1141–1148, May 2017.

[40] S. Zhao, Y. S. Shmaliy, F. Liu, and S. H. Khan, "Unbiased, optimal, and in-betweens: The trade-off in discrete finite impulse response filtering," *IET Signal Process.*, vol. 10, no. 4, pp. 325–334, 2016.

[41] K. V. Ling and K. W. Lim, "Receding horizon recursive state estimation," *IEEE Trans. Autom. Control*, vol. 44, no. 9, pp. 1750–1753, Sep. 1999.

[42] S. Zhao, C. K. Ahn, Y. S. Shmaily, P. Shi, and R. K. Agarwal, "Iterative filter with finite measurements for suddenly maneuvering targets," *J. Guid., Control, Dyn.*, vol. 40, no. 9, pp. 2316–2322, 2017.

[43] C. K. Ahn and Y. S. Shmaliy, "New receding horizon FIR estimator for blind smart sensing of velocity via position measurements," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 1, pp. 135–139, Jan. 2018.

[44] C. K. Ahn, "A new solution to the induced ℓ_∞ finite impulse response filtering problem based on two matrix inequalities," *Int. J. Control*, vol. 87, no. 2, pp. 404–409, 2014.

[45] S. Zhao, Y. S. Shmaliy, and F. Liu, "Fast Kalman-like optimal unbiased FIR filtering with applications," *IEEE Trans. Signal Process.*, vol. 64, no. 9, pp. 2284–2297, May 2016.

[46] S. Zhao, Y. S. Shmaliy, and F. Liu, "Fast computation of discrete optimal FIR estimates in white Gaussian noise," *IEEE Trans. Signal Process. Lett.*, vol. 22, no. 6, pp. 718–722, Jun. 2015.

[47] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural exponential inertia weight strategy in particle swarm optimization," in *Proc. 6th World Congr. Intell. Control Autom.*, vol. 1. 2002, pp. 3672–3675.
[48] L. J. Morales-Mendoza, R. F. Vázquez-Bautista, E. Morales-Mendoza, Y. Shmaliy, and H. Gamboa-Rosales, "A new recursive scheme of the unbiased FIR filter to image processing," *Procedia Eng.*, vol. 35, pp. 202–209, 2012.
[49] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou China, Tech. Rep. 201311, 2013.
[50] N. Azlina, A. Aziz, M. Mubin, Z. Ibrahim, and S. W. Nawawi, "Statistical analysis for swarm intelligence—Simplified," *Int. J. Future Comput. Commun.*, vol. 4, no. 3, pp. 193–197, 2015.
[51] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[52] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, 2006.

TASIRANSURINI AB RAHMAN received the B.Eng. degree (Hons) in electrical engineering from Kolej Universiti Tun Hussein Onn, Johor, Malaysia, the M.Eng. degree in electrical engineering (communication and computer network) from Universiti Kebangsaan Malaysia, Bangi, Malaysia, and the certificate and Diploma degrees in electronic engineering from the Politeknik Sultan Haji Ahmad Shah, Pahang, Malaysia.

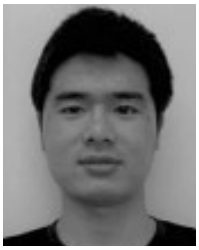
She is currently pursuing the Ph.D. degree with the Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, Pekan, Pahang, Malaysia. Her current research interests include computational intelligence and its application in engineering.

ZUWAIRIE IBRAHIM received the B.Eng. degree in electrical engineering and the M.Eng. degree in image processing from Universiti Teknologi Malaysia, Johor, Malaysia, in 2000 and 2003, respectively, and the Ph.D. degree in DNA computing from Meiji University, Tokyo, Japan, in 2006.

From 2002 to 2008, he was a Lecturer with Universiti Teknologi Malaysia, Johor, Malaysia. He was then promoted to Senior Lecturer in 2008 and was with Universiti Teknologi Malaysia until 2012. In 2012, he moved to Universiti Malaysia Pahang, Pahang, Malaysia, to serve as an Associate Professor. He is one of the inventors of the simulated Kalman filter, which is an optimization algorithm based on the Kalman filter framework. His research interests include the fundamentals and applications of computational intelligence, specifically, particle swarm optimization, ant colony optimization, gravitational search algorithms, and black hole algorithms.

NOR AZLINA AB. AZIZ received the B.Eng. degree (Hons.) in electronics majoring in computer and the M.Eng.Sc degree from Multimedia University, Malaysia, and the Ph.D. degree from the University of Malaya, Malaysia. Her Ph.D. and master's research works were in the field of computational intelligence.

She is currently a Lecturer with the Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia. She is also the Chair of the Engineering Computational Intelligence Special Interest Group, Multimedia University. Her research interests include the fundamental aspects and applications of computational intelligence in engineering.



SHUNYI ZHAO (M'14) was born in Jinhua, China, in 1987. He received the Ph.D. degree in control theory and application from the Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Institute of Automation, Jiangnan University, Wuxi, China, in 2015. From 2013 to 2014, he was a Visiting Student with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada.

He joined Jiangnan University in 2015 as an Associate Professor. He is currently a Post-Doctoral fellow with the University of Alberta. He is one of the inventors of ultimate unbiased finite impulse response filters. His research interests include statistical signal processing, estimation theory, and fault detection and diagnosis.

NOR HIDAYATI ABDUL AZIZ (M'06–SM'10) received the B.Eng. degree (Hons) in electronics engineering majoring in computer from Multimedia University in 2002 and the M.Eng. degree in electrical (electronics and telecommunications) from Universiti Teknologi Malaysia in 2005.

She is currently pursuing the Ph.D. degree in computational intelligence with Universiti Malaysia Pahang. From 2002 to 2006, she was an Engineer with Telekom Malaysia Berhad, Malaysia. She then joined Multimedia University as a Lecturer in 2006. She is one of the inventors of the simulated Kalman filter, which is an optimization algorithm based on Kalman filter framework. Her research interests include fundamental and applications of computational intelligence.