Sādhanā Vol. 123, No.4, January 2016, pp.2333–2335
DOI 12.3456/s78910-011-012-3

Single-solution simulated Kalman filter algorithm for global optimisation problems

NOR HIDAYATI ABDUL AZIZ[1,2,*], ZUWAIRIE IBRAHIM[2], NOR AZLINA AB AZIZ[1], MOHD SABERI MOHAMAD[3] and JUNZO WATADA[4]

[1] Faculty of Engineering and Technology, Multimedia University, 75450 Bukit Beruang, Melaka, Malaysia.
[2] Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia.
[3] Faculty of Creative Technology and Heritage, Universiti Malaysia Kelantan, Karung Berkunci 01, 16300 Bachok, Kelantan, Malaysia
[4] Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar, 32610 Teronoh, Perak, Malaysia.

Abstract. This paper introduces single-solution Simulated Kalman Filter (ssSKF), a new single-agent optimisation algorithm inspired by Kalman Filter, for solving real-valued numerical optimisation problems. In comparison, the proposed ssSKF algorithm supersedes the original population-based Simulated Kalman Filter (SKF) algorithm by operating with only a single agent, and having less parameters to be tuned. In the proposed ssSKF algorithm, the initialisation parameters are not constants, but, are produced by random numbers taken from a normal distribution in the range of $[0, 1]$, thus excluding them from tuning requirement. In order to balance between the exploration and exploitation in ssSKF, the proposed algorithm uses an adaptive neighbourhood mechanism during its prediction step. The proposed ssSKF algorithm is tested using the 30 benchmark functions of CEC 2014, and its performance is compared to the original SKF algorithm, Black Hole (BH) algorithm, Particle Swarm Optimisation (PSO) algorithm, Grey Wolf Optimiser (GWO) algorithm, and Genetic Algorithm (GA). The results show that the proposed ssSKF algorithm is a promising approach and able to outperform GWO and GA algorithms, significantly.

MS received 1 January 2016; revised 1 January 2016; accepted 1 January 2016

Keywords. single-solution, adaptive neighbourhood, SKF, Kalman, optimisation, metaheuristics.

1 Introduction

Heuristic optimisation method is becoming more relevant in today's world. The complexity of many real-world optimisation problems have turn scientists and engineers to heuristic methods to solve their problems, where optimality is being traded-off with near optimal solutions that can be achieved within reasonable computational time.

Metaheuristic optimisation algorithms characterise a group of general-purpose heuristic optimisation methods that is governed by a higher-level strategy that leads the search [1]. Derivation of mathematical models of the optimisation problems is not required when using metaheuristics methods as the problems are treated like black boxes [2]. Genetic Algorithm (GA) [3] and Particle Swarm Optimisation (PSO) [4] are some well-known examples of metaheuristics algorithms. The search for the best global optimisation algorithm still continues despite the introduction of "No Free Lunch (NFL) Theorems for Optimisation" by Wolpert and Macready in 1997 [5]. The NFL theorems suggest that in average, all optimisation algorithms perform equally when all types of optimisation problems are taken into consideration. However, instead of hampering the field, these NFL theorems have inspired researchers to keep on improving and proposing new optimisation algorithms in search for the best optimisation algorithm that works for most problems, even if not for all. Thus, several new global optimisation algorithms, mostly nature-inspired, have been developed over the last few decades [6–8]. The Black Hole (BH) [9], Grey Wolf Optimiser (GWO) [10] and Simulated Kalman Filter (SKF) [11] are few examples of recently proposed metaheuristics algorithms.

Based on the number of agents used, a metaheuristic algorithm can be classified into either single solution-based metaheuristics or population-based metaheuristics. Single solution-based metaheuristics make use of only a single agent, improved from one iteration to another. Simulated Annealing (SA) [12], Tabu Search (TS) [13], and Variable Neighbourhood Search (VNS) [14] are examples of algorithms in this category. Population-based metaheuristics adopt a number of agents to explore the search space in order to solve an optimisation problem. Besides GA, PSO, BH, GWO and SKF, examples of population-based metaheuristics include Ant Colony Optimisation (ACO) [15], Firefly (FA) [16] and Cuckoo Search (CS) [17]. Population-based algorithms are said to perform better because they employ a number of agents (normally many) that shares information about the search space to avoid local optima stagnation [18]. Due to this

*For correspondence

strong point, many population-based algorithms are employed to solve challenging optimisation problems.

Previously, the original SKF algorithm was introduced as a population-based metaheuristic algorithm. It has been subjected to solve various types of benchmark optimisation problems [19]. SKF makes use of a population of agents that operates using a standard Kalman Filter framework to solve optimisation problems. Each agent in SKF makes estimation of the optimum based on a simulated measurement process that is guided by a best-so-far solution. Kalman Filter, named after its founder, is a renowned state estimation algorithm based on minimum mean square error method [20]. Kalman Filter is considered as an optimal estimator for a linear system, especially when the noises are Gaussian in nature. While multiple sequential measurements are normally required to come out with a good estimation of a system's state, Kalman Filter requires only the last estimated state and a new measurement to come out with a better estimation. The capability of the Kalman Filter to make a good estimation, supported by the information sharing between agents, make SKF a good global optimiser and a competitive algorithm compared to existing metaheuristic algorithms. Ever since its introduction, SKF has undergoes various adaptations and applied in many applications. These include extensions of the SKF algorithm to deal with combinatorial optimisation problems [21–23], and hybridisation of the SKF algorithm with PSO algorithm [24] and GSA algorithm [25]. The population-based SKF algorithm has been applied to find the optimal path of a 14-hole drill path optimisation problem [26] and Airport Gate Allocation Problems (AGAP) [27]. The discrete type of the SKF algorithm has been subjected to resolving the AGAP [28] and to solve feature selection problem for EEG peak detection [29].

Despite its good performance, SKF is not a parameterless algorithm. In the SKF algorithm, three parameter values are assigned during initialisation, the initial error covariance, $P(0)$, the process noise, Q, and the measurement noise, R. Parameter tuning is a tedious task, and the process itself can be considered as an optimisation problem. For example, some Evolutionary Algorithms (EA) have many parameters that are hard to tune. The challenges in EA are not only the requirement of good initial parameter values, but also the excessive sensitivities of some of the EA's parameters towards the overall performance. Genetic Algorithm (GA) is another example of algorithms that has many setting parameters. Parameters in GA include the probability of mutation, probability of crossover, and the selection procedure. Particle Swarm Optimisation (PSO) on the other hand, despite being easy to be understood, has 3 parameters to be tuned. Some classical algorithms, such as Tabu Search (TS) and Simulated Annealing (SA), has at least 1 or 2 parameters that require tuning. Usage of such algorithms requires some preliminary tuning computation of the parameters before it can be applied to solve an optimisation problem. One alternative is to offer some default values for the parameters. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [30] is an example of algorithms that offers some default parameter values to the users. These values are claimed to be applicable to any optimisation problems in hand. Self-tuning parameters, like what has been introduced in Differential Evolution (DE) [31], is another alternative solution. Ultimately, parameter-free algorithms such as Black Hole and Symbiotic Organisms Search (SOS) [32] are desirable.

Further investigations on the effectiveness of Kalman Filter estimation capability as a source of inspiration for metaheuristics optimisation algorithms suggests that it can be realised by using only a single Kalman Filter estimator. Thus, in this paper, a single agent version of the SKF algorithm is proposed to solve single objective, real parameter optimisation problems. The proposed algorithm, named as single-solution Simulated Kalman Filter (ssSKF), requires only one single agent. This agent iteratively improves its estimation according to the standard Kalman Filter framework with the help of adaptive neighbourhood method during its prediction step. The problem of parameter tuning is reduced by adopting normally distributed random numbers for all three parameters, $P(0)$, Q, and R whenever they are needed [33, 34]. The normally distributed random numbers are scaled and shifted so that they lie in the range of 0 to 1, defined by $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0.5, 0.1)$. However, a new parameter α is introduced, but, a fixed value is suggested. In this study, the ssSKF algorithm is tested using all 30 benchmark functions of CEC 2014 benchmark suite [35], and is compared against some existing metaheuristic optimisation algorithms including the state-of-art PSO and GA.

The remaining part of the paper is organised as follows. Section 2 gives a brief description of the Kalman Filter framework. In Section 3, a detailed description of the proposed single solution-based SKF algorithm is explained. Section 4 describes the experimental setup. Next, the experimental results and discussion are presented in Section 5. And finally, Section 6 concludes the paper.

2 Kalman Filter Framework

Kalman Filter is a well-known state estimation method of a dynamic system that is excited by a stochastic process and a measurement noise. Kalman Filter is used to estimate the state variable, $\mathbf{X}$, of a discrete time controlled process that is governed by a linear stochastic difference equation by using a measurement, $\mathbf{Z}$.

Kalman Filter is based on two sets of recursive mathematical equations that efficiently estimate the state of a process by obtaining feedback in the form of noisy measurements. The first set of equations is called the *predict* equations, whereas the second set of equations is called the *estimate* equations.

Given the current state and error covariance estimates ($\mathbf{X}(t)$ and $P(t)$), predict equations are used to make prediction of the next estimation. Equation (1) and (2) show the Kalman Filter predict equations with the superscript (T) represents a transpose operation.

$$\mathbf{X}(t|t+1) = A(t) \times \mathbf{X}(t) \times A^T(t) + B(t) \times \mathbf{u}(t) \qquad (1)$$

$$P(t|t+1) = P(t) + Q(t) \quad (2)$$

where $\mathbf{X}(t|t+1)$ is the predicted state vector, $\mathbf{X}(t)$ is the estimated state vector at time t, $P(t|t+1)$ is the predicted error covariance matrix reflecting the error of estimation, $P(t)$ is the estimated error covariance matrix at time t, $Q(t)$ is the process covariance matrix reflecting the error due to process. $A(t)$ is the state transition matrix that determines the transition between states for each system state variable, $B(t)$ is the control input matrix that maps the control input vector, $\mathbf{u}(t)$, to state vector, $\mathbf{X}(t)$, and $\mathbf{u}(t)$ is the control input vector containing any control input variables.

Once a measurement is received, estimate equations are used to obtain a better estimation by incorporating the new measurement into the predicted values. Equation (3) and (4) show the Kalman Filter estimate equations.

$$\mathbf{X}(t+1) = \mathbf{X}(t|t+1) + K(t) \times (\mathbf{Z}(t) - H(t) \times \mathbf{X}(t|t+1)) \quad (3)$$

$$P(t+1) = P(t|t+1) - K(t) \times H(t) \times P(t|t+1) \quad (4)$$

where $\mathbf{X}(t+1)$ is the estimated state vector for the next time step, $P(t+1)$ is the estimated error covariance matrix for the next time step, $\mathbf{Z}(t)$ is the measurement at time t, $H(t)$ is the measurement matrix that maps the measurements onto the state vector variables, and $K(t)$ is the Kalman gain.

The Kalman gain improves state estimate by minimising the error covariance during each iteration, and can be calculated as (5).

$$K(t) = (P(t|t+1) \times H^T(t)) \times (H(t) \times P(t|t+1) \times H^T(t) + R(t))^{-1} \quad (5)$$

where $R(t)$ is the measurement covariance matrix reflecting the error from measurements.

In Kalman Filter framework, the process and the measurement noise are assumed to be independent of each other and normally distributed. If the assumption is true, Kalman Filter effectively minimises the mean squared error (MSE) of the estimated state variable. However, Kalman Filter is still able to give a very good estimate even if the noises are not Gaussian.

3 The Proposed Single-Solution Simulated Kalman Filter (ssSKF) Algorithm

The single-solution Simulated Kalman Filter (ssSKF) algorithm is a single agent version of the population-based Simulated Kalman Filter algorithm. It is inspired by the estimation capability of the Kalman Filter. Principally, ssSKF embraces the same principle as SKF. However, instead of relying on a population of agents to estimate the optimum solution, ssSKF uses only a single agent. The ssSKF algorithm depends on the efficiency of the Kalman Filter algorithm itself to iteratively improve its estimation.

Like SKF, the ssSKF algorithm attempts to solve optimisation problems by iteratively estimating the optimum solution using the scalar model of Discrete Kalman Filter framework. By using this model, the state vector, $\mathbf{X}$, holds the agent's estimated position, which is a scalar value for each dimension in the search space. This estimated state variable is used in the calculation of fitness based on the specific objective function. The use of the scalar model also reduces the computation complexity because all the matrices can be reduced to scalar values.

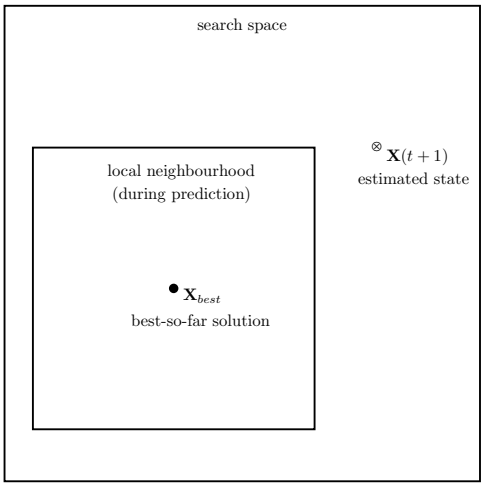


Figure 1. Local neighbourhood concept in ssSKF.

In ssSKF, the algorithm starts with the initialisation of the single agent. This agent represents a Kalman Filter. Next, fitness of the agent is evaluated. During each iteration, $\mathbf{X}_{best}$, which holds the best-so-far solution, is updated. The single agent in ssSKF algorithm iteratively improves its estimation by using the standard Kalman Filter framework which comprises of predict, measure, and estimate steps. An adaptive neighbourhood is employed to make prediction during the prediction step. The measurement, guided by the best-so-far solution, $\mathbf{X}_{best}$, is simulated during the measurement step. Finally, the agent makes an improved estimation during the estimation step, by using the predicted and the measured values under the influence of the calculated Kalman gain. This process continues until the stopping condition is met.

Besides operating with only a single agent, as opposed to a population of agents in SKF, the ssSKF differs from the population-based SKF in its prediction step. The idea of having a prediction step is to make a best guess on the location of the optimal solution. This element is missing in the original SKF algorithm. In the view of having $\mathbf{X}_{best}$ as the best-so-far solution, it is wise to predict that the position of the optimum solution is somewhere near the $\mathbf{X}_{best}$. Therefore, in ssSKF, a decreasing local neighbourhood is employed during the prediction step to further exploit the information. The decreasing neighbourhood mechanism can be seen as a type of adaptive step-size adjustment process, used by many other single solution-based metaheuristic algorithms. The idea of having a local neighbourhood during the prediction step is illustrated in Fig. 1 in 2D representation.

Considering the search space, S, of an optimisation problem, given by $[lowerlimit, upperlimit]$, the local neighbour-

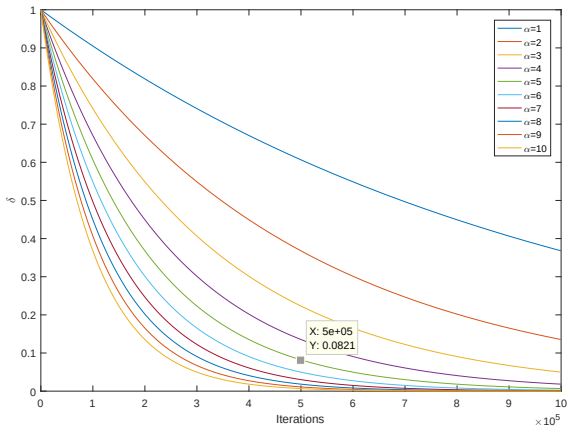


Figure 2. Plot of δ for different α values.

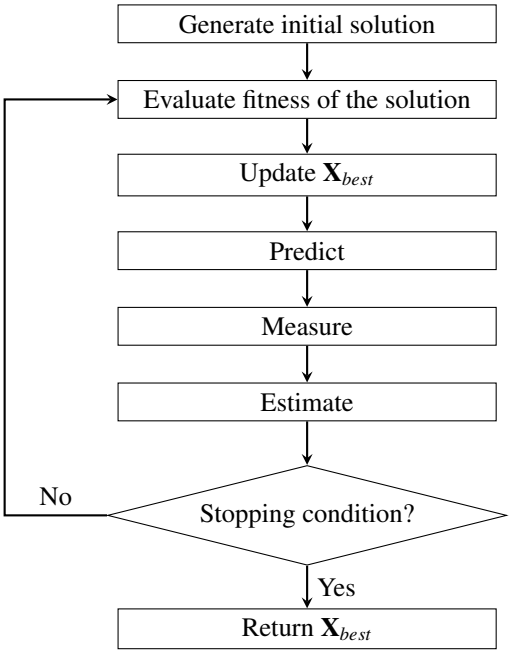


Figure 3. The ssSKF Algorithm.

hood, N_S, which is centred around $\mathbf{X}_{best}$ is defined to be $[\mathbf{X}^d_{best} - \delta, \mathbf{X}^d_{best} + \delta]$. The size of the local neighbourhood is determined by the step-size, δ, which is adaptively decreasing following (6).

$$\delta = \exp^{-\alpha \times t/tMax} \times \delta_0 \tag{6}$$

In an iteration t, once the $\mathbf{X}_{best}$ has been updated, the search strategy in ssSKF starts by predicting the location of the optimum solution. By using the concept of local neighbourhood during the prediction step, the single agent in ssSKF makes a prediction that the optimum solution resides in a confined neighbourhood of $[\mathbf{X}^d_{best} - \delta, \mathbf{X}^d_{best} + \delta]$ with $\mathbf{X}_{best}$ to be the centre of the neighbourhood.

The initial neighbourhood limit, δ_0, is selected to be $\delta_0 = max(|lowerlimit|, |upperlimit|)$ to ensure maximum coverage of the search space during the first iteration. Fig. 2 shows the plot of the step-size, δ, for different adaptive coefficient, α, over $1,000,000$ iterations, with t to be the iteration number, and δ_0 is set at 100. From Fig. 2, we can see that a small α value will lead to a linear decrement while a larger α value will lead to a faster convergence. In ssSKF, α is chosen to be 5, to allow exploration to take place during the first half of the iterations while focusing on exploitation in the neighbourhood of around 10% of the original search space for the remaining half iterations.

The ssSKF algorithm is illustrated in Fig. 3.

3.1 *Initialisation*

The initialisation of the ssSKF algorithm is almost similar to the population-based SKF, where $\mathbf{X}(0)$ is randomly initialised according to search space of the problem to be solved. However, instead of initialising multiple agents, only a single agent is initialised in ssSKF. The initial error covariance, $P(0)$, is set to a normally distributed random number instead of using the suggested value of 1000 [11]. The normally distributed random number is denoted as $randn^d$. This value is generated in every dimension.

3.2 *Fitness evaluation and* $\mathbf{X}_{best}$ *update*

At the beginning of each iteration, the fitness of the solution is evaluated. Then, according to type of the problem, the best-so-far solution, $\mathbf{X}_{best}$, is updated. The $\mathbf{X}_{best}$ is updated only if $\mathbf{X}(t)$, which holds the estimation in the corresponding iteration is a better solution. For minimisation problem, $\mathbf{X}_{best}$ will be updated if the fitness of $\mathbf{X}(t)$ is less than the fitness of $\mathbf{X}_{best}$. While, for maximisation problem, $\mathbf{X}_{best}$ will be updated if the fitness of $\mathbf{X}(t)$ is greater than the fitness of $\mathbf{X}_{best}$.

3.3 *Predict, measure, and estimate*

The search strategy in ssSKF mimics the cycle in Kalman Filter estimation method. Kalman Filter is conceptualised as having two distinct phases: prediction and estimation. Kalman Filter works recursively with prediction advancing the state prior to the next measurement. Then, the estimation phase incorporates the new measurement to improve the state estimation. Thus, there are three steps in the ssSKF search strategy: prediction, measurement, and estimation.

During prediction, the following predict equations are used to make prediction of the optimum solution. Instead of using the suggested value of 0.5 for the process noise, Q [11], a normally distributed random number, $randn^d$, is used whenever the parameter value is needed for each dimension in every iteration.

$$\mathbf{X}^d(t|t+1) \sim U[\mathbf{X}^d_{best} - \delta, \mathbf{X}^d_{best} + \delta] \tag{7}$$

$$P^d(t|t+1) = P^d(t) + randn^d \tag{8}$$

Prediction in ssSKF is carried out by using (7) instead of (1). Equation (7) indicates that the prediction is carried out in

a local neighbourhood surrounding the best-so-far solution, $\mathbf{X}^d_{best}$. The local neighbourhood, N_S, is adaptively reduced in size as the iteration increases. This adaptive reduction of the step-size causes the algorithm to move from exploration to exploitation during the prediction step itself. In (8), $randn^d$ is used to replace the parameter $Q(t)$ in (2).

The next step is measurement. Measurement in ssSKF is simulated in a similar manner as in the population-based SKF algorithm. The best-so-far solution, $\mathbf{X}^d_{best}$, steered the agent's simulated measurement value, $\mathbf{Z}^d(t)$, by using (9).

$$\mathbf{Z}^d(t) = \mathbf{X}^d(t|t+1) + sin(rand^d \times 2\pi) \times \mid \mathbf{X}^d(t|t+1) - \mathbf{X}^d_{best} \mid \quad (9)$$

The purpose of measurement is to give feedback to the estimation process. According to (9), the measurement process is simulated in such a way that the measured value of the agent may take any random value surrounding the predicted value, $\mathbf{X}^d(t|t+1)$, either approaching to or moving away from the best-so-far solution, $\mathbf{X}^d_{best}$. The random element, $rand^d$, which is uniformly distributed in the range of $[0, 1]$, is responsible for the stochastic nature in SKF. The sine function in (9) on the other hand, is the one that causes the measured value to either being located closer to or distant from the best-so-far solution, $\mathbf{X}^d_{best}$, balancing between exploration and exploitation. The exploration and exploitation mechanism in SKF is further compromised as the distance between the predicted value, $\mathbf{X}^d(t|t+1)$, and the best-so-far solution, $\mathbf{X}^d_{best}$, decreases with the increase of the number of iteration.

Finally, the estimation step. During estimation, the state and error covariance estimates for the next iteration are calculated using the esimate equations right after the calculation of the Kalman gain. The estimate equations are used to improve the estimation by incorporating the simulated measurement value into the predicted value, with the influence of Kalman gain.

The measurement noise, $R(t)$, which involves in the calculation of the Kalman gain, $K(t)$ in (5), is given a normally distributed random number $randn^d$, defined in the range of between 0 to 1 with a mean of 0.5, generated for every dimension every time it is needed, as depicted in (10). Kalman gain acts as a weighted average between the prediction and measurement to produce a better estimated value for the next time step, $\mathbf{X}^d(t+1)$, as shown in (11). The corresponding error covariance, $P^d(t+1)$, is estimated to be reduced due to the effect of Kalman gain in (12).

$$K^d(t) = (P^d(t|t+1))/(P^d(t|t+1) + randn^d) \quad (10)$$

$$\mathbf{X}^d(t+1) = \mathbf{X}^d(t|t+1) + K^d(t) \times (\mathbf{Z}^d(t) - \mathbf{X}^d(t|t+1)) \quad (11)$$

$$P^d(t+1) = (1 - K^d(t)) \times P^d(t|t+1) \quad (12)$$

At the end of the estimation step, a better estimation for the next iteration that lies between the predicted and the measured value is produced. This process continues until the stopping condition is met.

4 Experiments

The CEC 2014's benchmark suite [35] is chosen to evaluate the performance of the ssSKF algorithm. There are 3 unimodal functions, 13 simple multimodal functions, 6 hybrid functions and 8 composition functions, which make up a total of 30 benchmark functions in the CEC 2014's benchmark suite. All these benchmark functions are minimisation type of optimisation problems.

The performance of the ssKSF is benchmarked against the population-based Simulated Kalman Filter algorithm, two state-of-art algorithms, the Particle Swarm Optimisation and the Genetic Algorithm, and two new nature-inspired algorithms, the Black Hole and the Grey Wolf Optimiser. The stopping condition is set at $1,000,000$ number of function evaluations. The complexity of the benchmark functions is set at 50 dimensions and the experiments are carried out for 50 times. The initialisation and parameter setting for all tested algorithms are listed in Table 1. All algorithms are implemented in MATLAB. The MATLAB codes for the CEC 2014's benchmark suite including the PSO algorithm are available in [36]. In the experiments, the parameter setting for PSO is set according to [37]. The algorithm employed for GWO is adapted from [38] and the algorithm for GA is adapted from [39]. BH and SKF algorithm on the other hand are coded according to the principle of search cited in their respective literature.

The results of the experiments are compared statistically by using the mean and the standard deviation for each benchmark function. Due to space limitation, only selected functions are chosen to be presented graphically. Convergence curves of the algorithms for selected functions are presented to show the comparison in the convergence behaviour of the algorithms in solving different type of benchmark problems. They are plotted for every 100 function evaluations to cater for the difference between single solution-based algorithm and population-based algorithms of 100 agents. Boxplots for the selected functions are provided to give a graphical representation of the data distribution by each algorithm. The use of boxplots is a convenient way to display surprisingly high maximums and low minimums, also known as outliers, produced by the algorithms.

A Friedman statistical test is then carried out to test for differences in performance between the algorithms at 5% significance level. To control the familywise error rate, the Holm procedure is chosen. These analyses are performed by using the KEEL software [40].

5 Results and Discussion

This section presents the results of the proposed ssSKF algorithm, compared against the population-based SKF, BH, PSO, GWO and GA algorithms using the CEC 2014's benchmark suite. Table 2 - 5 present the mean and standard deviation achieved by all tested algorithms for unimodal, simple multimodal, hybrid and composition benchmark functions respectively. The numbers written in **bold** indicates the best

Table 1. Initialisation and Parameter Settings for ssSKF, SKF, BH, PSO, GWO and GA

Algorithm	Initialisation (random)	Parameter settings
ssSKF	Initial state estimate	Number of agent = 1
		Adaptive coefficient, α = 5
SKF	Initial state estimate	Number of agents = 100
		Initial error covariance estimates, $P(0)$ = 1000
		Process error, Q = 0.5
		Measurement error, R = 0.5
BH	Initial stars	Number of stars = 100
PSO	Initial positions and velocities	Swarm size = 100
		Initial inertia weight = 0.9
		Final inertia weight = 0.4
		Cognitive acceleration factor, $c1$ = 2
		Social acceleration factor, $c2$ = 2
		$[V_{min}, V_{max}] = [X_{min}, X_{max}]$
GWO	Initial search agents	Number of agents = 100
		Adaptive parameter, a = linear decrease from 2 to 0
		Coefficient vector, $C = random[0, 2]$
GA	Initial population	Population size = 100
		Mutation probability = 0.2
		Crossover probability = 0.5

mean value obtained for the corresponding objective function among all tested algorithms.

5.1 *Unimodal Functions*

The results in Table 2 shows an exceptional performance of the ssSKF algorithm in solving function no. 3 compared to the other algorithms. The ssSKF algorithm managed to converge near to the ideal fitness of 300 with a very small standard deviation value. This achievement is far better than the population-based SKF algorithm and the other tested algorithms. Benchmark function no. 3 is a rotated discus function, where the properties are non-separable with one sensitive direction.

Both function no. 1 and no. 2 are also rotated and non-separable unimodal functions, but are a bit harder to be solved. Fig. 4 shows the convergence curves and boxplots comparison in solving for function no. 1. Function no. 1 is a rotated high-conditioned elliptic function. The quadratic ill-conditioned property of the function makes it hard to be solved to optimality. The ssSKF algorithm however, still performs competitively compared to the other algorithms, especially to PSO, GWO and GA algorithm, with a small deviation from the median value as can be seen in Fig. 4b. We can also see from the convergence curve of the ssSKF algorithm that once the agent enters the second half of the iterations, exploitation kicks in to refine the estimation.

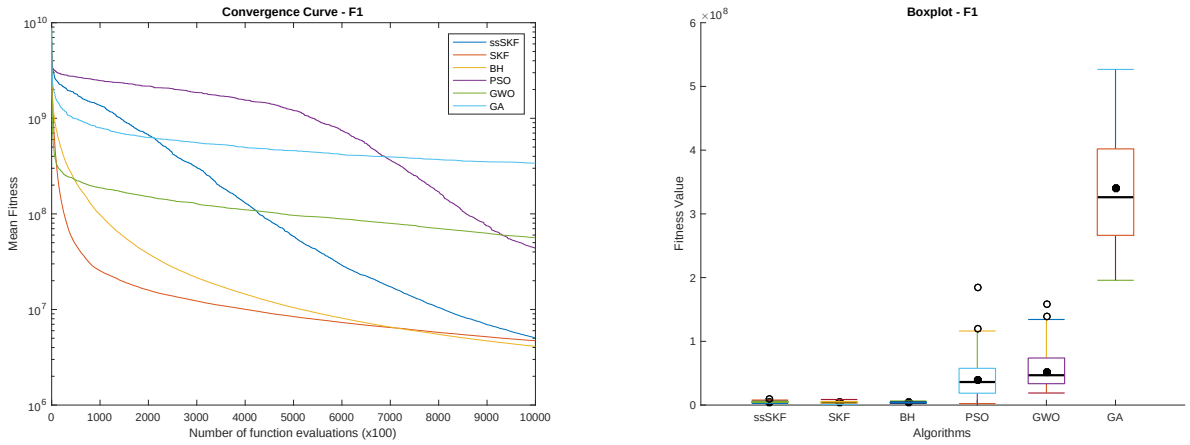
5.2 *Simple Multimodal Functions*

The results in Table 3 shows that although ssSKF managed to outperform the others in only 3 out of 13 simple multimodal functions, the results for the other functions are very competitive. The ssSKF algorithm managed to solve most of the problems closer to optimality. This shows that even with only a single agent, ssSKF algorithm managed to escape from the local optima stagnation problem.

Function no. 4 is selected to show the convergence behaviour of the ssSKF algorithm in comparison to the other al-

Table 2. Experimental Results Comparison (Unimodal Functions)

No.	ssSKF	SKF	BH	PSO	GWO	GA
1	5.02E+06	4.70E+06	**4.11**E+06	4.35E+07	5.63E+07	3.40E+08
	±1.30E+06	±1.84E+06	±82.5E+06	±3.45E+07	±3.11E+07	±0.845E+08
2	1.34E+07	2.45E+07	**1.93**E+05	1.14E+07	5.27E+09	2.36E+10
	±0.136E+07	±5.65E+07	±2.57E+05	±6.72E+07	±3.18E+09	±0.339E+10
3	**367.88** ± 11.909	18148 ± 8044.5	11558 ± 1989.3	9934.1 ± 9628.3	49774 ± 13648	62700 ± 10493



(a) Convergence curve (b) Boxplot

Figure 4. Experimental results comparison for function no. 1 (rotated high-conditioned elliptic function)

gorithms in solving simple multimodal problem, as shown in Fig. 5a. Function no. 4 is a shifted and rotated rosenbrock's function. Besides being multimodal and non-separable, this benchmark function has a very narrow valley from a local optimum to the global minimum. Based on the convergence behaviour of the algorithms, we can see that the ssSKF algorithm able to do exploitation better although converge a bit later than the population-based SKF, BH, GWO and GA. When the performance of ssSKF is compared graphically to other tested algorithms, we can see from Fig. 5b that ssSKF algorithm has a very good and consistent performance depicted by the position of the boxplot and its size.

5.3 *Hybrid Functions*

The ssSKF algorithm managed to outperform the other tested algorithms the most in solving for hybrid benchmark problems. Hybrid functions mimic real-world optimisation problems, as the variables in the hybrid functions are randomly divided into subcomponents. Then, different basic functions are applied to these different subcomponents making each of them having different properties. From Table 4, it can be seen that ssSKF algorithm performs the best compared to other tested algorithms in all but one hybrid function.

The convergence curve and boxplot for function no. 17 are selected for comparison purposes. Function no. 17 is a hybrid of 3 basic functions which is the modified shwefel's function, the rastrigin's function and the high conditioned elliptic function. It can be seen from Fig. 6a that the exploration phase ends a bit early in ssSKF algorithm, however, further improvements are carried out during the exploitation phase. We can also see from Fig. 6b that ssSKF algorithm has a very consistent performance throughout the 50 runs.

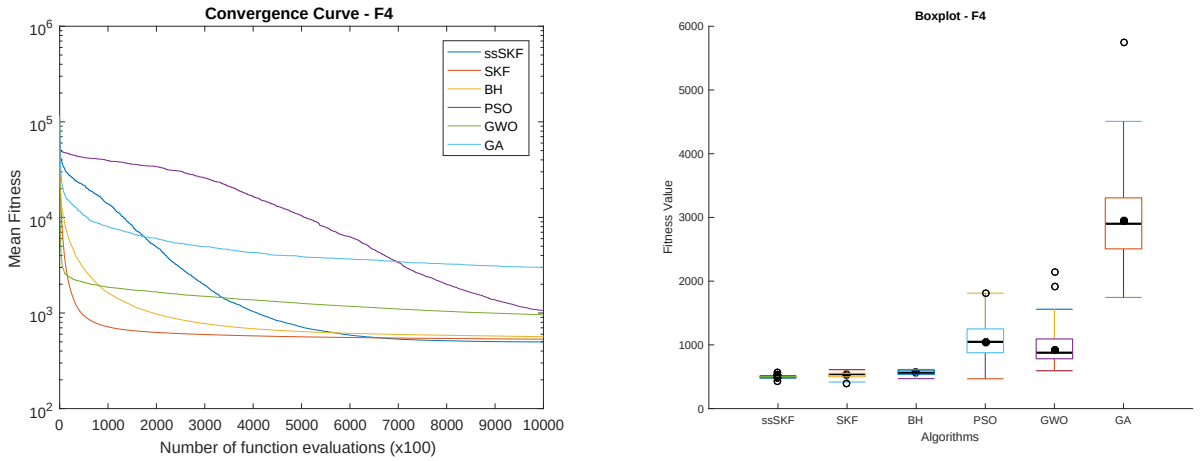
5.4 *Composition Functions*

Composition functions are used to test the algorithms' tendency to converge to the centre of the search space. A local optimum is set to the origin as a trap for each composition functions.

Results in Table 5 shows that the ssSKF algorithm outperform the other tested algorithms for 3 out of 8 composi-

Table 3. Experimental Results Comparison (Simple Multimodal Functions)

No.	ssSKF	SKF	BH	PSO	GWO	GA
4	**498.46** ± 17.462	532.77 ± 42.776	564.79 ± 40.504	1062.1 ± 277.16	958.42 ± 288.82	3008.5 ± 717.97
5	521.11 ± 0.0323	**520.01** ± 0.0196	**520.01** ± 0.0210	521.06 ± 0.0594	521.11 ± 0.0319	521.01 ± 0.0592
6	**619.91** ± 4.5744	633.44 ± 3.9608	658.13 ± 5.0154	631.49 ± 5.2445	625.95 ± 3.1894	655.83 ± 2.7594
7	701.13 ± 0.0108	700.25 ± 0.9076	700.13 ± 0.0801	**700.02** ± 0.0334	745.23 ± 27.639	924.79 ± 32.65
8	974.5 ± 39.406	**807.98** ± 5.3569	922.25 ± 14.311	858.72 ± 12.203	975.3 ± 28.277	1067.9 ± 18.924
9	1082.2 ± 38.155	1059.1 ± 27.703	1212.1 ± 42.612	**1051.7** ± 29.129	1078.9 ± 27.883	1400.3 ± 35.998
10	5769.1 ± 737.4	**1335.2** ± 159.62	3121 ± 546.24	1644 ± 227.67	6381 ± 645.94	6254.2 ± 492.28
11	**6240.5** ± 902.72	6249.4 ± 641.48	8051 ± 876.77	11966 ± 2741.9	6582.8 ± 1287.8	12793 ± 540.93
12	1201 ± 0.34346	**1200.2** ± 0.0825	1200.7 ± 0.2264	1202.6 ± 0.4937	1202 ± 1.5515	1202.2 ± 0.2794
13	1300.6 ± 0.1343	1300.6 ± 0.0873	**1300.5** ± 0.0367	1300.6 ± 0.1127	1300.6 ± 0.2986	1302.8 ± 0.5706
14	1400.4 ± 0.3032	**1400.3** ± 0.0390	**1400.3** ± 0.0152	**1400.3** ± 0.0683	1407.3 ± 8.3571	1461.6 ± 11.996
15	1532.6 ± 3.5675	1551.7 ± 16.126	1787.8 ± 50.267	**1528.8** ± 8.9937	1965.3 ± 548.62	35514 ± 27831
16	1620.8 ± 0.6714	**1619.1** ± 0.8810	1621.5 ± 0.6899	1621.7 ± 0.6907	1619.5 ± 1.0406	1621.9 ± 0.3835



(a) Convergence curve (b) Boxplot

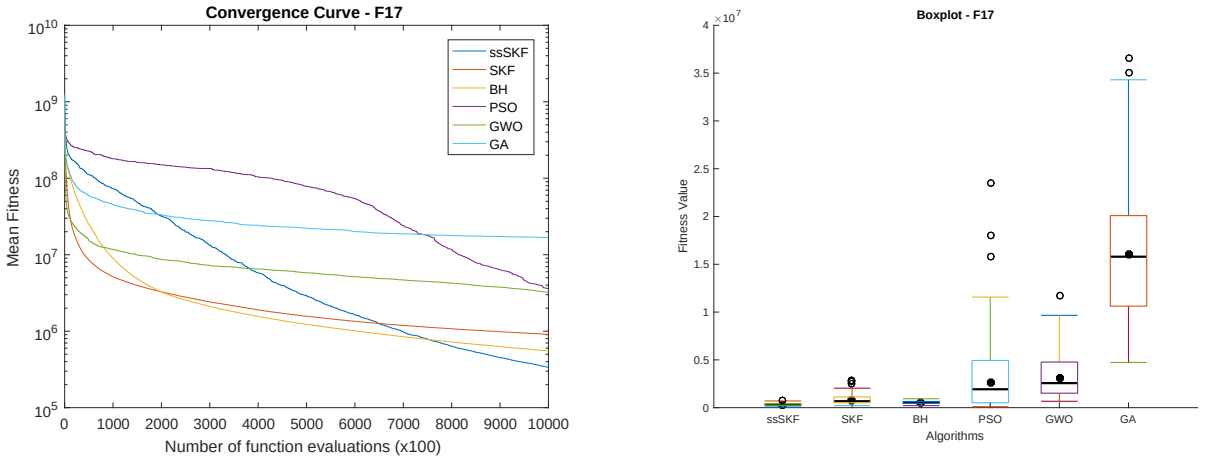
Figure 5. Experimental results comparison for function no. 4 (shifted and rotated rosenbrock´s function)

tion benchmark functions, specifically for function no. 25, 27 and 28. The ssSKF algorithm also gives a very high competition for the other composition functions, especially in solving function no. 23, 24 and 26. These results exhibit the ssSKF's ability in avoiding the centre trap while searching for the optimum position.

Fig. 7a gives a graphical view of the convergence behaviour of the ssSKF algorithm in comparison to the other tested algorithms when solving benchmark function no. 23. This is followed by Fig. 7b which shows the results distri-

Table 4. Experimental Results Comparison (Hybrid Functions)

No.	ssSKF	SKF	BH	PSO	GWO	GA
17	**3.36**E+05	9.08E+05	5.52E+05	3.59E+06	3.23E+06	1.68E+07
	±1.41E+05	±6.16E+05	±1.60E+05	±4.79E+06	±2.10E+06	±0.818E+07
18	3.64E+05	6.94E+07	**2433.4**	30741	4.82E+07	5.48E+06
	±76796	±18.6E+07	±269.07	±1.46E+05	±10.1E+07	±3.08E+06
19	**1920** ± 2.5822	1950.2 ± 31.019	1952.8 ± 32.801	1962.3 ± 29.926	1979.5 ± 32.789	2004.5 ± 18.806
20	**2488.9** ± 96.724	34799 ± 13337	8499.6 ± 2606.1	6513.6 ± 2534.9	14603 ± 5844.8	35020 ± 14191
21	**2.55**E+05	1.19E+06	3.95E+05	7.10E+05	1.92E+06	5.30E+06
	±1.15E+05	±0.625E+06	±1.16E+05	±8.06E+05	±2.31E+06	±2.76E+06
22	**2806.3** ± 260.07	3429.1 ± 306.4	3708.2 ± 349.39	3421.4 ± 331.98	2873.3 ± 221	3429 ± 249.95



(a) Convergence curve (b) Boxplot

Figure 6. Experimental results comparison for function no. 17 (hybrid function 1)

bution comparison between algorithms tested in solving the same benchmark problem over 50 runs. From Fig. 7a, we can see the improvement of the estimation continues from the exploration phase throughout the exploitation phase which is the second half of the total iterations. Performance wise, compared to SKF, the ssSKF algorithm has a better consistency than the population-based SKF, depicted by a narrower boxplot as in Fig. 7b.

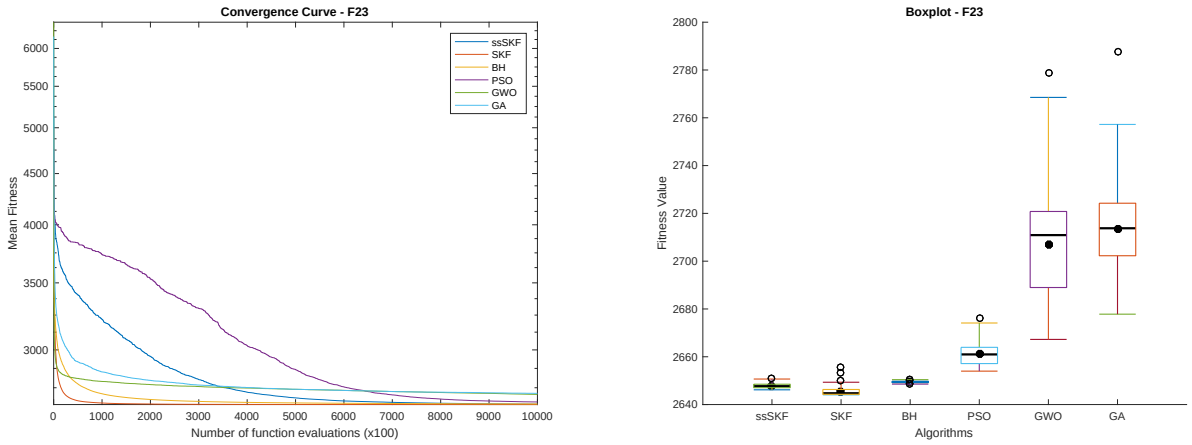
5.5 *Statistical Analyses*

Statistically, the ssSKF algorithm performs better compared to the other tested algorithms. The average ranking according to Friedman 1×5 statistical test is given by Table 6. Friedman test ranks ssSKF algorithm the highest, followed by the population-based SKF algorithm, BH, PSO, GWO and then the GA algorithm.

According to the Friedman test statistic of 47.319048, distributed according to the χ^2-distribution with 5 degrees of freedom, a significant difference exists between the algorithms. In order to test which algorithm is significantly

Table 5. Experimental Results Comparison (Composition Functions)

No.	ssSKF	SKF	BH	PSO	GWO	GA
23	2647.9 ± 1.065	**2645.7** ± 2.2752	2649.5 ± 0.4324	2661.5 ± 5.2579	2708.3 ± 21.85	2714.8 ± 18.45
24	2676.4 ± 6.2471	2667.2 ± 5.8049	2666.4 ± 9.8025	2672.8 ± 6.819	**2600** ± 0.0002	2777.2 ± 11.155
25	**2711.9** ± 1.9048	2730.4 ± 3.716	2750.5 ± 7.9287	2729.8 ± 4.7321	2725.3 ± 6.4155	2761.2 ± 10.72
26	2710.6 ± 30.387	2766.4 ± 47.794	2792.2 ± 27.298	**2700.5** ± 0.1206	2769.2 ± 54.188	2702.7 ± 0.4142
27	**3516.2** ± 119.07	3883.3 ± 123.41	4654.8 ± 263.57	3843 ± 204.98	3672.9 ± 95.244	4473.2 ± 72.069
28	**4621.5** ± 399.55	7223.4 ± 1153.5	11048 ± 1068.1	9891.7 ± 1869.1	4647 ± 457.53	6288.7 ± 461.05
29	76422	**5997.8**	10361	23202	3.28E+06	6.72E+06
	±20519	±5310.7	±1668.5	±1.07E+05	±8.34E+06	±4.09E+06
30	47030	**19753**	58613	1.95E+05	1.12E+05	1.62E+05
	±10475	±3511.6	±4320	±1.17E+05	±68623	±55409



(a) Convergence curve (b) Boxplot

Figure 7. Experimental results comparison for function no. 23 (composition function 1)

better than the other, Holm procedure is carried out. Post-hoc Holm's procedure control the family-wise error rate by adjusting the rejection criteria of each of the individual hypotheses. The result of the analysis is given by Table 7.

Based on Table 7, the Holm's procedure rejects those hypotheses that have an unadjusted p-value less than 0.016667. Thus, the ssSKF algorithm can be concluded as performing significantly better than the GWO and GA algorithms in solving the CEC 2014 benchmark problems.

5.6 *Exploration and exploitation in ssSKF*

Although it is statistically proven that the ssSKF algorithm is performing better than the population-based SKF, BH, PSO, GWO and GA algorithm in the testing framework, the search behaviour in terms of exploration and exploitation of the single agent during optimisation should be observed to confirm its performance when solving for other problems, especially the real-world problems. Thus, for this purpose, the ssSKF algorithm is tested on the two-dimensional version of the

Table 6. Friedman Average Rankings of the Algorithms

Algorithm	Friedman Rank
ssSKF	2.4667
SKF	2.7667
BH	3.1833
PSO	3.2167
GWO	4.0000
GA	5.3667

Table 7. Post Hoc Holm's Analysis of ssSKF, SKF, BH, PSO, GWO and GA (α = 0.05)

i	Algorithms	z	p	Holm
5	ssSKF vs. GA	6.00357	0	0.01
4	ssSKF vs. GWO	3.174302	0.001502	0.0125
3	ssSKF vs. PSO	1.552648	0.120507	0.016667
2	ssSKF vs. BH	1.483641	0.137904	0.025
1	ssSKF vs. SKF	0.621059	0.534561	0.05

benchmark problems. Another round of experiment is conducted on two-dimensional problems of CEC 2014's benchmark functions except for hybrid functions and function no. 29 and 30, because these functions are not defined in two-dimensional domain. The maximum number of iteration is set to 100 iterations only. The results of some selected functions are presented graphically to show how the agent behaves over the course of iterations.

In order to observe the exploration and exploitation of the search agent in particular, the trajectory of the search agent in both dimensions are captured and presented in Fig. 8. From these figures, we can see clearly that the agent undergoes extensive exploration during the first half of the total iterations and then decrease gradually to exploit better around the improved best-so-far solution due to the effect of adaptively decreasing step size, δ, as given in (6).

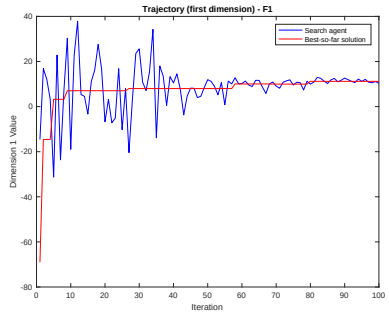
To further observe how the search agent moves around the search space, search history of search agent for selected functions are provided in Fig. 9. The search history marks the visited locations of the search agent in the search space throughout the optimisation process, plotted on the contour map of the specific problem. We can observe from these figures that there is sufficient exploration conducted by the search agent especially for multimodal problem in order to avoid being trapped in local optima before converging around the best solution obtained. Finally, Fig. 10 shows the fitness trend obtained by the search agent throughout the course of iterations. High fluctuations in the fitness value obtained during the first half of the search process is preferable because it confirms the extensive exploration behaviour in the ssSKF algorithm, while the low changes observed after that confirms the exploitation behaviour in the ssSKF algorithm. These fluctuations are the results of the stochastic element during the measurement step given in (9). As the iteration increases, the exploration is being compromised by exploitation due to the fact that the agent is predicted to move closer to the best-so-far solution. The best-so-far solution, $\mathbf{X}_{best}$, exhibit a descending pattern which reflects the convergence in finding the optimal solution of the minimisation problem. All these evidences prove the ability of the ssSKF algorithm to solve optimisation problem by iteratively estimating the optimum solution.

6 Conclusion

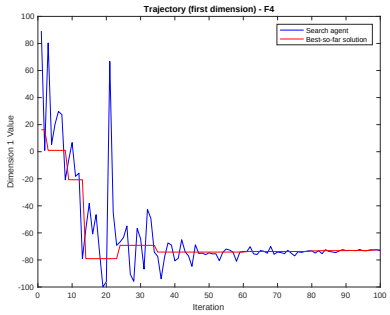
In this paper, a novel single solution-based metaheuristic optimisation algorithm, named ssSKF, is introduced to solve real-valued, numerical optimisation problem. The ssSKF algorithm utilises a decreasing neighbourhood mechanism together with the Kalman Filter framework in order to improve its estimation of the optimum. The ssSKF algorithm reduces number of parameters needed as the $P(0)$, Q and R values are replaced with normally distributed random numbers in the range of [0,1] with 0.5 mean, generated for every dimension whenever the parameters' value are required. However, an adaptive coefficient, α, is introduced for the adaptive neighbourhood, with a suggested value of 5. The ssSKF algorithm is implemented to CEC 2014 benchmark suite which consists of unimodal, simple multimodal, hybrid and composition functions of 50 dimensions. The performance of ssSKF algorithm is benchmarked against the population-based SKF algorithm, BH, PSO, GWO and GA algorithms. The results show that despite being a single solution-based algorithm, ssSKF managed to outperform population-based algorithms under the same number of function evaluations.

Acknowledgment

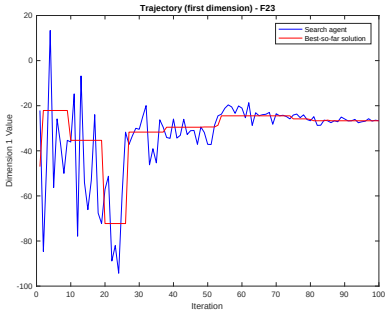
This research was financially supported by Fundamental Research Grant Scheme (FRGS) awarded by the Ministry of Education (MOE) to Multimedia University under grant no. FRGS/1/2015/TK04/MMU/03/02 and Universiti Malaysia Pahang under grant no. RDU140114. We would like to thank Multimedia University and Universiti Malaysia Pahang for providing all the facilities required for this study.



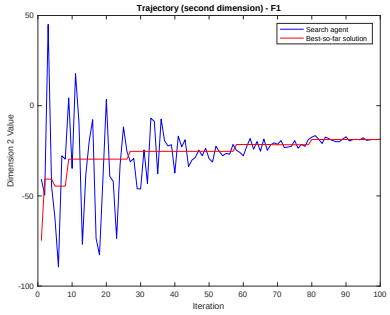
(a) Unimodal - first dimension



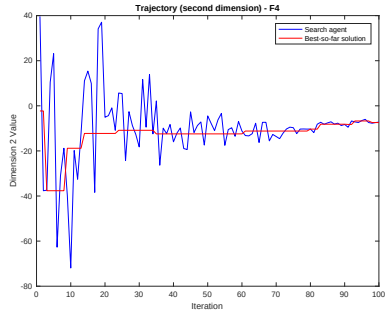
(b) Simple multimodal - first dimension



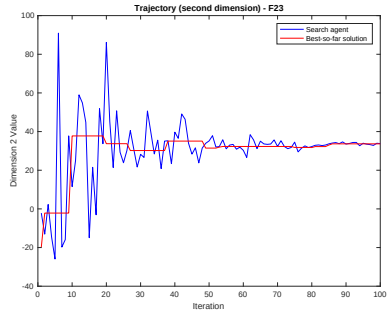
(c) Composition - first dimension



(d) Unimodal - second dimension



(e) Simple multimodal - second dimension



(f) Composition - second dimension

Figure 8. Trajectory of the ssSKF's agent during optimisation of benchmark functions.

References

[1] E. Talbi, *Metaheuristics*. Hoboken, NJ: John Wiley & Sons, 2009.

[2] S. Droste, T. Jansen and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box Optimisation", *Theory of Computing Systems*, vol. 39, no. 4, pp. 525–544, Jul. 2006.

[3] J. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA: The MIT Press, 1992.

[4] J. Kennedy and R. Eberhart, "Particle swarm Optimisation," in *Proc. IEEE International Conference on Neural Networks*, Dec. 1995, pp. 1942–1948.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for Optimisation," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[6] I. Boussaid, J. Lepagnot, P. Siarry, "A survey on Optimisation metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, Jul. 2013.

[7] J. A. Parejo, A. Ruiz-Cortes, S. Lozano, P. Fernandez, "Metaheuristic Optimisation frameworks: a survey and benchmarking," *Soft Computing*, vol. 16, no. 3, pp. 527–561, Mar. 2012.

[8] I. Fister Jr. *et al.*, "A brief review of nature-inspired algorithms for Optimisation," *Elektrotehniski Vestnik [English]*, vol. 80, no. 3, pp. 1–7, 2013.

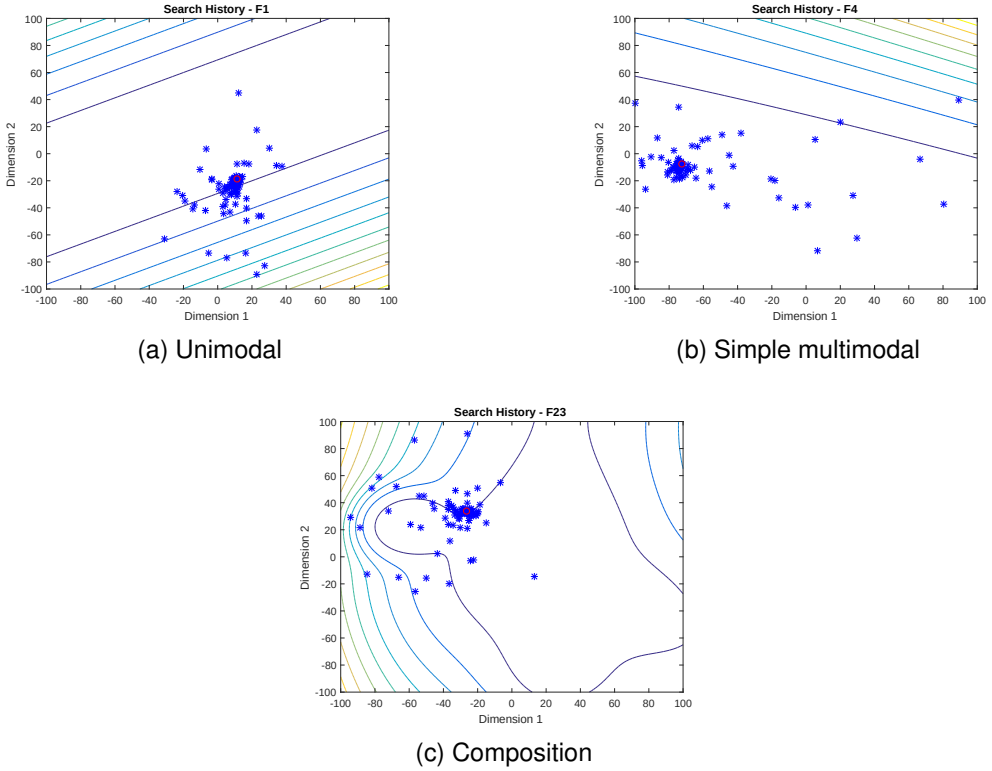


Figure 9. Search history of the ssSKF's agent during optimisation of benchmark functions.

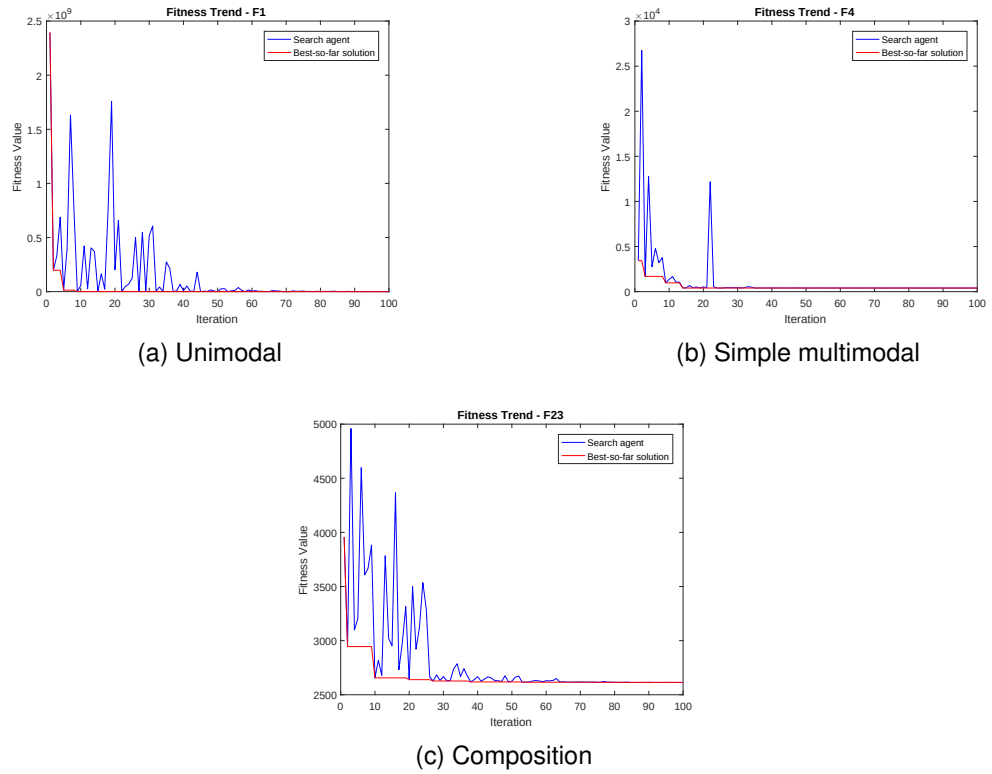


Figure 10. Fitness trend of the ssSKF's agent during optimisation of benchmark functions.

[9] A. Hatamlou, "Black hole: A new heuristic Optimisation approach for data clustering", *Information Sciences*, vol. 222, pp. 175–184, Feb. 2013.

[10] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey Wolf Optimiser", *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014.

[11] Z. Ibrahim *et al.*, "A Kalman filter approach for solving unimodal Optimisation problems", *ICIC Express Lett.*, vol. 9, no. 12, pp. 3415–3422, Dec. 2015.

[12] S. Kirkpatrick, C. D. Gelatt Jr. and M. Vecchi, "Optimisation by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[13] F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[14] N. Mladenovic and P. Hansen, "Variable neighborhood search", *Computers & Operations Research*, vol. 24, no. 11, pp. 1097-1100, Nov. 1997.

[15] M. Dorigo, "Optimisation, Learning and Natural Algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[16] X. Yang, "Firefly algorithm, stochastic test functions and design optimisation", *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, Mar. 2010.

[17] X. Yang and S. Deb, "Cuckoo Search Via Levy Fight," in *Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, India, Dec. 2009, pp 210–214.

[18] S. Mirjalili, "SCA: A Sine Cosine Algorithm for solving Optimisation problems", *Knowledge-Based Systems*, vol. 96, pp. 120–133, Mar. 2016.

[19] Z. Ibrahim *et al.*, "Simulated Kalman Filter: A Novel Estimation-Based Metaheuristic Optimisation Algorithm," *Advance Science Lett.*, vol. 22, no. 10, pp. 2941–2946, Oct. 2016.

[20] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *Journal of Basic Engineering*, vol. 82(Series D), pp. 35–45, 1960.

[21] Z. Md Yusof *et al.*, "Angle Modulated Simulated Kalman Filter algorithm for combinatorial Optimisation problems," *ARPN Journal of Engineering and Applied Science*, vol. 11, no. 7, pp. 4854-4859, Apr. 2016.

[22] Z. Md Yusof *et al.*, "Distance Evaluated Simulated Kalman Filter algorithm for combinatorial Optimisation problems," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 7, pp. 4911-4916, Apr. 2016.

[23] Z. Md Yusof, I. Ibrahim, S. N. Satiman, Z. Ibrahim, N. H. Abdul Aziz and N. A. Ab. Aziz, "BSKF: Binary Simulated Kalman Filter," in *Proc. of 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, Dec. 2015, pp. 77-81.

[24] B. Muhammad *et al.*, "A new hybrid Simulated Kalman Filter and Particle Swarm Optimisation for continuous numerical Optimisation problems," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 22, pp. 17171-17176, Dec, 2015.

[25] B. Muhammad, Z. Ibrahim, K. Z. Mohd Azmi, K. H. Abas, N. A. Ab. Aziz, N. H. Abdul Aziz and M. S. Mohamad, "Four different methods to hybrid simulated Kalman filter (SKF) with gravitational search algorithm (GSA)," in *Proc. of 3rd National Conference of Postgraduate Research*, Sep. 2016, pp. 854–864.

[26] N. H. Abdul Aziz, N. A. Ab. Aziz, Z. Ibrahim, S. Razali, K. H. Abas and M. S. Mohamad "A Kalman Filter approach to PCB drill path Optimisation problem," in *Proc. of IEEE Conference on Systems, Process and Control*, Dec. 2016, pp. 33–36.

[27] Z. Md Yusof, S. N. Satiman, B. Muhammad, S. Razali, Z. Ibrahim, Z. Aspar and S. Ismail, "Solving airport gate allocation problem using Simulated Kalman Filter," in *Proc. of International Conference on Knowledge Transfer (ICKT'15)*, Malaysia, Dec. 2015, pp. 121–127.

[28] K. Z. Mohd Azmi, Z. Md Yusof, S. N. Satiman, Z. Ibrahim, N. A. Ab. Aziz and N. H. Abdul Aziz, "Solving airport gate allocation problem using angle modulated simulated Kalman filter," in *Proc. of 3rd National Conference of Postgraduate Research*, Sep. 2016, pp. 875–885.

[29] A. Adam *et al.*, "Feature selection using angle modulated simulated Kalman filter for peak classification of EEG signals," *SpringerPlus*, vol. 5, pp. 1580–1603, 2016.

[30] N. Hansen,A. Ostermeier and A. Gawelczyk, "On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation," in *Proc. of the 6th International Conference on Genetic Algorithms*, 1995, pp. 57–64.

[31] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global Optimisation over continuous spaces," *Journal of Global Optimisation*, vol. 11, pp. 341–359, Dec. 1997.

[32] M. Cheng and D. Prayogo, "Symbiotic organisms search: a new metaheuristic Optimisation algorithm," *Computers & Structures*, vol. 139, pp. 98-112, Jul. 2014.

[33] N. H. Abdul Aziz, Z. Ibrahim, N. A. Ab. Aziz, and S. Razali, "Parameter-less Simulated Kalman Filter," *International Journal of Software Engineering and Computer Systems (IJSECS)*, vol. 3(February), pp. 129-137, 2017.

[34] N. H. Abdul Aziz, Z. Ibrahim, T. A. Bakare, and N. A. Ab. Aziz, "How Important the Error Covariance in Simulated Kalman Filter?," in *Proc. of The National Conference for Postgraduate Research 2016*, Sept. 2016, pp. 315-320.

[35] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problems definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical Optimisation," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, Tech. Rep.

201311 and Nanyang Technological University, Singapore, Tech. Rep., Dec. 2013.

[36] P. N. Suganthan. (2016) Shared Documents. [Online]. Available: http://http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2014/cec14-matlab-code.zip

[37] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm Optimisation," in *Proc. of the 2000 Congress on Evolutionary Computation*, Jul. 2000, pp. 84–88.

[38] S. Mirjalili. (2016) Seyedali Mirjalili homepage. [Online]. Available: http://www.alimirjalili.com/GWO.html.

[39] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*. Hoboken, NJ: John Wiley & Sons, 2004.

[40] J. Alcala-Fdez *et al.*, "KEEL: a software tool to assess evolutionary algorithms for data mining problems", *Soft Computing*, vol. 13, no. 3, pp. 307–318, May 2008.