Cubature Kalman Optimizer versus Teaching Learning Based Optimization: A Performance Comparison

Zulkifli Musa[1*], Zuwairie Ibrahim[2], Mohd Ibrahim Shapiai[3], Nor Azlina Ab. Aziz[4]

[1]*Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang Al Sultan Abdullah, 26600 Pekan, Pahang, Malaysia; E-mail: zulkifli3104@gmail.com*

[2]*Faculty of Manufacturing and Mechatronic Engineering Technology, Universiti Malaysia Pahang Al Sultan Abdullah, 26600 Pekan, Pahang, Malaysia*

[3]*Malaysia-Japan International Institute of Technology, Universiti Technologi Malaysia, 54100 Kuala Lumpur, Malaysia*

[4]*Faculty of Engineering and Technology, Multimedia University, 75450 Bukit Beruang, Melaka, Malaysia*

Abstracts: This paper compares a new Cubature Kalman Optimizer performance against the Teaching Learning Based Optimization in solving the CEC2014 test suite. The Cubature Kalman Optimizer is inspired by the estimation algorithm named Cubature Kalman filter, while the Teaching Learning Based Optimization is inspired by the teaching-learning process in a classroom. Both algorithms can be characterized as a parameter-less nature. Graphical analysis based on convergence curve shows that Cubature Kalman Optimizer has better exploration than Teaching Learning Based Optimization in the first half of the total iteration that make it able to find better solution. On the other hand, for boxplot, both algorithms show comparative based on consistency. Meanwhile, statistical analysis shows that the Cubature Kalman Optimizer algorithm is a promising approach compared to Teaching Learning Based Optimization.

Keywords: Cubature Kalman Optimizer, Teaching Learning Based Optimization, Metaheuristic, Optimization.

1. INTRODUCTION

Exact optimization methods often struggle to solve complex nonlinear and multimodal problems encountered in many real-world applications within a reasonable computational time. As a result, researchers turn to metaheuristic optimization methods to tackle such challenges. Metaheuristic algorithms are versatile techniques that can be adapted to address a wide range of optimization problems. These algorithms rely on a collection of agents to search for near-optimal solutions within a reasonable computational effort. Typically, they start by initializing the population and evaluating the fitness of the initial population. Subsequently, these algorithms iteratively generate a new population to replace the current one by defining the search direction of the agents. The way these steps are performed distinguishes one algorithm from another (Talbi, 2009).

In recent years, the versatility and effectiveness of metaheuristic algorithms in solving large-scale and diverse optimization problems have captivated researchers' attention. Metaheuristics can be classified into five categories based on their sources of inspiration: evolution algorithms, swarm intelligence algorithms, physics-inspired algorithms, human and animal lifestyle, and estimation-based algorithms.

In the evolution category, the Genetic Algorithm (GA) (Holland, 1984) simulates Darwinian evolution. It involves selection, crossover, and mutation to replace the worst solution in each generation. Solutions improve based on the best solutions obtained from particles and the swarm. Swarm intelligence encompasses the Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), which simulates fish schooling and bird flocking to determine the best course of action. Each particle represents a potential solution with its position and velocity. The best solution influences velocity updates in the entire population and the solutions of individual particles. The physics-inspired category includes the Black Hole (BH) (Hatamlou, 2013) algorithm, which emulates the phenomenon of black holes attracting stars in space. It mimics the massive gravitational power of black holes and their ability to engulf nearby objects. In the human and animal lifestyle, the Teaching Learning Based Optimization (TLBO) (Rao et al., 2012) algorithm draws inspiration from the teaching-learning process in a classroom. It simulates the influence of a knowledgeable teacher on learners. TLBO comprises two phases: the teacher phase and the learner phase. The teaching phase represents the exploitation component, where the teacher's knowledge guides the learners in improving their solutions. The learners adjust their solutions based on the teacher's solution, incorporating the best attributes to refine their solutions towards the global optimum. In the estimation- based category, popular algorithms include the Simulated Kalman Filter (SKF) (Ibrahim et al., 2016), inspired by the Kalman Filter. SKF operates similarly to the KF algorithm (Kalman, 1960), going through the prediction, measurement, and estimation steps in every iteration. Another estimation-based algorithm is the Single Agent Finite Impulse Response Optimizer (SAFIRO) (Ab Rahman et al., 2018), which draws inspiration from unbiased finite impulse response techniques (Shmaliy et al., 2016; Uribe-Murcia et al., 2021).

The Cubature Kalman Optimizer (CKO) (Musa et al., 2023) algorithm is a new proposed algorithm draws inspiration from the estimation algorithm known as the Cubature Kalman filter (CKF). As an optimizer, the CKO agent works as CKF to estimate an optimal or near-optimal solution. The overall process of the CKF algorithm is divided into six phases: initialization, fitness evaluation and update best solution, solution prediction, simulate measurement, measurement prediction, and solution update. The first two processes are similar to most metaheuristic algorithms. In contrast, the proposed optimizer has four dedicated operations: solution prediction, simulated measurement, measurement prediction, and solution update phases. The purpose of the simulated measurement phase is to simulate the actual measurement output in a real estimation process, while the solution prediction, measurement prediction, and solution update phases are adopted from the CKF. CKO only has one parameter, which makes it easy to implement. The formulation of CKO aims to strike a balance between the exploration and exploitation phase. It has produced competitive results compared to well-known algorithms, including single-agent finite impulse response optimizer (SAFIRO), single-solution simulated Kalman filter (ssSKF), simulated Kalman filter (SKF), asynchronous simulated Kalman filter (ASKF), particle swarm optimization algorithm (PSO), genetic algorithm (GA), grey wolf optimization algorithm (GWO), and black hole algorithm (BH).

The literature demonstrates that CKO consistently outperforms algorithms belonging to different classes of metaheuristics, excluding those inspired by human and animal behavior. However, to further investigate the performance of CKO compared to a human and animal behavior algorithm, we conducted a comparative analysis between CKO and TLBO. This research aims to fill the gap in understanding the relative strengths of these algorithms by evaluating their performance on 30 benchmark functions from the CEC2014. The obtained results clearly indicate that CKO exhibits significant advantages over TLBO, thereby emphasizing the importance of this research.

2. THE CUBATURE KALMAN OPTIMIZER (CKO) ALGORITHM

The cubature Kalman optimizer (CKO) is a single-agent metaheuristic algorithm based on CKF framework, where agent is employed to estimate the global minima or maxima. The process of the CKO algorithm is divided into six main phases: (1) initialization, (2) fitness evaluation and update (3) solution prediction, (4) simulate measurement, (5) measurement prediction, and (6) solution update as depicted in Figure 1**.**

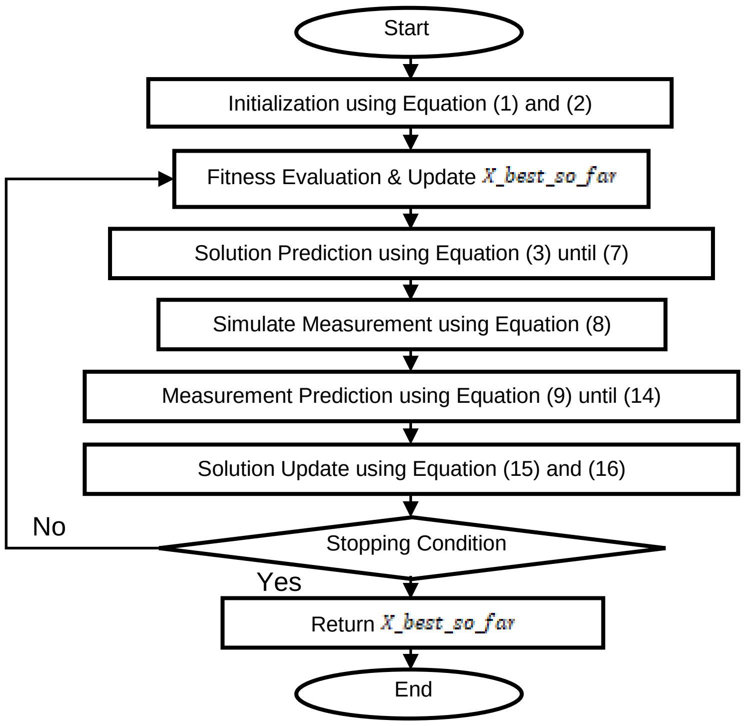
Start
Initialization using Equation (1) and (2)
Fitness Evaluation & Update $X\_best\_so\_far$
Solution Prediction using Equation (3) until (7)
Simulate Measurement using Equation (8)
Measurement Prediction using Equation (9) until (14)
Solution Update using Equation (15) and (16)
No
Stopping Condition
Yes
Return $X\_best\_so\_far$
End

Figure 1. The CKO algorithm flowchart.

The detailed process of the CKO is given as follows:

2.1. Initialization

The CKO begins with the random initialization of its agent, $x_d(0)$, within the search space as (1), where $\underline{x}_d$ is the lower limit, $\overline{x}_d$ is the upper limit of the search space in the dth dimension. Additionally, the initial value of the solution error, $P_d(0) \in [0,1]$, is generated using a random value as (2).

$$x_d(0) = randn_d + [\cup(\underline{x}_d, \overline{x}_d)] \quad \text{Equation (1)}$$

$$P_d(0) = randn_d \quad \text{Equation (2)}$$

2.2. Fitness Evaluation, Update $X\_best\_so\_far_d$

The iteration begins with the fitness calculation of the solution. The fitness of the solution $x_d(t)$ is compared to the fitness of $X\_best\_so\_far_d$ whereby $X\_best\_so\_far_d$ will be updated if the better solution ($x_d(t) < X\_best\_so\_far_d$ for minimization problems, or $x_d(t) > X\_best\_so\_far_d$ for maximization problem) is found.

2.3. Solution Prediction Phase

At first, the δ in (3) is determined, where t is the current iteration, and $tMax$ is the maximum number of iterations.

$$\delta = e^{-\beta \times \frac{t}{tMax}} \times \frac{\underline{x}_d - \overline{x}_d}{2} \quad \text{Equation (3)}$$

The following equations determine the predicted solution candidate, $xp_a(t)$, where $T1_d^j(t)$ in (4) is the generated cubature point, $U1_d^j(t)$ in (5) is the propagation of both cubature points randomly in the search space by using a random element, $rand_d \in [0,1]$.

$$T1_d^j(t) = x_d(t) + \left[\sqrt{P_d(t)} \quad -\sqrt{P_d(t)}\right] \qquad \text{Equation (4)}$$

$$U1_d^j(t) = T1_d^j(t) + rand_d(\mathrm{U}\,[-\delta\,,\delta]) \qquad \text{Equation (5)}$$

$$xp_d(t) = \frac{1}{2}\sum_{j=1}^{2} U1_d^j(t) \qquad \text{Equation (6)}$$

Two cubature point involved in these steps is indicated by $j = 1$ (first point) and $j = 2$ (second point). Once the predicted solution candidate is calculated, the solution error, $Pp_d(t)$, must be predicted using (7), where $rand_d \in [0,1]$ is used to present the system error.

$$Pp_d(t) = \left(\left(U1_d^1(t) - xp_d(t)\right) \times \left(U1_d^2(t) - xp_d(t)\right)\right) + randn_d \qquad \text{Equation (7)}$$

2.4. Simulate Measurement Phase

The measurement step performs the role of feedback of estimation process. The measurement of each solution is simulated based on the following (8):

$$z_d(t) = (xp_d(t) + sin(rand_d \times 2\pi) \times |xp_d(t) - X\_best\_so\_far_d(t)|) \qquad \text{Equation (8)}$$

where the simulated measurement value for the agent, $z_d(t)$ may take any random position in a locus $|xp_d(t) - X\_best\_so\_far_d(t)|$. A random element, $rand_d \in [0,1]$ in $sin(rand_d \times 2\pi)$ isresponsible for the stochastic aspect of the CKO algorithm.

2.5. Measurement Prediction Phase

The predicted measurement vector, $zp_d(t)$ is determined by the following equations:

$$T2_d^j(t) = xp_d(t) + \left[\sqrt{Pp_d(t)} \quad -\sqrt{Pp_d(t)}\right] \qquad \text{Equation (9)}$$

$$U2_d^j(t) = T2_d^j(t) + rand_d(\mathrm{U}\,[-\delta\,,\delta]) \qquad \text{Equation (10)}$$

$$zp_d(t) = \frac{1}{2}\sum_{j=1}^{2} U2_d^j(t) \qquad \text{Equation (11)}$$

where $T2_d^j(t)$ in (9) are the generated cubature point and $U2_d^j(t)$ in (10) is the propagation cubature point randomly in the search space by using a random element, $rand_d \in [0,1]$. Then, the measurement error, $Pzz_d(t)$ and the cross-error, $Pxz_d(t)$ need to be estimated for gain calculation. These two types of error can be obtained by using (12) and (13), respectively:

$$Pzz_d(t) = \left(\left(U2_d^1(t) - zp_d(t)\right) \times \left(U2_d^2(t) - zp_d(t)\right)\right) + randn_a \qquad \text{Equation (12)}$$

$$Pxz_d(t) = \left(\left(U1_d^1(t) - xp_d(t)\right) \times \left(U2_d^2(t) - zp_d(t)\right)\right) \qquad \text{Equation (13)}$$

where $rand_d \in [0,1]$ is measurement noise. Then, the gain, $W_d(t)$ can be calculated using (14):

$$W_d(t) = Pxz_d(t)/Pzz_d(t) \qquad \text{Equation (14)}$$

2.6. Solution Update Phase

Finally, the solution, $x_d(t+1)$ and solution error, $P_d(t+1)$ are updated by the following equations:

$$x_d(t+1) = xp_d(t) + W_d(t)\left(z_d(t) - zp_d(t)\right) \qquad \text{Equation (15)}$$

$$P_d(t+1) = Pp_d(t) - W_d(t)\,Pzz_d(t) \qquad \text{Equation (16)}$$

This process is iteratively updated until the stopping condition is fulfilled. The pseudo-code of CKO is given in Figure 2.

Pseudocode: CKO Algorithm

```
01:  Initialize solution distribution: $x_d(0)$ using Equation (1)
02:  Initialize solution error: $P_d(0)$ using Equation (2)
03:  Initial the best solution: $X\_best\_so\_far_d = infinity$
     for $t = 1 : tMax$
04:      Fitness calculation: $fitness = calculate\ fitness\ of\ x_d(0)$
         if $x_d(t) < X\_best\_so\_far_d$
05:          $X\_best\_so\_far_d = x_d(t)$
         end
06:      Update radius: $\delta$ using (3)
07:      Generate cubature points: $T1_d^j(t)$ using Equation (4)
08:      Propagate cubature points: $U1_d^j(t)$ using Equation (5)
09:      Generate predicted solution: $xp_d(t)$ using Equation (6)
10:      Generate predicted error: $Pp_d(t)$ using Equation (7)
11:      Simulate measurement: $z_d(t)$ using Equation (8)
12:      Generate cubature points: $T2_d^j(t)$ using Equation (9)
13:      Propagate cubature points: $U2_d^j(t)$ using Equation (10)
14:      Generate predicted measurement: $zp_d(t)$ using Equation (11)
15:      Calculate measurement error: $Pzz_d(t)$ using Equation (12)
16:      Calculate cross error: $Pxz_d(t)$ using Equation (13)
17:      Calculate Gain: $W_d(t)$ using Equation (14)
18:      Calculate estimated solution: $x_d(t+1)$ using Equation (15)
19:      Calculate estimated error: $P_d(t+1)$ using Equation (16)
     end
20:  return
```

Figure 2. The CKO algorithm pseudocode.

3. Teaching Learning Based Optimization

The Teaching Learning Based Optimization (TLBO) algorithm is formulated based on the principles of teaching and learning processes observed in a classroom environment. TLBO serves as a population-based optimization algorithm, where learners represent potential solutions to the optimization problem. The formulation of TLBO involves the following steps: initialization, teaching phase and learning phase.

In the initialization phase, the population of learners, X_i distributed within the search space randomly as in (17), where $\underline{x}_d$ is the lower limit, $\overline{x}_d$ is the upper limit of the search space.

$$X_i = unifrnd(\underline{x}, \overline{x}) \quad \text{Equation (17)}$$

3.1. Teaching Phase

The algorithm TLBO begins by calculate the mean of all 'learners' solutions, M_i and identifying the best solution as the 'teacher', T_i for that iteration, i. Then, T_i will try to move mean M_i towards its own level, so now the new mean will be T_i designated as M_{new}. The difference between the existing, $Difference\_Mean_i$ and the new mean, M_{new} given in (18):

$$Difference\_Mean_i = r_i(M_{new} - T_F M_i) \quad \text{Equation (18)}$$

where r_i is a random number in the range [0,1], and T_F is a teaching factor that decides the value of the mean to be changed. The value of T_F can be either 1 or 2 which is again a heuristic step and decided randomly with equal probability as (19).

$$T_F = round[1 + rand(0,1)\{2-1\}] \quad \text{Equation (19)}$$

This heuristic step helps regulate the modification of the mean and introduces variability into the algorithm. After that, update the teacher's solution using (20), where $X_{new,i}$ is known as new solution and $X_{old,i}$ is current solution.

$$X_{new,i} = X_{old,i} + Difference\_Mean_i \quad \text{Equation (20)}$$

3.2. Learning Phase

In the learning phase, each learner increases their knowledge by two different mean: one through input from the teacher in (21) and other through interaction between themselves in (22). A learner interacts randomly with other learners with the help of group discussions, presentations, formal communications, etc. A learner learns something new if the other learner has more knowledge than him or her.

$$X_{new,i} = X_{old,i} + r_i(X_i - X_j) \quad \text{Equation (21)}$$
$$X_{new,i} = X_{old,i} + r_i(X_j - X_i) \quad \text{Equation (22)}$$

Finally, evaluate the objective function for the new solution, $X_{new,i}$. If the new solution is better than the current solution, $X_{old,i}$ re-place the current solution with the new solution. The process repeated the teaching and learning phases until a predefined termination criterion is met.

4. Experiment Setup

The performance of the proposed CKO is compared with the TLBO using CEC2014 Benchmark Test Suite for single-objective optimization. The CEC2014 test suite comprises of 30 functions consisting of mixture of: three unimodal test suite (Fn1, Fn2 and Fn3) to investigate the exploration capability, 13 simple multimodal test suite (Fn4 until Fn16) to investigate the exploitation capability, six hybrid test suite (Fn17 until Fn22) to investigate the capability of solving complex problem, and eight composition test suites (Fn23 until Fn30) to investigate the balance between exploration and exploitation capability. The test functions are tabulated in Table 1.

Table 1. The CEC2014 Benchmark Test Suite

Type	No	Function	Ideal Fitness
Unimodal function	1	Rotated High Conditioned Elliptic function	100
	2	Rotated Bent Cigar function	200
	3	Rotated Discus function	300
Simple multimodal function	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 function	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 function	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 comparison between CKO and TLBO is conducted over 50 trials, each with 1 million function evaluations and complexity set to 50-dimension sizes. For algorithm parameter setting, the coefficient value of CKO algorithm is set to 12, while the random value in TLBO is set between 1 and 2. The MATLAB code for the CEC 2014 benchmark suite can be accessed on the website http://github.com/P-N-Suganthan/CEC2014. In this experiment, the Wilcoxon signed rank test is conducted to provide an unbiased observation, based on their performance differences at a 5% significance level.

5. RESULT AND DISCUSSION

This section presents the results of the CKO algorithm, compared against TLBO algorithms using the CEC 2014's benchmark suite. Tables 2 present the mean of the fitness achieved by both algorithms for unimodal, simple multimodal, hybrid and composition benchmark functions. The numbers written in **bold** indicate the best mean value obtained for the corresponding objective function among both algorithms. The CKO is able to find better performance for 22 functions from the 30 test functions.

Table 2. Average Fitness Value

Function	Average fitness CKO	Average Fitness TLBO
Fn1	1.79E+06	**6.55E+05**
Fn2	**5855.3**	5908.3
Fn3	**301.67**	345.58
Fn4	503.08	**492.85**
Fn5	**520**	521.11
Fn6	**618.91**	637.87
Fn7	**700.01**	700.12
Fn8	**803.46**	981.69
Fn9	**1051.4**	1091
Fn10	**1232.6**	5232.2
Fn11	**5648.3**	10753
Fn12	**1200**	1203.2
Fn13	**1300.4**	1300.6
Fn14	**1400.1**	1400.3
Fn15	**1511**	1575.7
Fn16	**1618.8**	1620.3
Fn17	**1.53E+05**	1.73E+05
Fn18	**3374.6**	3586.5
Fn19	1936.9	**1922**
Fn20	**2241.9**	2321
Fn21	1.14E+05	**98795**
Fn22	**2974.7**	3007.2
Fn23	2644	**2644**
Fn24	2662.2	**2600**
Fn25	2713.7	**2700**
Fn26	**2700.5**	2759
Fn27	**3536.4**	4117.2
Fn28	**4711.3**	5279.2
Fn29	**10468**	4.64E+07
Fn30	18347	**16079**

5.1. Unimodal function

The results in table 2 (Fn1, Fn2 and Fn3) show the CKO algorithm better than TLBO algorithmin solving unimodal function, especially for solving Fn2 and Fn3. The CKO algorithm managed to converge near the ideal fitness value. The comparison of boxplots and convergence curves in solving unimodal function are shown in Figure 3 and 4 respectively. In the boxplot, CKO still performs competitively compared with TLBO algorithms, with a small deviation from the median value. We can also see from the convergence curve of the CKO algorithm that once the agent enters the second half of the iterations, exploitation kicks in to refine the estimation.

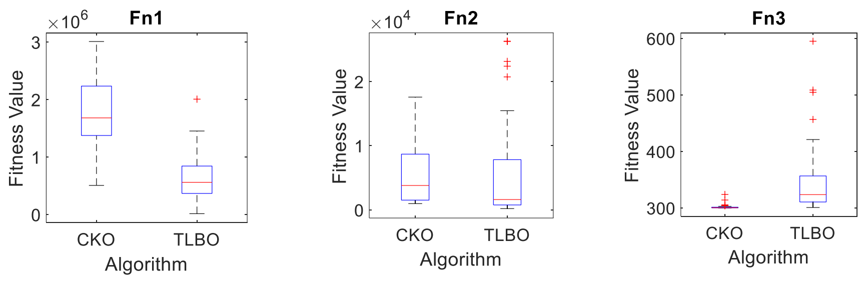

Figure 3. Comparison of boxplot for unimodal functions

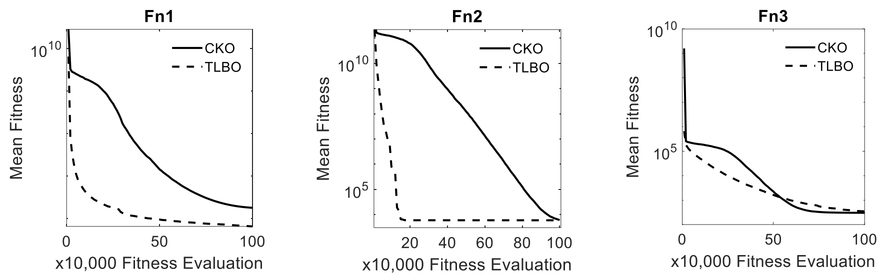

Figure 4. Comparison of convergence curve for unimodal functions

5.2. Simple multimodal function

The results in Table 2 (Fn4 until Fn16) show that although CKO managed to outperform TLBO by 12 out of 13 simple multi modal functions. The CKO algorithm managed to solve most of the problems closer to optimality especially for Fn5, Fn6, Fn7, Fn8, Fn12, Fn13, Fn14, Fn15, and Fn16. Based on the convergence behavior of the algorithms, we can see that CKO algorithm is able to do exploitation better than the TLBO as shows in Figure 6. Meanwhile, in Figure 5, we can see that CKO has a very good and consistent performance depicted by the position of the boxplot and its size.

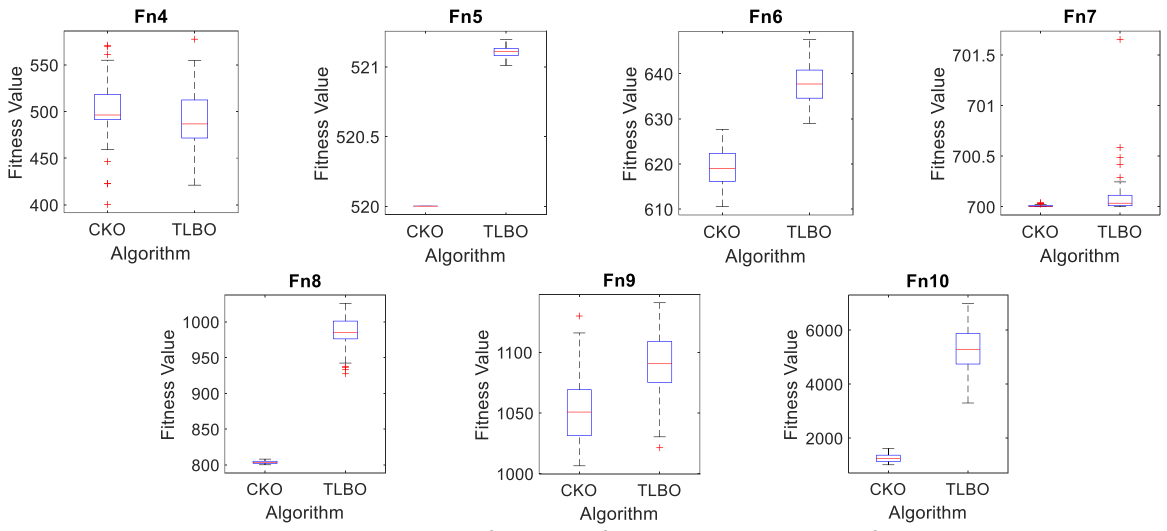

Figure 5. Comparison of boxplot for simple multimodal functions.

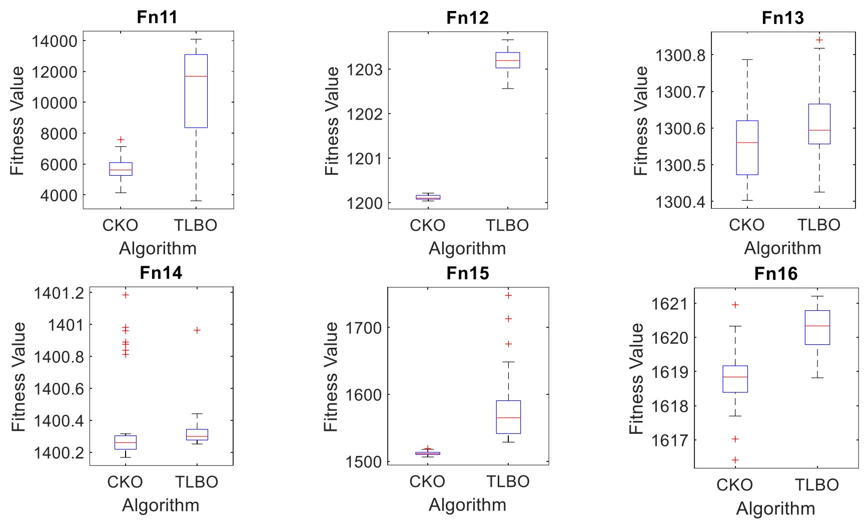

Figure 5. Comparison of boxplot for simple multimodal functions (continues)

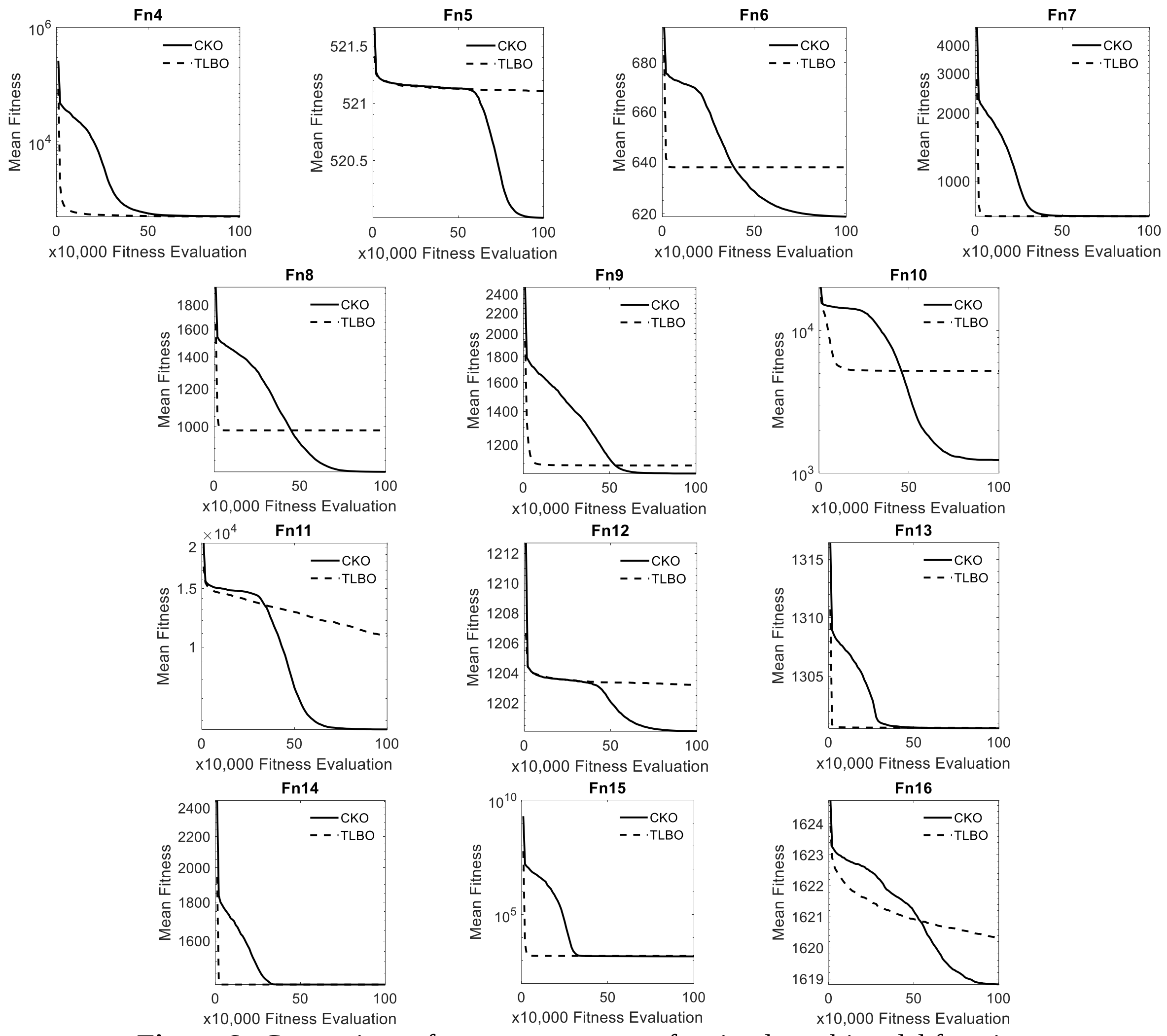

Figure 6. Comparison of convergence curve for simple multimodal functions

5.3. Hybrid function

The CKO algorithm managed to outperform TLBO the most in solving hybrid benchmark problems. Hybrid functions mimic real-world optimization 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 have different properties. From table 2 (Fn17 until Fn22), it can be seen that CKO algorithm performs the best to TLBO in 4 out of 6 hybrid function. The convergence curve in Figure 8 shows, CKO has better exploration phase. Meanwhile, the boxplot comparison shown in Figure 7 demonstrates that CKO has a very consistent performance throughout the 50 runs.

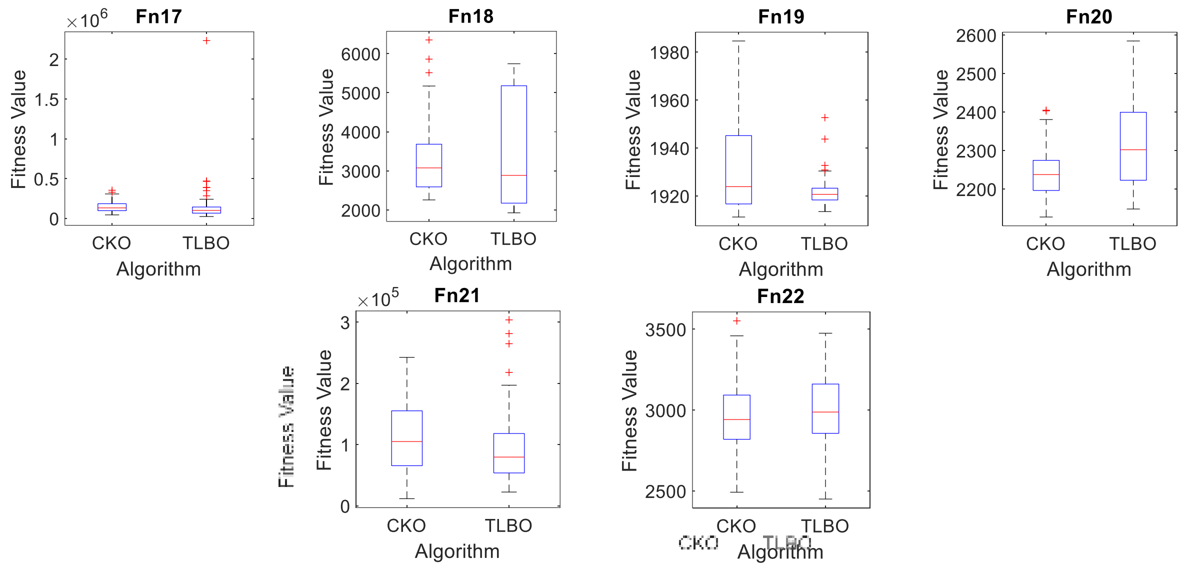

Figure 7. Comparison of boxplot for hybrid functions

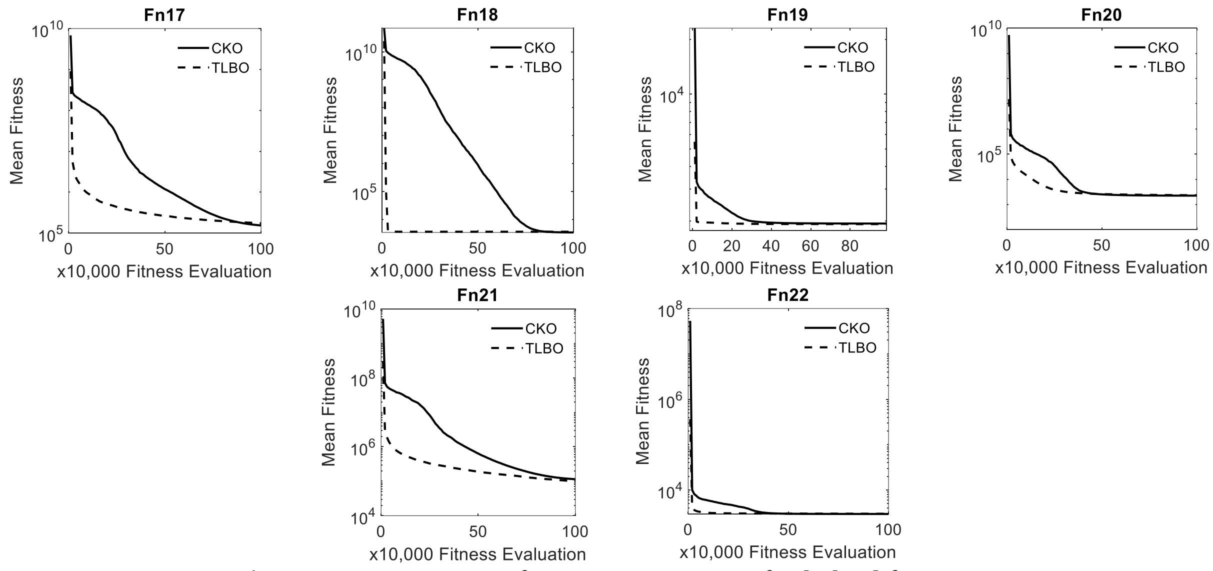

Figure 8. Comparison of convergence curve for hybrid functions

5.4. Composition function

Composition functions are used to test the algorithms' tendency to converge to the center of the search space. A local optimum is set to the origin as a trap for each composition function. Results in table 2(Fn23 until Fn30) show that the CKO algorithm has the same performance compared to TLBO where both win 4 out of 8 composition benchmark functions. Figure 10 gives a graphical view of the convergence behavior of the CKO algorithm in comparison to the TLBO algorithms when solving benchmark function. It shows that, CKO algorithm has better exploitation phase especially in the first half of the total iterations. This is followed by figure 9, which shows CKO algorithm has a competitive consistency with the TLBO, depicted by a narrower boxplot.

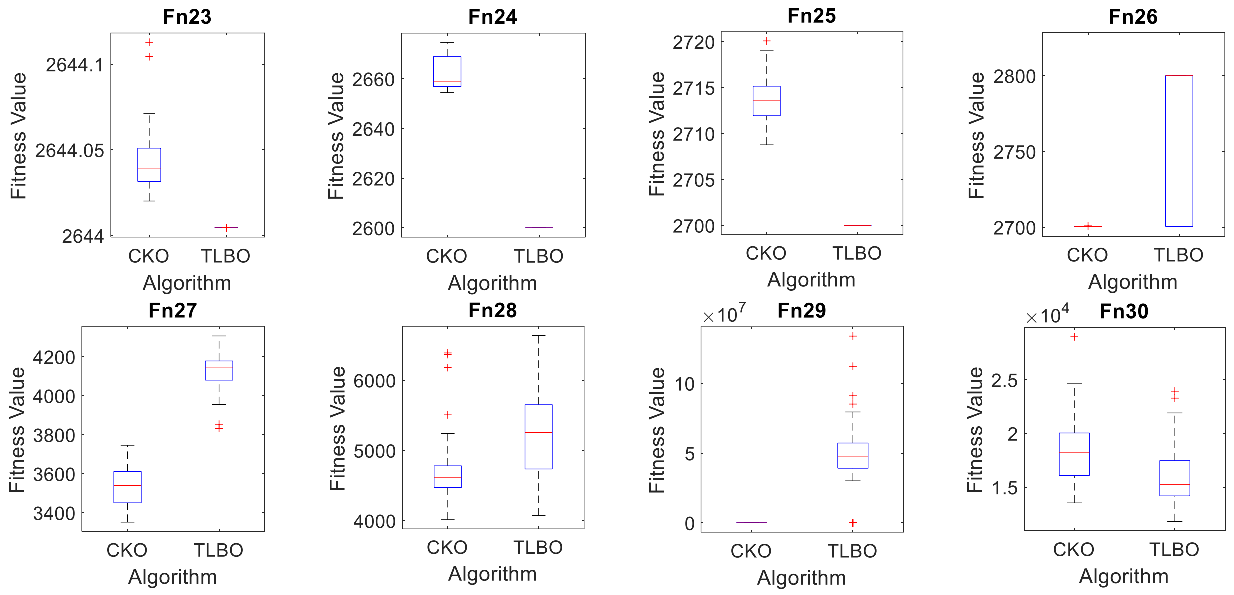

Figure 9. Comparison of boxplot for composition functions

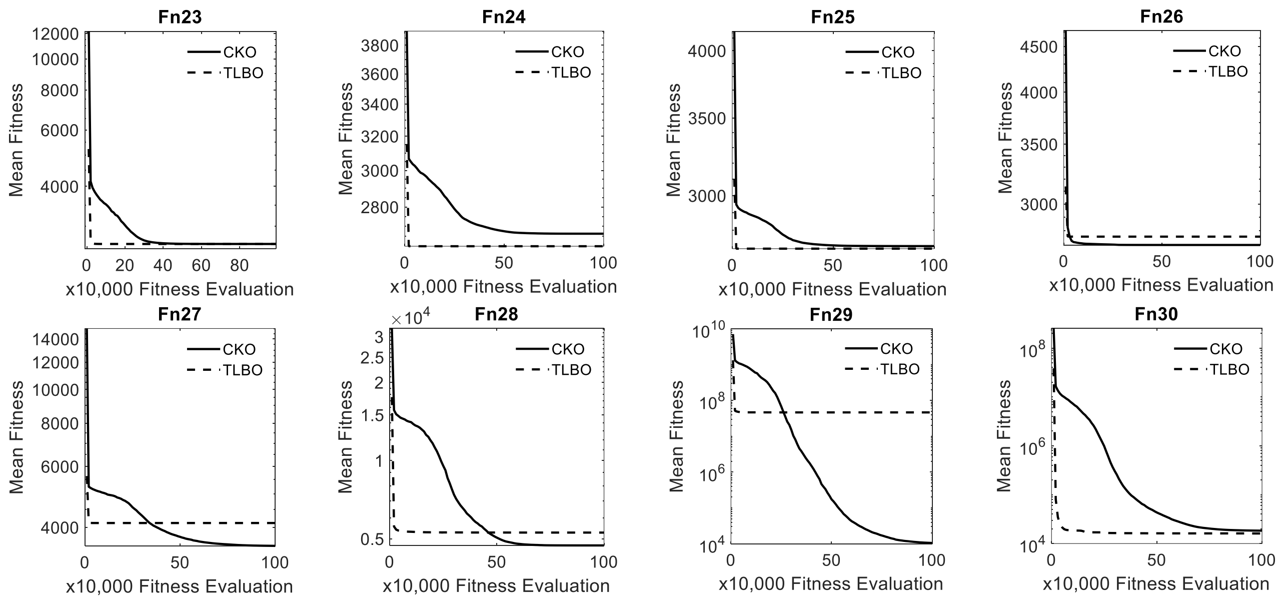

Figure 10. Comparison of convergence curve for composition functions

5.5. Statistical analysis

The Wilcoxon signed rank test with significance level, α = 0.05 is chosen to provide an unbiased observation. The test gives a statistical value of 125 which is smaller than 137, thus the null hypothesis of equivalent performance is rejected and CKO is concluded to be significantly better than the TLBO.

Overall, CKO proved to have a good balance between the exploration and exploitation phase due to the CKF update strategy associated with the shrinking local neighborhood. The CKO is a practical algorithm for solving simple multimodal functions and demonstrates strong competitiveness in solving unimodal, hybrid, and composition functions. CKO successfully led 22 out of 30 functions in CEC2014 test suite. Furthermore, CKO uses only one parameter compared to different estimation-based algorithms, making it easy to tune.

CONCLUSION

The CKO and TLBO are metaheuristic algorithm inspired by Cubature Kalman filter and teaching- learning process in classroom, respectively. Both algorithms have been incorporated to balance exploration and exploitation. For that reason, both algorithms have been tested with 30 benchmarking functions in CEC2014. The accuracy performance of the algorithms has been analyzed and discussed. Graphical plots have been included to portray the convergence trend and the accuracy achievement. The test results show that the CKO graphs have outperformed the TLBO graphs in both convergence and accuracy (box plot) for most functions. In the future, the algorithm will be used to solve and optimize a robotic system's modelling and control problems. These compelling findings not only highlight CKO's effectiveness in optimizing complex problems but also provide robust evidence of its clear superiority over TLBO. In the future, the algorithm will be used to solve and optimize a robotic system's modelling and control problems.

REFERENCES

[1] Ab Rahman, T., Ibrahim, Z., Ab Aziz, N. A., Zhao, S., & Abdul Aziz, N. H. (2018). Single-Agent Finite Impulse Response Optimizer for Numerical Optimization Problems. IEEE Access, 6, 9358–9374. https://doi.org/10.1109/ACCESS.2017.2777894

[2] Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. Information Sciences, 222. https://doi.org/10.1016/j.ins.2012.08.023

[3] Holland, J. H. (1984). Genetic Algorithms and Adaptation. In Adaptive Control of Ill-Defined Systems (Vol. 16, pp. 317–333). Springer US. https://doi.org/10.1007/978-1-4684-8941-5_21

[4] Ibrahim, Z., Aziz, N. H. A., Nor, N. A., Razali, S., & Mohamad, M. S. (2016). Simulated Kalman Filter: A novel estimation-based metaheuristic optimization algorithm. Advanced Science Letters, 22(10), 2941–2946. https://doi.org/10.1166/asl.2016.7083

[5] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Journal of Fluids Engineering, Transactions of the ASME, 82(1). https://doi.org/10.1115/1.3662552

[6] Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimisation. Proc. of IEEE Int. Conf. on Neural Network, 1972–1978. https://doi.org/10.1007/978-3-030-61111-8_2

[7] Musa, Z., Ibrahim, Z., Shapiai, M. I., & Tsuboi, Y. (2023). Cubature Kalman Optimizer: A Novel Metaheuristic Algorithm for Solving Numerical Optimization Problems. Journal of Advanced Research in Applied Sciences and Engineering Technology, (accepted).

[8] Rao, R. V, Savsani, V. J., & Vakharia, D. P. (2012). Teaching-Learning-Based Optimization: An optimization method for continuous non-linear large scale problems. Information Sciences, 183(1). https://doi.org/10.1016/j.ins.2011.08.006

[9] Shmaliy, Y. S., Khan, S., & Zhao, S. (2016). Ultimate iterative UFIR filtering algorithm. Measurement: Journal of the International Measurement Confederation, 92. https://doi.org/10.1016/j.measurement.2016.06.029

[10] Talbi, E.-G. (2009). Metaheuristics. In Metaheuristics: From Design to Implementation. John Wiley & Sons, Inc. https://doi.org/10.1002/9780470496916

[11] Uribe-Murcia, K., Andrade-Lucio, J. A., Shmaliy, Y. S., & Xu, Y. (2021). Unbiased FIR filtering under bernoulli-distributed binary randomly delayed and missing data. European Signal Processing Conference, 2021-Janua, 2408–2412. https://doi.org/10.23919/Eusipco47968.2020.9287509

DOI: https://doi.org/10.15379/ijmst.v10i3.1847