Telecommun Syst (2017) 65:351–364
DOI 10.1007/s11235-016-0237-4

Monkey King Evolution: an enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment

Jeng-Shyang Pan[1] · **Zhenyu Meng**[2] · **Shu-Chuan Chu**[3] · **Hua-Rong Xu**[4]

Published online: 28 September 2016

Abstract Optimization algorithms are proposed to maximize the desirable properties while simultaneously minimizing the undesirable characteristics. Particle Swarm Optimization (PSO) is a famous optimization algorithm, and it has undergone many variants since its inception in 1995. Though different topologies and relations among particles are used in some state-of-the-art PSO variants, the overall performance on high dimensional multimodal optimization problem is still not very good. In this paper, we present a new memetic optimization algorithm, named Monkey King Evolutionary (MKE) algorithm, and give a comparative view of the PSO variants, including the canonical PSO, Inertia Weighted PSO, Constriction Coefficients PSO, Fully-Informed Particle Sawrm, Cooperative PSO, Comprehensive Learning PSO and some variants proposed in recent years, such as Dynamic Neighborhood Learning PSO, Social Learning Particle Swarm Optimization etc. The proposed MKE algorithm is a further work of ebb-tide-fish algorithm and what's more it performs very well not only on unimodal benchmark functions but also on multimodal ones on high dimensions. Comparison results under CEC2013 test suite for real parameter optimization show that the proposed MKE algorithm outperforms state-of-the-art PSO variants significantly. An application of the vehicle navigation optimization is also discussed in the paper, and the conducted experiment shows that the proposed approach to path navigation optimization saves travel time of real-time traffic navigation in a micro-scope traffic networks.

Keywords Benchmark function · Monkey King Evolutionary algorithm · Number of function evaluation · Particle Swarm Optimization · Vehicle navigation · Wireless sensor network

✉ Zhenyu Meng
mzy1314@gmail.com

Jeng-Shyang Pan
jengshyangpan@gmail.com

[1] Fuzhou University of International Studies and Trade, Fuzhou, China

[2] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

[3] School of Computer Science, Engineering and Mathematics, Flinders University, Adelaide, Australia

[4] Department of Computer Science and Technology, Xiamen University of Technology, Xiamen, China

1 Introduction

With increasing optimization demands in our lives, more and more powerful optimization algorithms are proposed to tackle these problems in different areas. Computational Intelligence gives computational methodologies and approaches to address and tackle these complex real-world optimization problems especially for those with noise and uncertainty. The standard approach to tackle these complex problems often begins by designing an objective function that can model the problems' objectives while incorporating any constraints [1]. Optimization algorithm targeting a specific problem may be of less use for different categories of optimization tasks. The idea driving the development of a new optimization algorithm, is that the heuristic do not necessarily have to be completely problem-dependent, but that general optimization techniques could be developed that were applicable to a broad class of different problems [2]. Therefore, the aim of the paper is to propose a powerful optimization algorithm for a broad class of different problems rather than a problem-specific approach for a concrete use.

Particle Swarm Optimization (PSO) was a powerful evolutionary computational algorithm introduced in 1995 [3]. Since the inception of PSO algorithm, many researchers had expanded on the original idea of the canonical PSO. Representative changes of the PSO evolution equation varied from the alterations of parameter adjustments to alterations of particle topologies. As the original PSO algorithm was inspired by the behavior of the bird flocks, particle neighborhoods were firstly used for the simulation. The calculation of Euclidian neighborhood was time consuming, and the performance of the optimization results that utilized Euclidian neighborhood were not good enough as well. All these resulted in the abandon of Euclidian neighborhood model. Neighborhood topology unrelated to particle locations came to be useful in particle evolution. Particles used their historical best (lbest) and topology best (gbest) for evolution, and they did not use crossover operations which used in Genetic Algorithm. The moving velocity was used to make a balance between the exploitation and exploration of a particle.

Parameters adjustments showed significant improvement of the canonical PSO, [4] proposed a new optimizer using particle swarm theory, and examined how the changes in the paradigm affected the number of iteration required to meet an error criterion. Inertia weight and constriction coefficient of velocity were consequently proposed and analyzed for the importance of convergence [5–7]. Particle neighborhood topologies also played important roles in many complex especially multimodal optimization problems. Mendes et al. [8] analyzed the effect of different neighborhood topologies and consequently the authors proposed a Fully-Informed Particle Swarm (FIPS) variant giving some hints that simpler maybe better. Other variants solved some shifted or rotated benchmark functions by using dynamic multi-swarm or dynamic topology [9,10]. Bratton and Kennedy [11] defined a standard PSO as a baseline for performance testing of improvements among newly proposed PSO variants. Topology, bound constraints, particle initialization and population size were all analyzed in the paper. Some recently proposed PSO variants such as Dynamic Neighborhood Learning based PSO (DNLPSO) [12] and Social Learning PSO (SLPSO) [13] used dynamic topology to void premature of convergence aiming at better performance on multimodal functions.

With the development of optimization research, many new optimization algorithms have also been proposed in the recent few decades such as Cat Swarm Optimization [14], Bat Algorithm [15], and Ebb-tide-fish algorithm [16,17], etc. Some of the newly proposed algorithm make contrasts with the canonical PSO algorithm, and they may be better than this canonical algorithm, but these comparisons usually make little sense as that much improvement have been done on the classical algorithms. That why the standard PSO form was proposed in [11] as a baseline for performance comparison. So the paper gives a comparative view of some state-of-the-art PSO variants with the new proposed algorithm, contribution and tradeoff are also analyzed herein the paper.

Optimization of vehicle navigation in a city is an urgent problem nowadays with the increasing vehicles on the road. The optimization often aims at a better satisfaction of individual drivers as well as a better throughput of the whole city traffic networks. Vehicle localization and traffic condition acquisitions play important roles in such optimization of a certain routing scheme, and GPS based localization approach is usually a commonly choice for many applications. As there are still some week points of GPS based approach, some other localization techniques are proposed for robustness and critical use, such as dead reckoning, cellular localization, image/video localization in vehicular ad-hoc network to overcome GPS limitations [18]. Many applications of wireless sensor networks use the sensor node for localization, these different approaches for localization and navigation can be seen in these papers [19–21]. As wireless sensor network is a convenient approach to grasp the traffic conditions of a certain areas [22], the grasped traffic condition can be used as a key element in vehicle navigation, and this is also used in the model proposed in this paper.

The rest of the paper is organized as follows. Section 2 shows some earlier work of swarm based algorithms. Section 3 discusses the details of the newly proposed Monkey King Evolution algorithm. Section 4 shows the benchmark functions and verification of MKE algorithm. Section 5 shows the application of MKE algorithm to tackle microscope vehicle navigation problem in wireless sensor network environment. Section 6 gives the final conclusion.

2 Related works

PSO is simple and powerful algorithm for optimization problems, many researchers have learned this technique and proposed many variants to improve the performance for a large domain of optimization tasks or for a certain specific problem. Equation 1 presents a Inertia Weighted Particle Swarm Optimization(IWPSO) variant, a smaller inertia weight value often results in a fast convergence when the variant can find the global optima.

$$\begin{cases} v_i^{t+1} \leftarrow \omega * v_i^t + c_1 * (X_{fb} - x_i) + c_2 * (X_{gb} - x_i), \\ X_i^{t+1} \leftarrow X_i^t + v_i^{t+1}. \end{cases} \tag{1}$$

A smaller value often means that the variant pays more attention on exploitation while a larger one pays more attention on exploration. Equation 2 is called a constriction coefficient

variant, and the inertia weighted variant can be considered as a special form of the constriction coefficient variant. The pseudo code of Constriction Coefficient Particle Swarm Optimization (CCPSO) variant is shown in Algorithm 1.

$$\begin{cases} v_i^{t+1} \leftarrow \chi * (v_i^t + c_1 * (X_{fb} - x_i) + c_2 * (X_{gb} - x_i)), \\ X_i^{t+1} \leftarrow X_i^t + v_i^{t+1}. \end{cases} \tag{2}$$

Algorithm 1 Pseudo code of the PSO Algorithm

Require:
Initialize the searching space V, Dimension D, and the benchmark function $f(X)$.
Ensure:
while $exeTime < MaxIteration | !stopCriterion$ **do**
  if $exeTime = 1$ **then**
    Generate the population P, Coordinates X, Velocity V, X_{lbest}.
  end if
  if $exeTime > 1$ **then**
    for $pSize = 1 : PopSize$ **do**
      $V_{pSize} = \chi * (V_{pSize} + c_1 * (X_{lbest} - X_{pSize}) + c_2 * (X_{gbest} - X_{pSize}))$
      $X_{pSize} = X_{pSize} + V_{pSize}$
    end for
  end if
  if $f(X)$ optimal than $f(X_{lbest})$ **then**
    $X_{lbest} \leftarrow X$
  end if
  $X_{gbest} = max\{X_{lbest}\}$.
end while
Output:
The global optima X_{gbest}, $f(X_{gbest})$.

FIPS uses specific neighborhood topology, such as ring topology, four cluster topology, Von Neumann topology et al., rather than the population global best topology for evolution. Equation 3 shows the evolution equation of FIPS. FIPS seems to find optima in few iterations but it depends much more on particle topology. Other new proposed PSO variants use different topologies for evolution, Comprehensive Learning Particle Swarm Optimization (CLPSO) uses randomly generated neighborhood relation to supervise the learning process, and particles learn from worse solutions of the neighborhood under certain learning probability to void premature. SLPSO also uses a dynamic neighborhood and particles learn from any better particles in the sorted neighborhood.

$$\begin{cases} v(t+1) = \chi * \left(v(t) + \dfrac{1}{K_i} \displaystyle\sum_{n=1}^{K_i} c * (Xnb_n - x(t)) \right), \\ x(t+1) = x(t) + v(t+1). \end{cases} \tag{3}$$

3 The Monkey King Evolutionary algorithm

3.1 Overview of ebb-tide-fish algorithm

Ebb-tide-fish algorithm is a newly proposed meta-heuristic algorithm mimicking foraging behavior of fish [16,17]. The algorithm examines the different velocity initialization approaches of PSO algorithm, and shows a velocity-free evolution scheme. Particles in ebb-tide-fish algorithm are divided into two categories, and particles in the first category search the local area around their locations while particles in the second category search towards the global best particle in the population. The category labels of all particles change dynamically as to the change of the search behaviors. The ebb-tide-fish algorithm performs very well in lower dimension optimization problems. Equation 4 shows the evolution equation of the ebb-tide-fish algorithm.

$$\begin{aligned} X_{etf,G}^i &= \{x_1, x_2, \ldots, x_j, \ldots, x_D\}, \\ x_j &= x_j \pm r * rand() * x_j, \ j \in D, \\ X_{i,G+1} &= X_{i,pbest} + F * rand() * (X_{gbest} - X_{i,G}). \end{aligned} \tag{4}$$

The pseudo code of the ebb-tide-fish algorithm is shown in Algorithm 2. Deeper analysis of ebb-tide-fish on high dimensional multimodal benchmark function shows that the algorithm may be premature for some complex optimization problems. Improvements should be done to change the local search behavior to void premature when tackling these tough problems, and this the reason why we propose an enhanced version (Monkey King Evolution algorithm) of it.

3.2 Details of Monkey King Evolution algorithm

"Journey to the West" is one of the four great classical novels of Chinese literature, the novel related to the amazing adventures of the priest Sanzang as travels west in search of Buddhist Sutras with his three disciples. Monkey King is the most powerful disciple of the three and he can transforms into several different small monkeys to acquire the circumstances around current location. All these small monkeys give feedbacks to Monkey King after their own exploration, and then Monkey King can grasp where is the optima among these locations. Exploration behaviors of these small monkeys are implemented by Eq. 5. $x_{i,G+1}$ denotes the ith particle in $G+1$th generation, $x_{i,pbest}$ denotes the ith particle's historical best, x_{r1} is the first randomly selected particle while x_{r2} denotes the second randomly selected particle from the population. x_{r1} and x_{r2} in the equation can be the same, and this is also similar but different in comparison to DE [1].

$$x_{i,G+1} = x_{i,pbest} + FC * (x_{r1} - x_{r2}). \tag{5}$$

Algorithm 2 Pseudo code of the ebb tide fish algorithm

Require:
Initialize the searching space $V(v_1, v_2, \ldots, v_d)$ and the benchmark function $f(X)$(X denotes the coordinate of a virtual fish).
Ensure:
while $exeTime < MaxIteration$ **do**
  if $exeTime = 1$ **then**
    Generate the fish population T_j $(j = 1, 2, \ldots, n)$ with coordinate $X_j = (x_1, x_2, \ldots, x_d)^T$ and generate single search fish and change its $Flag_j$.
  end if
  if $exeTime > 1$ **then**
    for $pSize = 1 : PopSize$ **do**
      if $Flag_i = true$ & $A_i < A_{thres}$ **then**
        for $C_{swim} = 1 : N$ **do**
          $x_i = x_i \pm r$
          Calculate the benchmark value and record local best X^t_{lbest}.
        end for
        $X_{pos_i} \leftarrow X^t_{lbest}$
        $Flag_i = false$
      else
        $X_{pos_i} \leftarrow X^t_i + (X^t_{gbest} - X^t_i) * rand();$
      end if
      Generate single search fish and change its $Flag$
    end for
  end if
  Calculate the benchmark value of all the fish.
  if $f(X_{pos_i})$ is optima rather than $f(X^t_i)$ **then**
    $X^{t+1}_i \leftarrow X_{pos_i}$
  end if
  Record the optima fish with coordinate X^t_{gbest}.
end while
Output:
The global optima X_{gbest}.

Figure 1 shows the exploitation behavior of Monkey King, and this is implemented by the exploration of small monkeys. The origin point the coordinate in Fig. 1 denotes Monkey King's location. The arrows in the figure show the moving direction and step size of the small monkeys. The small monkeys move from the origin point of the coordinate to the arrows pointed points, and then Monkey King can grasp where the optimal location is among all the locations of small monkeys after their exploration. There are several schemes to generate Monkey King particles in the population. The first scheme is that Monkey King particles are randomly generated with the number related to a certain rate of the population, then the left particles in the population are labeled as common particle. In the second scheme, the global best particle is denoted as monkey king particle as Monkey King in the mythological novel "Journey to the West" is the most powerful disciple. The last scheme is that all particles are equal and all of them are labeled as monkey king particles. They evolve according to the following equation written in a matrix style Eq. 6. $\widehat{X}$ denotes the matrix of all particles in population, and $\widehat{X}_{gbest}$ denotes the global best matrix. Both of the two matrices have ps components, ps means the population size. The ith component of $\widehat{X}$ is the coordinate of the ith particle.

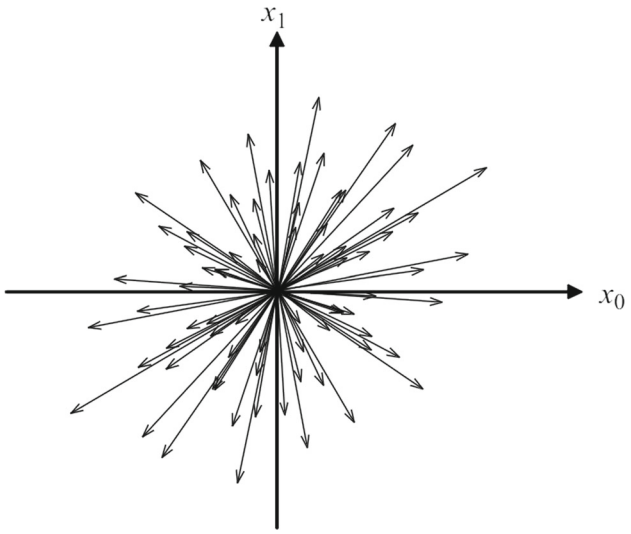


Fig. 1 The illustration of Monkey King particle's exploitation vector

$$\begin{cases} \widehat{X}_{diff} = (\widehat{X}_{r1} - \widehat{X}_{r2}) \\ \widehat{X}_{gbest,G+1} = \widehat{X}_{gbest,G} + FC * \widehat{X}_{diff} \\ \widehat{X}_{G+1} = M \bigotimes \widehat{X}_G + Bias \\ Bias = \overline{M} \bigotimes \widehat{X}_{gbest,G+1}. \end{cases} \tag{6}$$

For the global best matrix $\widehat{X}_{gbest}$, all the components are the same with the value equaling to X_{gbest} (X_{gbest} is the coordinate of the particle that finds the global optima in the population). The equations of the two matrices are shown in Eq. 7.

$$\widehat{X} = \begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ X_{ps} \end{bmatrix} \widehat{X}_{gbest,G} = \begin{bmatrix} X_{gbest,G} \\ X_{gbest,G} \\ \cdots \\ X_{gbest,G}, \end{bmatrix} \tag{7}$$

$\widehat{X}_{diff}$ denotes the difference matrix, and it is calculated by the difference of two particle-randomly-permutated matrices of $\widehat{X}$. In other words, $\widehat{X}_{r1}$ and $\widehat{X}_{r2}$ are generated by randomly permutating the row vector of matrix $\widehat{X}$. The equations of $\widehat{X}_{r1}$ and $\widehat{X}_{r2}$ are shown in Eq. 8.

$$\widehat{X}_{r1} = \begin{bmatrix} X_{j1} \\ X_{j2} \\ \cdots \\ X_{jps} \end{bmatrix} \widehat{X}_{r2} = \begin{bmatrix} X_{k1} \\ X_{k2} \\ \cdots \\ X_{kps}. \end{bmatrix} \tag{8}$$

FC is a constant value and it can be considered as the fluctuation coefficient of the difference matrix. $\widehat{X}_G$ denotes the Gth generation of $\widehat{X}$. M is the selection matrix, and $\overline{M}$ denotes the

reverse binary operation of matrix M. The selection matrix is defined in the following equation in Eq. 9. Selection matrix is generated by two consequent steps from an extended lower-triangular matrix M_{tmp} in Eq. 9. The first step is to randomly permutate row elements of M_{tmp}, each row of M_{tmp} is dealt separately. The second step is to randomly permutate the row vectors with each row element unchanged. [1]

$$M_{tmp} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \cdots & & \\ 1 & 1 & \cdots & 1 \\ 1 & & & \\ 1 & 1 & & \\ & \cdots & & \\ 1 & 1 & \cdots & 1 \\ 1 & & & \\ 1 & 1 & & \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & & \\ & \cdots & & \\ 1 & 1 & \cdots & 1 \\ & \cdots & & \\ & & 1 & \\ & \cdots & & \\ 1 & \cdots & 1 & \\ & \cdots & & \\ & 1 & & \\ 1 & & \cdots & 1 \end{bmatrix} = M. \tag{9}$$

We can see that each particle in MKE algorithm only has one mode which means that each particle in the population is equal. The exploration and exploitation are implemented by both exploitation vector and the evolution matrix M. For the ebb-tide-fish algorithm, particles have two modes in the evolution. One mode is for exploitation and the other is for exploration. The exploitation of ebb-tide-fish algorithm is implemented by Eq. 10. Difference vector is first used for the enhancement of ebb-tide-fish algorithm, and then the evolution equation is changed to Eq. 11.

$$\begin{aligned} X^i_{etf,G} &= \{x_1, x_2, \ldots, x_j, \ldots, x_D\}, \\ x_j &= x_j \pm r * rand() * x_j, j \in D. \end{aligned} \tag{10}$$

$$\begin{aligned} X^i_{etf,G} &= \{x_1, x_2, \ldots, x_j, \ldots, x_D\}, \\ x_j &= x_j + FC * (x_{r1} - x_{r2}). \end{aligned} \tag{11}$$

Then we find that the two mode in the particle evolution consumes much time with less performance improved. Then we use the evolution matrix M instead of two modes for particle evolution. Finally, we found the powerful monkey king evolution algorithm. Particles' distribution within 50 generations without fitness value selections over a 20 population size in the search domain $[-5, +5]$ on each dimension of ebb-tide-fish algorithm and monkey king evolutionary algorithm are shown in Fig. 2.

[1] The work mentioned in this manuscript is an earlier work and also submitted earlier than the article [23] published in Knowledge-based Systems. The difference and main contribution of the manuscript is to give some clues for algorithm enhancement. The relation between a former weaker algorithm ebb-tide-fish algorithm and the more powerful algorithm onkey king evolutionary algorithm is shown herein the paper. Another contribution of the paper is the adaptively use of Monkey King Evolutionary (MKE) algorithm for tacking vehicle navigation problem under wireless sensor network environment.

4 Benchmark functions and algorithm verification

To predict the performance of optimization algorithm is difficult as the available theory is limited. Therefore benchmark functions play important roles in analyzing these algorithm. For the verification of algorithms that suit for a broad class of different problems, the carefully selected benchmark functions make more senses than some problem-specified functions. In this section for the verification of proposed MKE algorithm, CEC2013 test suite for real-parameter optimization is used here. The benchmark functions in this test suite are many sophisticated benchmark functions which reflect many real-world applications. All these 28 benchmarks can be split into several groups, such as separate functions, function with low conditioning, functions with high conditioning and unimodal, multimodal functions with global structure, multimodal functions with low global structure, etc., to examine different characteristics of optimization algorithms. All these benchmark functions are rotated and shifted with the global optima locating in a same coordinate. The names, domain and optima are listed in Table 1 while the detailed function equations are listed in Table 2.

We compare IWPSO [5], CCPSO [6], FIPS [24], CLPSO [10], and SLPSO [13] in our experiment for 10-D optimization. The parameter setting of each algorithm is the recommended one, and population size of each compared algorithm is set to a constant number $ps = 100$. Previous work shows that an empirically good constant ω performs better than decreasing ω related to iterations in the range $[0.4, 0.9]$, so we use constant $\omega = 0.5$ for IWPSO, $\chi = 0.7298$, $c_1 = 2.05, c_2 = 2.05$ for CCPSO. The maximum velocity is constrained to maximal range of each dimension as this setting is empirical best. Fluctuation coefficient in MKE algorithm F is set to a constant value ($F = 0.7$).

We run 20 times for each of the benchmark function in CEC2013 test suite with the number of function evaluations equaling to $1,000,000$ ($10,000$ iterations) for each run. The best values and average/standard values of fitness errors ($f_{target} - f_{optima}$) for 10-D optimization are listed in Tables 3 and 4 respectively. From Table 3, we can see that the proposed MKE algorithm can find 8 optima out of the 28 benchmarks and the success rate that algorithm can find the global optimum is much higher than any of the contrasted algorithm. The proposed MKE algorithm performs very well on unimodal functions such as f_1–f_5, and it also performs very well on many multimodal benchmark functions such as f_6, f_9, f_{11}, f_{17}, f_{21}, f_{24}, and f_{28}.

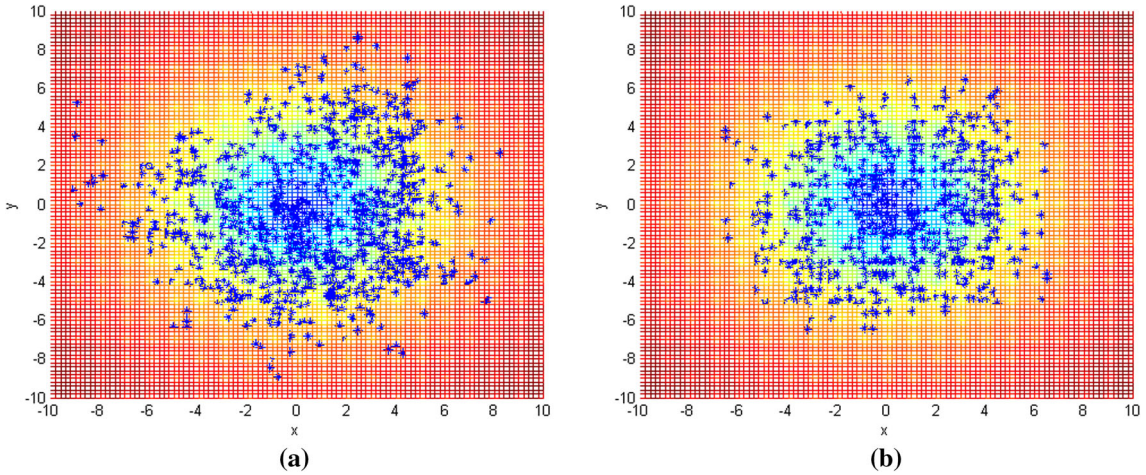


Fig. 2 Empirical particles' positions within 50 generations. Particles are of no selection according to their fitness values. The range of each dimension is $[-5, +5]$, the population size is 20. **a** Ebb-tide-fish algorithm, **b** Monkey King Evolution algorithm

Table 1 Search domain and minimum of CEC2013 benchmark functions

No.	Name	Minimum value
1	Sphere Function	$f(o_1, o_2, \ldots, o_d) = -1400$
2	Rotated High Conditioned Elliptic Function	$f(o_1, o_2, \ldots, o_d) = -1300$
3	Rotated Bent Cigar Function	$f(o_1, o_2, \ldots, o_d) = -1200$
4	Rotated Discuss Function	$f(o_1, o_2, \ldots, o_d) = -1100$
5	Different Powers Function	$f(o_1, o_2, \ldots, o_d) = -1000$
6	Rotated Rosenbrock's Function	$f(o_1, o_2, \ldots, o_d) = -900$
7	Rotated Schaffers F7 Function	$f(o_1, o_2, \ldots, o_d) = -800$
8	Rotated Ackley's Function	$f(o_1, o_2, \ldots, o_d) = -700$
9	Rotated Weierstrass Function	$f(o_1, o_2, \ldots, o_d) = -600$
10	Rotated Griewank's Function	$f(o_1, o_2, \ldots, o_d) = -500$
11	Rastrigin's Function	$f(o_1, o_2, \ldots, o_d) = -400$
12	Rotated Rastrigin's Function	$f(o_1, o_2, \ldots, o_d) = -300$
13	Non-continuous Rotated Rastrigin's Function	$f(o_1, o_2, \ldots, o_d) = -200$
14	Schwefel's Function	$f(o_1, o_2, \ldots, o_d) = -100$
15	Rotated Schwefel's Function	$f(o_1, o_2, \ldots, o_d) = 100$
16	Rotated Katsuura Function	$f(o_1, o_2, \ldots, o_d) = 200$
17	Lunacek Bi-Rastrigin Function	$f(o_1, o_2, \ldots, o_d) = 300$
18	Rotated Lunacek Bi-Rastrigin Function	$f(o_1, o_2, \ldots, o_d) = 400$
19	Expanded Griewank's plus Rosenbrock's Function	$f(o_1, o_2, \ldots, o_d) = 500$
20	Expanded Scaffer's F6 Function	$f(o_1, o_2, \ldots, o_d) = 600$
21	Composition Function1 (n = 5, Rotated)	$f(o_1, o_2, \ldots, o_d) = 700$
22	Composition Function2 (n = 3, Unrotated)	$f(o_1, o_2, \ldots, o_d) = 800$
23	Composition Function3 (n = 3, Rotated)	$f(o_1, o_2, \ldots, o_d) = 900$
24	Composition Function4 (n = 3, Rotated)	$f(o_1, o_2, \ldots, o_d) = 1000$
25	Composition Function5 (n = 3, Rotated)	$f(o_1, o_2, \ldots, o_d) = 1100$
26	Composition Function6 (n = 5, Rotated)	$f(o_1, o_2, \ldots, o_d) = 1200$
27	Composition Function7 (n = 5, Rotated)	$f(o_1, o_2, \ldots, o_d) = 1300$
28	Composition Function8 (n = 5, Rotated)	$f(o_1, o_2, \ldots, o_d) = 1400$
All	Search domain	$[-100, 100]^D$

Table 2 CEC2013 test suite for real-parameter optimization benchmarks

No.	Name	Benchmark function
1	Sphere Function	$f_1(x) = \sum_{i=1}^{D} z_i^2 + f_1^*, Z = X - O$
2	Rotated High Conditioned Elliptic Function	$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2^*, Z = T_{osz}(M_1(X - O))$
3	Rotated Bent Cigar Function	$f_3(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_3^*, Z = M_2 T_{asy}^{0.5}(M_1(X - O))$
4	Rotated Discuss Function	$f_4(x) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + f_4^*, Z = T_{osz}(M_1(X - O))$
5	Different Powers Function	$f_5(x) = \sqrt{\sum_{i=1}^{D} \lvert z_i \rvert^{2+4\frac{i-1}{D-1}}} + f_5^*, Z = X - O$
6	Rotated Rosenbrock's Function	$f_6(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_6^*, Z = M_1 \left(\frac{2.048(X-O)}{100}\right) + 1$
7	Rotated Schaffers F7 Function	$f_7(x) = \left(\frac{1}{D-1} \sum_{i=1}^{D-1} (\sqrt{z_i} + \sqrt{z_i} sin^2(50 z_i^{0.2}))\right)^2 + f_7^*,$ $z_i = \sqrt{y_i^2 + y_{i+1}^2}, Y = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(X - O))$
8	Rotated Ackley's Function	$f_8(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1} Dcos(2\pi z_i)\right) + 20 + e + f_8^*$ $Z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(X - O))$
9	Rotated Weierstrass Function	$f_9(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{kmax} [a^k cos(2\pi b^k (z_i + 0.5))]\right) - D \sum_{k=0}^{kmax} [a^k cos(2\pi b^k 0.5)] + f_9^*$ $a = 0.5, b = 3, kmax = 20, Z = \Lambda^{10} M_2 T_{asy}^{0.5} \left(M_1 \frac{0.5(X-O)}{100}\right)$
10	Rotated Griewank's Function	$f_{10}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{10}^*,$ $Z = \Lambda^{100} M_1 \frac{600(X-O)}{100}$
11	Rastrigin's Function	$f_{11}(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_{11}^*,$ $Z = \Lambda^{10} T_{asy}^{0.2} \left(T_{osz}\left(\frac{5.12(X-O)}{100}\right)\right)$
12	Rotated Rastrigin's Function	$f_{12}(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_{12}^*,$ $Z = M_1 \Lambda^{10} M_2 T_{asy}^{0.2} \left(T_{osz}\left(M_1 \frac{5.12(X-O)}{100}\right)\right)$
13	Non-continuous Rotated Rastrigin's Function	$f_{13}(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_{13}^*, Z = M_1 \Lambda^{10} M_2 T_{asy}^{0.2}(T_{osz}(Y))$ $\hat{x} = M_1 \frac{5.12(X-O)}{100}, y_i = \begin{cases} \hat{x_i}, & if \lvert \hat{x_i} \rvert \le 0.5 \\ round(2\hat{x_i})/2, & if \lvert \hat{x_i} \rvert > 0.5 \end{cases}$
14	Schwefel's Function	$f_{14}(Z) = 418.9829 * D - \sum_{i=1}^{D} g(z_i) + f_{14}^*,$ $Z = \Lambda^{10}\left(\frac{100(X-O)}{100}\right) + 4.209687462275036e + 002$
15	Rotated Schwefel's Function	$f_{15}(Z) = 418.9829 * D - \sum_{i=1}^{D} g(z_i) + f_{15}^*,$ $Z = \Lambda^{10} M_1\left(\frac{100(X-O)}{100}\right) + 4.209687462275036e + 002$
16	Rotated Katsuura Function	$f_{16}(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum j = 132 \frac{\lvert 2^j z_i - round(2^j z_i) \rvert}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{16}^*$ $Z = M_2 \Lambda^{100} \left(M_1\left(\frac{5(X-O)}{100}\right)\right)$
17	Lunacek Bi-Rastrigin Function	$f_{17}(x) = min\left(\sum_{i=1}^{D} y_0^2, dD + s\sum_{i=1}^{D} y_1^2\right) + 10\left(D - \sum_{i=1}^{D} cos(2\pi \hat{z}_i)\right) + f_{17}^*$

Table 2 continued

No.	Name	Benchmark function
		$y_0 = (\hat{x}_i - \mu_0), y_1 = (\hat{x}_i - \mu_1), z = \Lambda^{100}(\hat{x} - \mu_0)$
18	Rotated Lunacek Bi-Rastrigin Function	$f_{18}(x) = min\left(\sum_{i=1}^{D} y_0^2, dD + s\sum_{i=1}^{D} y_1^2\right) + 10\left(D - \sum_{i=1}^{D} cos(2\pi\hat{z}_i)\right) + f_{18}^*$
		$y_0 = (\hat{x}_i - \mu_0), y_1 = (\hat{x}_i - \mu_1), z = M_2\Lambda^{100}(M_1(\hat{x} - \mu_0))$
19	Expanded Griewank's plus Rosenbrock's Function	$f_{19}(x) = g_1(g_2(z_1, z_2)) + g_1(g_2(z_2, z_3)) + \cdots + g_1(g_2(z_D, z_1)) + f_{19}^*$
		$g_1(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1, z = M_1\left(\frac{5(x-o)}{100}\right) + 1$
20	Expanded Scaffer's F6 Function	$f_{20}(x) = g(z_1, z_2) + g(z_2, z_3) + \cdots + g(z_D, z_1) + f_{20}^*$
		$g(x, y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2}) - 0.5}{(1 + 0.001(x^2+y^2))^2}, Z = M_2 T_{asy}^{0.5}(M_1(X - O))$
21	Composition Function 1	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_i = f_6', g_2 = f_5', g_3 = f_3', g_4 = f_4', g_5 = f_1'$
22	Composition Function 2	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_{1-3} = f_{14}'$
23	Composition Function 3	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_{1-3} = f_{15}'$
24	Composition Function 4	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_9', \sigma = [20, 20, 20]$
25	Composition Function 5	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_9', \sigma = [10, 30, 50]$
26	Composition Function 6	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_2', g_4 = f_9', g_5 = f_{10}'$
27	Composition Function 7	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_1 = f_{10}', g_2 = f_{12}', g_3 = f_{15}', g_4 = f_9', g_5 = f_1'$
28	Composition Function 8	$f(x) = \sum_{i=1}^{n} \omega_i * [\lambda_i g_i(x) + bias_i] + f^*$
		$f_i' = f_i - f_i^*, g_1 = f_{19}', g_2 = f_7', g_3 = f_{15}', g_4 = f_{20}', g_5 = f_1'$

The overall performance of our proposed MKE algorithm is much better than other contrasted state-of-the-art PSO variants. We can see that many of the contrasted algorithms can find global optima of function f_1 and f_5 from Table 4, so we figure out the convergence curve of these algorithms in Figs. 3 and 4. The average iteration for the contrasted algorithms to find the optima of these two benchmark functions are shown in Table 5.

5 Vehicle navigation under wireless sensor network environment

With the development of micro-electronic technology, wireless sensor networks have been widely used in many applications. Intelligent transportation systems under a wireless sensor network environment show tremendous advantageous in vehicle information collecting, vehicle localization and tracking. As more and more vehicles are traveling on the road nowadays, the desire of a better travel path attributes to the development of vehicle navigation technique. In this paper, the proposed monkey king algorithm is utilized to tackle the vehicle navigation problem aiming at less travel time and overall better throughput of a micro-scope city traffic networks under wireless sensor network environment. SUMO [25,26] platform is used in our simulation, and Fig. 5 shows a small region of the real traffic networks in Shenzhen city on SUMO platform. A grid network on SUMO platform is also used in our simulation and the related figure is shown in Fig. 6.

To achieve an efficient and robust navigation, end device sensors are emplaced in the intersections of the road networks to grasp traffic information. Road distance between two intersections, lanes of the road, maximum velocity restriction, moving direction, safe distance between two vehicles, are all collected and used in traffic navigation. In the simulation,

Table 3 Comparison results of best values in 20 runs with the same population size (100) and number of function evaluations

D = 10	IWPSO	CCPSO	FIPS_Uring	CLPSO	SLPSO	MKE
1	***0***	***0***	***0***	***0***	***0***	***0***
2	6.2219E+03	1.5341E+03	2.5958E+05	1.0742E+05	3.1307E+03	**0**
3	8.4504E+00	7.2012E−03	9.3183E+02	2.5073E+04	1.0975E−01	**0**
4	2.3032E−03	1.0383E−05	1.0442E+03	8.1690E+02	3.0682E+02	**0**
5	***0***	***0***	***0***	***0***	***0***	***0***
6	1.0407E−01	5.8119E−03	2.9921E+00	5.7861E−02	7.4780E−03	**0**
7	5.4720E−01	7.6699E−01	1.4475E−01	2.7911E+00	8.2545E−08	**2.6603E−11**
8	2.0109E+01	2.0142E+01	2.0135E+01	2.0108E+01	**2.0104E+01**	2.0227E+01
9	7.5826E−01	6.6214E−01	1.1535E+00	2.0014E+00	***0***	***0***
10	1.4268E−01	9.1142E−02	5.4348E−01	1.4956E−01	**9.8647E−03**	4.917E−02
11	***0***	9.9496E−01	1.2921E+00	***0***	9.9496E−01	***0***
12	5.9698E+00	4.9748E+01	1.9855E+01	2.2242E+00	**9.9496E−01**	4.7202E+00
13	6.3297E+00	3.1902E+00	2.1921E+01	4.1636E+00	***9***.9496E−***0***1	***9***.9496E−***0***1
14	3.5399E+00	3.5399E+00	2.4878E+02	**0**	2.4982E−01	3.8531E+00
15	1.6691E+02	8.6620E+01	8.4792E+02	3.4004E+02	**6.8924E+00**	2.9998E+02
16	**1.4823E−01**	1.5265E−01	5.9590E−01	3.0200E−01	6.7894E−01	6.1354E−01
17	1.0312E+01	1.7721E+00	2.4666E+01	2.9551E+00	1.0615E+01	**3.9636E−01**
18	1.2079E+01	**4.8434E+00**	3.0724E+01	1.7227E+01	1.9127E+01	1.2956E+01
19	3.0890E−01	6.9478E−02	1.3817E+00	**5.3203E−02**	5.4299E−01	2.8013E−01
20	**8.9747E−01**	1.7711E+00	2.5448E+00	1.9101E+00	1.4036E+00	1.0848E+00
21	2.0000E+02	2.0000E+02	2.3411E+02	4.1487E+00	4.0019E+02	**1.0000E+02**
22	1.7541E+01	4.1743E+01	1.9707E+02	**5.1004E+00**	1.1181E+01	3.1368E+01
23	3.8489E+02	3.7587E+02	1.0744E+03	2.6518E+02	**2.3024E+01**	1.5338E+02
24	2.0083E+02	1.2547E+02	2.0680E+02	1.1540E+02	2.0000E+02	**1.1186E+02**
25	2.0154E+02	2.0071E+02	2.0814E+02	**1.2615E+02**	2.0000E+02	2.0000E+02
26	1.0398E+02	1.0497E+02	1.2904E+02	1.0890E+02	**1.0199E+02**	1.0398E+02
27	3.0110E+02	3.5496E+02	3.5674E+02	**2.9236E+02**	3.0000E+02	3.0000E+02
28	3.0000E+02	3.0000E+02	1.5215E+02	1.1715E+02	3.0000E+02	**1.0000E+02**
Win	2	1	0	5	4	9
Draw	3	2	2	3	6	5
Total	5	3	2	8	10	14

The best result of each function is emphasized in boldface and the best draw results of each function is highlighted in italic fonts

ten thousand cars are randomly dispersed on the 8×8 grid network shown in Fig. 6, and the distance between two neighbor intersection node is 2 km. Congestion is happened in two predefined situations, one is that there are more than two cars within a safe distance of a single lane in the networks, the other is that there are more than 89 cars in the interval between two neighbor intersection nodes. In our simulation, the maximum velocity of the urban area is set to 72 km/h for easy calculation, and the safe distance is set to 70 m. Four different techniques are used for the navigation contrasts, and they are Dijkstra algorithm, A star algorithm, ebb-tide-fish algorithm [17] and the proposed monkey king algorithm in the paper.

In the simulation, there are 64 intersection nodes, and the navigation can be transformed into a 64-D optimization problem for least travel time of a specified vehicle under real-time traffic condition. The path from the start to the destination is within a sequence such as $Path_i$: $\{x_1, \ldots, 5, \ldots, x_k, \ldots, 59, \ldots, x_{64}\}, x_k \in [1, 64]$. The fitness function of travel time is calculated in Eq. 12. $g(\omega_{i,j})$ denotes the weight function of the edge between two intersection nodes i and j. $T_{i,j}$ is a function of the traffic density obtained by sensor nodes in the wireless sensor network, and it means the time consumed when traveling the interval between intersection nodes i and j while T_{delay} denotes the time delay crossing the intersection. The weight function $g(\omega_{i,j})$ is calculated with regard to traffic density and the value is stored in the adjacent matrix of the nodes. If there is no directly connected edge between two intersection nodes, the weight is set to $\omega_{i,j} = \infty$, and if any edge in the

Table 4 Comparison results of mean/standard deviation of 20 runs with the same population size and number of function evaluations

D = 10.	IWPSO	CCPSO	FIPS_Uring	CLPSO	SLPSO	MKE
1	***0/0***	5.6843E−14/1.0101E−13	***0/0***	***0/0***	***0/0***	***0/0***
2	1.4943E+05/1.8848E+05	2.0926E+04/1.8434E+04	8.3832E+04/2.6904E+05	2.8651E+05/1.8413E+05	7.7718E+04/1.6364E+05	2.2291E−14/6.8285E−14
3	1.6815E+06/3.7625E+07	4.1743E+06/1.3409E+07	7.7936E+03/8.5641E+03	2.4655E+05/1.9573E+05	1.2502E+06/2.1122E+06	**4.5959E−03/1.7094E−02**
4	4.5055E−02/5.1575E−02	1.9519E−04/5.7786E−04	1.9171E+03/5.1283E+02	1.8794E+03/5.9328E+02	4.0868E+03/3.3092E+03	**0/0**
5	7.9580E−14/5.3451E−14	7.9580E−14/5.3451E−14	***0/0***	***0/0***	5.6853E−15/2.5421E−14	***0/0***
6	9.3974E+00/1.1850E+01	8.6092E−01/1.6391E+00	3.8760E+00/5.3077E−01	**1.6891E−01/7.0527E+00**	8.9266E+00/3.0507E+00	4.4064E+00/4.8315E+00
7	1.7024E+01/2.2056E+01	1.4176E+01/1.1512E+01	**6.2262E−01/3.8467E−01**	5.9567E+00/1.7804E−00	1.4846E+00/2.5045E+00	7.0377E−01/2.1134E+00
8	**2.0213E+01/5.7059E−02**	2.0215E+01/4.4255E−02	2.0248E+01/4.3374E−02	2.0220E+01/5.7137E−02	2.0222E+01/5.8950E−02	2.0457E+01/8.6529E−02
9	3.9963E+00/1.5776E+00	3.4175E+00/1.7503E+00	3.6769E+00/9.6284E−01	3.0316E+00/5.0727E−01	2.1900E+00/1.2675E+00	**2.0279E+00/1.4519E+00**
10	4.9989E−01/2.9794E−01	3.4498E−01/1.8439E−01	6.3301E−01/5.8004E−02	3.1259E−01/8.0042E−02	1.9282E−01/1.3326E−01	**1.0887E−01/8.8611E−02**
11	1.7411E+00/9.6167E−01	3.5321E+00/3.1210E+00	8.6795E+00/3.8262E+00	**0/0**	2.6864E+01/1.1684E+E+00	3.1214E+00/1.8022E+00
12	1.5064E+01/5.5123E+00	1.6018E+01/6.7319E+00	2.7347E+01/3.4252E+00	5.5914E+00/1.6202E+00	4.7758E+00/1.8146E+00	**1.3973E+00/6.7581E+00**
13	1.8486E+01/6.5294E+00	1.8805E+01/8.2768E+00	2.6346E+01/2.6632E+00	9.1990E+00/2.2030E+00	**7.8077E+00/7.1979E+00**	1.9075E+01/8.6885E+00
14	1.2585E+02/9.7576E+01	1.9985E+02/1.3954E+02	5.2574E+02/1.1005E+02	**0/0**	3.8592E+01/5.0128E+01	1.2560E+02/1.1706E+02
15	6.3640E+02/2.5248E+02	7.0312E+02/2.7108E+02	1.1446E+03/1.1770E+02	5.0029E+02/8.4036E+01	**2.3116E+02/1.2805E+02**	1.0043E+03/3.2006E+02
16	4.6522E−01/1.9935E−01	**3.6472E−01/1.3443E−01**	7.9505E−01/1.1954E−01	6.7487E−01/1.1728E−01	8.3488E−01/1.0442E−01	1.2167E+00/3.5948E−01
17	1.1447E+01/1.0458E+00	1.1175E+01/2.5557E+00	3.2791E+01/4.0305E+00	8.7542E+00/2.4160E+00	1.2048E+01/7.1478E−01	**8.7361E+00/4.3007E+00**
18	2.2961E+01/6.7136E+00	**1.9318E+01/5.4133E+00**	3.7400E+01/3.6474E+00	2.1875E+01/2.2914E+00	2.3609E+01/2.3143E+00	3.1576E+01/8.1329E+00
19	5.6809E−01/1.7289E−01	5.3041E−01/1.8566E−01	1.9199E+00/3.0975E−01	**1.5905E−01/6.0262E−02**	8.7893E−01/1.9081E−01	5.9791E−01/1.3885E−01
20	2.9026E+00/6.2217E−01	3.1800E+00/6.9959E−01	2.7682E+00/1.2796E−01	2.4449E+00/2.5033E−01	1.8556E+00/3.4338E−01	**1.8075E+00/5.01660E−01**
21	3.6515E+02/7.4593E+01	3.9018E+02/4.4765E+01	3.7925E+02/5.0392E+01	**1.5772E+02/5.4660E+01**	4.0019E+02/1.7496E−13	3.7665E+02/7.37814E+01
22	1.9603E+02/1.1511E+02	2.4275E+02/1.4075E+02	5.0891E+02/1.4705E+02	**1.0170E+01/3.1552E+00**	5.1578E+01/2.9219E+01	9.0642E+01/7.4036E+01
23	8.6764E+02/2.6074E+02	8.7279E+02/2.6247E+02	1.2210E+03/9.4446E+01	5.6413E+02/1.0927E+02	1.5830E+03/1.7295E+02	9.8230E+02/3.1535E+02
24	2.1030E+02/4.8793E+00	2.0488E+02/1.9098E+02	2.1135E+02/2.2063E+00	**1.2301E+02/6.2363E+00**	2.0455E+02/5.5018E+00	2.0435E+02/1.3891E+01
25	2.1379E+02/4.4460E+00	2.0884E+02/3.8450E+00	2.1301E+02/2.1873E+00	**1.4422E+02/1.5950E+01**	2.0193E+02/2.4008E+00	2.0553E+02/5.2419E+00
26	1.8572E+02/6.8971E+01	1.8191E+02/7.0252E+01	1.4337E+02/1.5172E+01	1.1447E+02/3.4262E+00	1.4740E+02/4.8850E+01	**1.1269E+02/4.9148E+00**
27	4.5301E+02/1.0575E+02	4.0298E+02/2.8914E+01	4.1130E+02/2.7367E+01	3.2683E+02/1.1156E+01	**3.0000E+02/1.1687E−02**	3.2536E+02/8.9250E+01
28	3.0000E+02/0	3.5024E+02/1.2687E+02	2.9260E+02/3.3060E+01	**1.4081E+02/1.7011E+01**	3.0000E+02/0	2.9608E+02/2.8006E+01
Win	1	2	1	9	3	9
Draw	1	0	2	2	1	2
Total	2	2	3	11	4	11

The best result of each function is emphasized in boldface and the best draw results of each function is highlighted in italic fonts

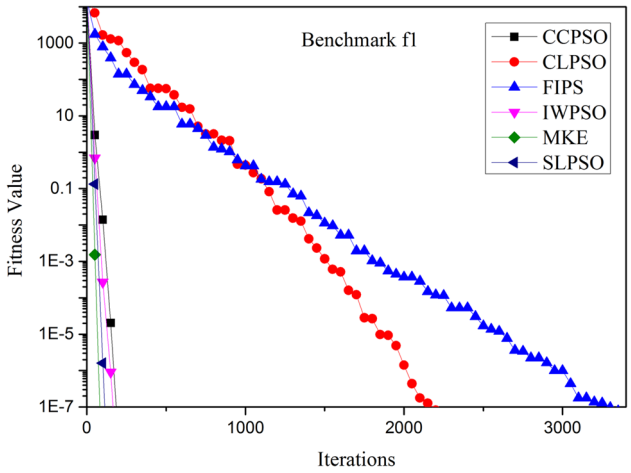


Fig. 3 Convergence curve of different algorithm on function f1

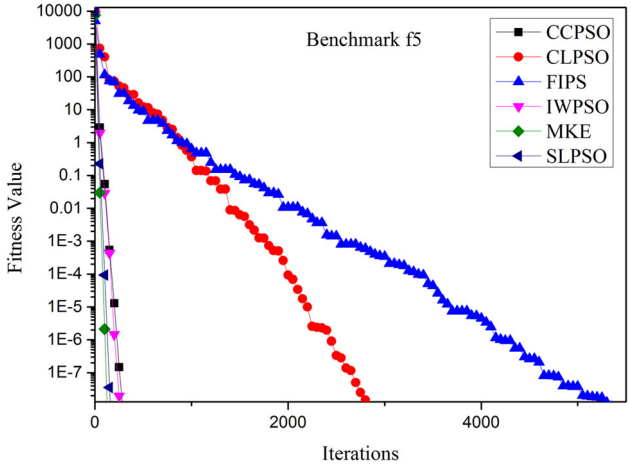


Fig. 4 Convergence curve of different algorithm on function f5

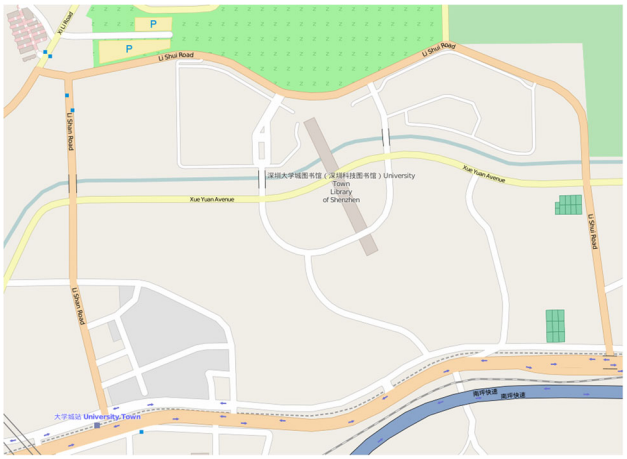


Fig. 5 Small region of real traffic networks in Shenzhen city on SUMO platform

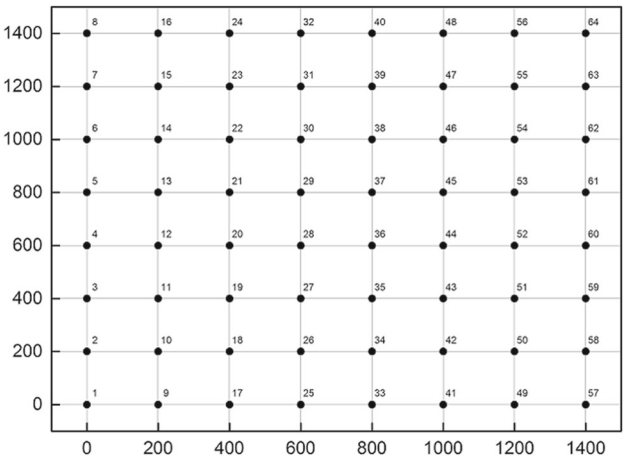


Fig. 6 Grid network used in our simulation for vehicle navigation

solution candidate is out of the edges in the navigation, the corresponding weight $\omega_{i,j}$ is set to 0. Fig. 7 shows once of navigation of the grid network simulation. The yellow point is the start, the pink point is the destination, the red points denote congestion, and the green points denote the available path of navigation.

$$f(x_1, x_2, \ldots, x_{64}) = \sum_{i=1,j=i+1}^{63} (g(\omega_{i,j}) * T_{i,j} + T_{delay}). \tag{12}$$

In the monkey king algorithm based navigation, each particle is initialized by a randomly generated traversing sequence of the nodes in the simulation, and after calculation of the fitness values, we can locate the temporal global optima. Hamming distance [27] in Eq. 13 is used for candidate updating scheme. The distance between two solution candidates is calculated in the hamming distance way with the equation shown in Eq. 13. $\oplus$ means XOR operation, and x_j and x_{gbest_j} mean the jth component of X_i solution candidate and global best candidate respectively. Equation 6 can be rewritten in a hamming distance way as illustrated in Eq. 14. $\widehat{X}_{diff}$ denotes the hamming distance between $\widehat{X}_{r1}$ and $\widehat{X}_{r2}$. $\widehat{X}_{gbest,G+1}$ is

Table 5 Iteration needed to find the global optima over different algorithms on function f_1 and f_5

D = 10	IWPSO	CCPSO	FIPS_Uring	CLPSO	SLPSO	MKE
f_1	268	–	5314	2969	192	128
f_5	–	–	8776	4310	–	195

"–" means that optima sometimes can not be found during the maximum iterations in the total 20 runs

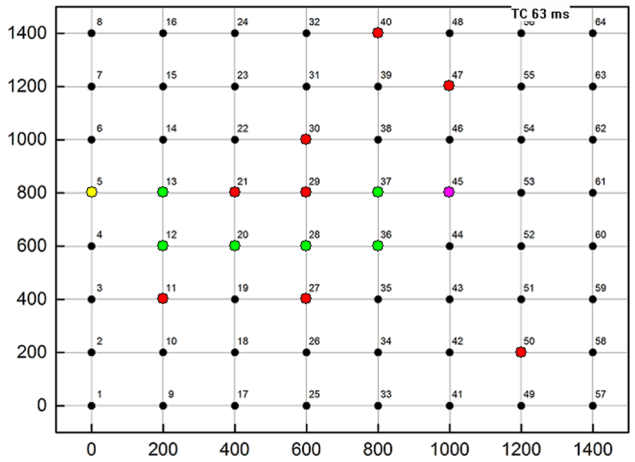


Fig. 7 One navigation of randomly generated start and destination in the grid network simulation

Table 6 The comparison of average fuel consumption and time consumption for 200 times navigation

Algorithms	Travel time cost (h)
Dijkstra	1.6
A*	1.6
Ebb-tide-fish	1.1
Monkey king algorithm	1.1

Table 7 The comparison of average fuel consumption and time consumption for 2000 times navigation

Algorithms	Travel time cost (h)
Dijkstra	1.9
A*	1.9
Ebb-tide-fish	1.4
Monkey king algorithm	1.2

a generated coordinate which satisfies the hamming distance $d(\widehat{X}_{gbest,G+1}, \widehat{X}_{gbest,G})$ equaling to $F * \widehat{X}_{diff}$.

$$d(X_{gbest}, X_i) = \sum_{j=1}^{d} x_{gbest_j} \oplus x_j \tag{13}$$

$$\begin{cases} \widehat{X}_{diff} = d(X_{r1}, X_{r2}) \\ d(\widehat{X}_{gbest,G+1}, \widehat{X}_{gbest,G}) = F * \widehat{X}_{diff} \\ \widehat{X}_{G+1} = M \otimes \widehat{X}_G + Bias \\ Bias = \overline{M} \otimes \widehat{X}_{gbest,G+1}. \end{cases} \tag{14}$$

In our simulation, we conduct two experiments, we run 200 times to calculate the average travel time for the first experiment, and run 2000 times for the second. The comparison results are shown in Tables 6 and 7 respectively. Both Dijkstra algorithm and A star algorithm can find the shortest pathes in these two experiments, and they consumed the same travel time to complete the journey from the start to the destination. There are the cases that congestion is in the shortest path, so the least travel time navigation may be not the shortest path. From the two tables, we also can see that the travel time of ebb-tide-fish algorithm and monkey king algorithm is the same in experiment one (results shown in Table 6, as the least travel time path can be found both of these two algorithm in a predefined time span. In the second experiment, the monkey king algorithm performs good as ebb-tide-fish algorithm can not find the least time path within the restricted time.

6 Conclusion

In this paper, we propose MKE algorithm, and it is an enhanced version of the former proposed ebb-tide-fish algorithm for global optimization. The new proposed algorithm in this paper is a much powerful technique to tackle optimization problems, CEC2013 test suite for real-parameter optimization benchmarks is used to verify the characteristic of the new algorithm. Contrasts are made between state-of-the-art PSO variants (including IWPSO, CCPSO, FIPS, CLPSO, SLPSO) and the proposed algorithm, and experiment results show that our proposed algorithm performs very well not only on unimodal optimization functions but also on multimodal optimization functions. An application of MKE algorithm to tackle vehicle navigation under wireless sensor network environment is also shown in the paper, and experiment results show that it also performs good on this application.

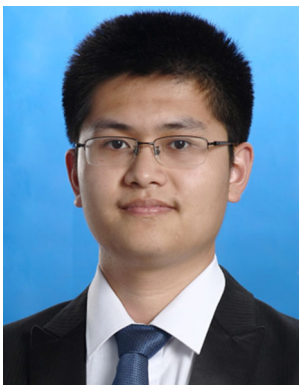
Acknowledgements The authors extend their appreciation to National Natural Science Foundation of China (61273290) for funding this work.

References

1. Storn, R., & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
2. Sörensen, Kenneth. (2015). Metaheuristicsthe metaphor exposed. *International Transactions in Operational Research*, *22*(1), 3–18.
3. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942–1948).
4. Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (Vol. 1, pp. 39–43).
5. Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary Computation Proceedings 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on* (pp. 69–73). IEEE.

6. Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (Vol. 1, pp. 84–88). IEEE.
7. Clerc, M., & Kennedy, J. (2002). The particle swarmexplosion, stability, and convergence in a multidimensional complex space. *IEEE Transaction on Evolutionary Computation*, *6*(1), 58–73.
8. Mendes, R., Kennedy, J., & Neves, J. (2003). Watch thy neighbor or how the swarm can learn from its environment. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)* (p. 88C94). Piscataway: IEEE.
9. Liang, J. J., & Suganthan, P. N. (2005). Dynamic multi-swarm particle swarm optimizer. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE* (pp. 124–129). IEEE.
10. Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, *10*(3), 281–295.
11. Bratton, D., & Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007*. SIS 2007. IEEE (pp. 120–127). IEEE.
12. Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., & Suganthan, P. N. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences*, *209*, 16–36.
13. Cheng, R., & Jin, Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, *291*, 43–60.
14. Chu, S. C., Tsai, P. W., Pan, J. S., & (2006). Cat swarm optimization. In PRICAI. *Trends in artificial intelligence* (pp. 854–858). Berlin: Springer.
15. Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. *In nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Berlin: Springer.
16. Meng, Z., & Pan, J. S. (2015). A simple and accurate global optimizer for continuous spaces optimization. In D. E. Goldberg & J. R. Koza (Eds.), *Genetic and evolutionary computing* (pp. 121–129). New York: Springer International Publishing.
17. Meng, Z., Pan, J. S., & Alelaiwi, A. (2015). A new meta-heuristic ebb-tide-fish-inspired algorithm for traffic navigation. *Telecommunication Systems*, *62*, 1–13.
18. Boukerche, A., Oliveira, H. A., Nakamura, E. F., & Loureiro, A. A. (2008). Vehicular ad hoc networks: A new challenge for localization-based systems. *Computer Communications*, *31*(12), 2838–2849.
19. Koutsonikolas, D., Das, S. M., & Hu, Y. C. (2007). Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications*, *30*(13), 2577–2592.
20. Li, H., Wang, J., Li, X., & Ma, H. (2008). Real-time path planning of mobile anchor node in localization for wireless sensor networks. In *Information and Automation, 2008. ICIA 2008. International Conference on* (pp. 384-3-89). IEEE.
21. Kuriakose, J., Joshi, S., Raju, R. V., & Kilaru, A. (2014). A review on localization in wireless sensor networks. In S. M. Thampi, A. Gelbukh, & J. Mukhopadhyay (Eds.), *Advances in signal processing and intelligent recognition systems* (pp. 599–610). New York: Springer International Publishing.
22. Qi, L. (2008). Research on intelligent transportation system technologies and applications. In *Power Electronics and Intelligent Transportation System, 2008*. PEITS'08. Workshop on (pp. 529–531). IEEE.
23. Meng, Zhenyu, & Pan, Jeng-Shyang. (2016). Monkey King Evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. *Knowledge-Based Systems*, *97*, 144–157.
24. Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, *8*(3), 204–210.
25. Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). SUMOC simulation of Urban Mobility. In *The Third International Conference on Advances in System Simulation* (SIMUL 2011), Barcelona.
26. Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMOC simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, *5*(3&4), 128–138.
27. Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, *29*(2), 147–160.

Jeng-Shyang Pan received the B.S. degree in Electronic Engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in Communication Engineering from the National Chiao Tung University, Taiwan in 1988, and the Ph.D. degree in Electrical Engineering from the University of Edinburgh, U.K. in 1996. Currently, he is a Dean for College of Information Science and Engineering, Fujian University of Technology, also a professor of Fuzhou University of International Studies and Trade and a director of Innovative Information Industry Research Center, Harbin Institute of Technology Shenzhen Graduate School, China. He jointed the editorial board of LNCS Transactions on Data Hiding and Multimedia Security, Journal of Computers, Journal of Information Hiding and Multimedia Signal Processing etc. His current research interests include soft computing, information security and signal processing.



Zhenyu Meng received the B.S. degree in Computer Science, Shandong Normal University in 2008, and the master degree in Computer Science from Harbin Institute of Technology in 2011. After the graduation of his master degree, he worked as a research assistant in Guangzhou Institute of Advanced Technology, Chinese Academy of Sciences during 2011–2013. Now he is a Ph.D. student in Harbin Institute of Technology Shenzhen Graduate School, and his research interest includes computer vision, computational intelligence, vehicle navigation.

Shu-Chuan Chu received the Ph.D. degree from Flinders University of South Australia, Australia in 2004. She was an Associate Professor at Cheng Shiu University, Taiwan. Currently, she is with the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia, Australia. Her research interests include pattern recognition and data mining, computational intelligence, cloud computing.

Hua-Rong Xu received the M.S. degree and Ph.D. degree in Computer Science from Ximen University in 2003 and 2013 respectively. His current research interests include computer vision, artificial intelligence and embedded system.