Wireless Networks
https://doi.org/10.1007/s11276-021-02563-9

A parallel compact cat swarm optimization and its application in DV-Hop node localization for wireless sensor network

Jianpo Li[1] · Min Gao[1] · Jeng-Shyang Pan[1,2] · Shu-Chuan Chu[2,3]

Accepted: 29 January 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract
Cat swarm optimization (CSO) has been applied to a variety of fields because of the better capacity of searching for optimum and higher robustness. However, the poor convergency and larger memory consumption are still core defects, which restricts the efficiency of optimization to a larger extent. A new heuristic algorithm named Parallel Compact Cat Swarm Optimization (PCCSO) with three separate communication strategies and the concept of the compact are presented in this article. The advantage of PCCSO is not only reflected in enhancing the ability of local search, but also in saving the computer memory. The experimental results on CEC2013 benchmark functions demonstrate that the PCCSO is always superior to PSO, CSO, and improved CSO in getting convergent. Then, the PCCSO is applied to DV-Hop to effectively improve the localization accuracy of unknown nodes while also saving WSN memory. The experimental results based on PCCSO from the different number of sensor nodes also illustrate that the PCCSO-DV-Hop shows a lower localization error compared to other optimization algorithms based on DV-Hop.

Keywords CSO · Parallel compact strategy · Wireless sensor networks · DV-Hop

1 Introduction

Recently, with the development of optimization theory and methodology, the global optimization algorithms have been widely applied to various fields and have attracted widespread attention [1, 2]. As meta-inspired heuristic algorithms, they have been confirmed to be effective in proposing global optimal solutions and solving the limitations of conventional methods of robust optimization. Furthermore, heuristic algorithms show an excellent ability in robustness, which can solve problems under different conditions, and have the advantages of wide application range, high nonlinearity, and easy modification at the same time. There are many heuristic algorithms with practical effects. For example: classic particle swarm optimization (PSO) [3, 4], grey wolf optimizer (GWO) [5–7], symbiotic organisms search (SOS) [8, 9], genetic algorithm (GA) [10] and ant colony optimization (ACO) [11], etc.

Cat Swarm Optimization (CSO) simulates the routine behavior patterns of cats, which was first proposed by Chu et al. in 2006 [12]. On the one hand, cats are usually more alert when they are idle in daily life. Consequently, the cats constantly look around to find target prey, which is called seeking mode. On the other hand, when the cat finds the target, it walks quietly step by step and finally seizes the opportunity to capture the prey, which is called the tracing mode. The optimization efficiency of CSO is greatly improved by using a certain proportion of cats to execute tracing mode, and the others seeking mode. Then, cats randomly select the cats of the two modes in each algorithm iteration.

However, the huge memory consumption of CSO algorithm limits its convergence speed and optimization accuracy. Many compact solutions have been proposed, such as compact particle swarm optimization (cPSO) [13], compact genetic algorithm (cGA) [14], compact differential evolution algorithm (cDE) [15], compact pigeon-

✉ Jeng-Shyang Pan
jengshyangpan@gmail.com

[1] School of Computer Science, Northeast Electric Power University, Jilin, China

[2] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China

[3] College of Science and Engineering, Flinders University, Clovelly Park, SA 5042, Australia

Published online: 15 February 2021

inspired optimization (cPIO) [16], compact bat algorithm (cBA) [17, 18], compact cuckoo search algorithm (cCS) [19] and compact cat swarm optimization (cCSO) [20], etc [21, 22]. Therefore, the application of compact not only improves the convergence of the algorithm, but also saves the computer memory [23].

The WSN perceives the surrounding information by sensor nodes [24–26]. Modern wireless communication technology and hardware conditions of sensor nodes continue to promote the application of wireless sensor networks in China, such as health care system, smart home system, intelligent transportation system and so on [27–29]. Besides, other wireless communication applications perform well. For example, In [30], the ant colony-based reinforcement learning algorithm is applied to the routing of wireless sensor networks to achieve the purpose of enhancing the success rate and energy consumption of sensor networks. In [31], the application of game theory to wireless sensor networks has identified existing problems and future trends for researchers. In [32], unmanned aerial vehicle (UAV) support is combined with wireless power communication technology to improve energy efficiency while enhancing its scalability and robustness. In [33], the evacuation planning problem is dealt with through the clustering of public safety systems and autonomously operated evacuation strategies. In [34], machine learning and intelligent communication are used in wireless sensor networks to solve current problems. Not only do they improve the connection between humans and the information world, but the interaction of information also provides great convenience for modern life. As a result, many researchers have applied CSO to wireless sensor networks (WSN). In [35], it adopts a ladder diffusion algorithm and cat swarm optimization algorithm to realize the energy-aware routing protocol of the wireless sensor network. In [36] proposes a cat swarm optimization method based on wavelet transform to optimize the configuration of deterministic sensors. In [37] uses the cat swarm optimization to optimize the vehicular ad-hoc networks to improve road safety. In [38] uses the enhanced parallel cat swarm optimization to improve the deployment of wireless sensor networks to reduce node energy consumption, which proves that the parallel method can optimize the algorithm well.

The distance vector-hop (DV-Hop) is a WSN localization method that does not require additional hardware devices to directly measure the distance between nodes [39, 40]. In terms of implementation, the DV-Hop only depends upon the connectivity degree of the entire network. Therefore, the implementation of DV-Hop is relatively simple, and the cost of establishing the network is moderate. However, there are some errors between the actual position and the estimated position of unknown nodes.

In this paper, an improved CSO is presented in the DV-Hop localization of WSN. Firstly, to speed up the convergence rate, a common parallel strategy that divides solutions into groups to perform their tasks and communicate within a specified number of iterations [41, 42] is proposed. Then, a novel compact scheme is introduced to the process of communication to further improve the speed and accuracy of optimization. Finally, this paper uses a compact and parallel combination to improve CSO to achieve the goal of saving storage space while effectively solving problems. The main contributions of this article are described as several points:

1. An improved parallel compact CSO (PCCSO) is proposed to solve the problems of convergence speed and memory space of the original algorithm.
2. Three novel parallel strategies are proposed and implemented, and the proposed algorithm uses the CEC 2013 test function to compare with the improved parallel CSO (PCSO).
3. The improved CSO in this paper uses some selected test function suites to compare with the traditional PSO algorithm to prove its excellent experimental effect.
4. The PCCSO carried out the original algorithm and applied it to DV-Hop localization in WSN, which not only improved the localization accuracy but also saved WSN storage space.

The remaining arrangements are as follows. The second section reviews the CSO algorithm. The third section gives details of the PCCSO algorithm and analyzes the experimental results based on CEC2013 benchmark functions. The fourth part introduces DV-Hop node localization and applies the novel algorithm in the DV-Hop localization method. In the fifth section, the performance of localization in WSN is discussed in detail. Finally, the work of this paper is summarized.

2 Cat swarm optimization

The CSO is composed by n cats ($x_i, i = 1, 2, 3, \ldots, n$) with D-dimensions. The position (x_i=[x_{i1}, x_{i2}, x_{i3}, …, x_{in}]) of every cat is capable of being explained as a candidate solution according to fitness values respectively. That is to say, a cat with higher fitness values is more likely to be the best solution among those cats. Apart from position, there is also a parameter called velocity (v_i=[v_{i1}, v_{i2}, v_{i3}, …, v_{in}]) in each cat. The combination of seeking mode (SM) and tracing mode (TM) gives CSO the ability of getting an optimal solution continuously.

2.1 Seeking mode

The purpose of using the search model is to imitate the cat's seeking behavior during rest and alertness. This model is used to help cats capture the best prey. There are four main parameters in the seeking mode, including seeking memory pool (*SMP*), seeking range of the selected dimension (*SRD*), counts of dimension to change (*CDC*), and self-position considering (*SPC*). *SMP* is defined as the number of copies of each cat and stores the updated location. *SRD* represents the change rate of each dimension. *CDC* defines the number of dimensions in which variation occurs. Each dimension cannot exceed the scope defined by *SRD* in the process of mutation. *SPC* is a Boolean variable that determines whether the cat's standing position is one of the candidates for moving. *SPC* does not affect the location of *SMP*. The process of searching the optimal solution is summarized as follows:

1. Get $j = SMP$ copy according to cat's current location C_k. make $j \leq SMP \leq 1$ when the *SPC* is true. After that, return to the current position representing as a candidate solution.
2. For each replica, the previous value is randomly added or subtracted and replaced by the original position.
3. The fitness function values are calculated for all candidate solutions.
4. The selection probability of candidate solution will be set as 1 when the values of all fitness functions are equivalent, otherwise, it is obtained by the Eq. (4).

$$P_i = \frac{CF_i - CF_x}{CF_{max} - CF_{min}} \in [0, 1] \tag{1}$$

   where CF_i represents the i-th cat's fitness value.
5. Step 5: The current cats' position is replaced by randomly selected cats from the candidate solution. $CF_x = CF_{max}$ with the aim at finding the minimum solution calculated by fitness function, otherwise, $CF_x = CF_{min}$.

2.2 Tracing mode

There is another mode in the CSO algorithm called tracing mode (TM), the process of instructing the cats to catch the target involves the following steps:

1. Each dimension's rate is updated according to Eq. (2). The optimal position that gets the best fitness function and current position are represented by $C_{best,d}$, $C_{i,d}$ respectively. The acceleration coefficient c which is used to control the influence strength of $C_{best,d}$ is set as 1.05. r is a random value in [0, 1].

$$V_{i,d}^{t+1} = V_{i,d}^{t} + c \times r \times (C_{best,d}^{t} - C_{i,d}^{t}) \tag{2}$$

2. Check whether the velocity is within the maximum speed range. The updated velocity will be set to the maximum value(that is, the upper boundary) once it's beyond the maximum.
3. Use following Eq. (3) to update the cats new position.

$$C_{i,d}^{t+1} = C_{i,d}^{t} + V_{i,d}^{t} \tag{3}$$

2.3 The description of CSO algorithm

As we mentioned before, the search for the best position in CSO is dependent on the cooperation between seeking mode and tracing mode interactive. At first, the CSO randomly divides all cats into these two search modes. Secondly, the ratio among the variables is controlled by the variable mixing ratio (MR). Thirdly, a combination of the two search methods is used to search for the optimal position, the process of CSO is concluded below:

1. Initialize n cat's position and velocity within a limited area at random.
2. Calculate the fitness function according to the position of cats and select the cat with the best fitness as $C_{best,d}$ at the same time.
3. Randomly selects the $n \times MR$ cats and make them carry out seeking mode, and others execute tracing mode.
4. Evaluate the quality of every cat according to its search patterns and update $C_{best,d}$.
5. If the termination conditions, including the tendency of convergence and the maximum iteration, are not met, turn to steps 3–5, otherwise terminate.

3 Parallel and compact CSO

In this section, the novel CSO is divided into three parts. The first part describes the parallel strategy proposed in this article. The second part describes the compact solution. Thirdly, the parallel strategy mentioned in this article is combined with the compact scheme. Finally, the performance analysis of a novel CSO algorithm is described. To improve readability, Table 1 lists the main notation and their meanings.

3.1 Parallel communication strategy

The parallel strategy is an algorithm strategy to group particles and makes them calculate simultaneously. It can improve the accuracy and convergence of the evolutionary

Table 1 List of the main notation and their meanings

Notation	Definition
D	Dimension
n	The number of population groups
t	The current iteration number of the population
ω	The weight is denoted as 0.1 to 0.4
A	The weighted result of each group
FDP	The probability density function
PV	The perturbation vector
erf	The error function
μ	Mean value
σ	Standard deviation
FS	Population fitness function value
SMP	Seeking memory pool
SRD	Seeking range of the selected dimension
CDC	Counts of dimension to change
SPC	Self-position considering

algorithm. Therefore, this paper optimizes the parallel strategy of the algorithm by improving the three communication schemes. Suppose there are n groups of populations, denoted as $N = N_1, N_2, \ldots, N_n$, where n is a positive integer greater than 1. t is the current iteration number of the population. When the current population is iterated for the first time, there will be exchanges among the populations, where $I = I_1, 2I_1, 3I_1, \ldots$. Below we have three novel parallel strategies that are explained in detail.

3.1.1 Communication strategy with average replacement

When the number of iterations reached I-th, the cats in each group communicated and negotiated. This communication strategy uses the idea of average substitution to communicate. Because the optimal value of each group will be slightly different when each subpopulation reaches the ith communication, this communication means is to average the optimal value of each group. This is good for avoiding local optimality. The specific execution plan is to average the optimal value of N groups, and then use the final average to replace the worst value of each group of N groups. We call this communication strategy (Average Replacement) AR. Figure 1 shows the communication strategy process diagram of AR. In Fig. 1, the red triangles in each group represent the worst value in each group. The arrow pointing to the red triangle means that each group uses the final average value to replace the worst value in each group.

3.1.2 Communication strategy with best replacement

In CSO, when all cats complete seeking mode and tracing mode before the I-th Tracing, cats of each subgroup communicate and meet. Since the optimal value calculated by each subpopulation of each group cannot determine which group is better, the communication strategy chooses to randomly select the best value to replace the worst value in each subpopulation. We call this exchange strategy (Best Replacement) BR. Figure 2 shows the BR communication strategy process diagram. In Fig. 2, the red star and red triangle represent the best values in all populations and the worst values in each group. The arrow pointing from the red star to the red triangle means that the best value in the population is used to replace the worst value of each subpopulation.

3.1.3 Communication strategy with weight replacement

If the cats of each sub-population are constantly searching for the best in their own group's orbit, then the algorithm will easily fall into a local optimum. Although each subpopulation of each group is constantly searching for optimization, each subpopulation has no reasonable communication. If each subpopulation of cats is constantly searching for optimization in its group of tracks, then the algorithm can easily fall into local optimization. Although each subpopulation of each group is constantly searching for optimization, each subpopulation has no reasonable communication. Therefore, we use a weighted method to make good use of each population. Suppose the weight is ω, and ω is denoted as 0.1 to 0.4. Then, the weighted result of each group is denoted as A.

$$A = 0.1 \times N_1^b + 0.2 \times N_2^b + 0.3 \times N_3^b + 0.4 \times N_4^b \qquad (4)$$

where N_n^b is the est value for the n-th group and n is a positive integer. Then, A is used to replace a random number in each group, so that each group can be effectively utilized and local optimality can be avoided. We call this communication strategy (Weight Replacement) WR. Figure 3 illustrates WR's communication strategy. In Fig. 3, the colored triangles in each group represent the random value of each group. The pentagon, triangle, circle, and star at the bottom of all groups represent the weight of each group. The arrow points to the colored triangles to represent the random value in each group replaced by A.

3.2 Compact scheme

The compact scheme provides an efficient way to take advantage of variable memory savings because it adopts a community of solutions with probabilities to simulate the

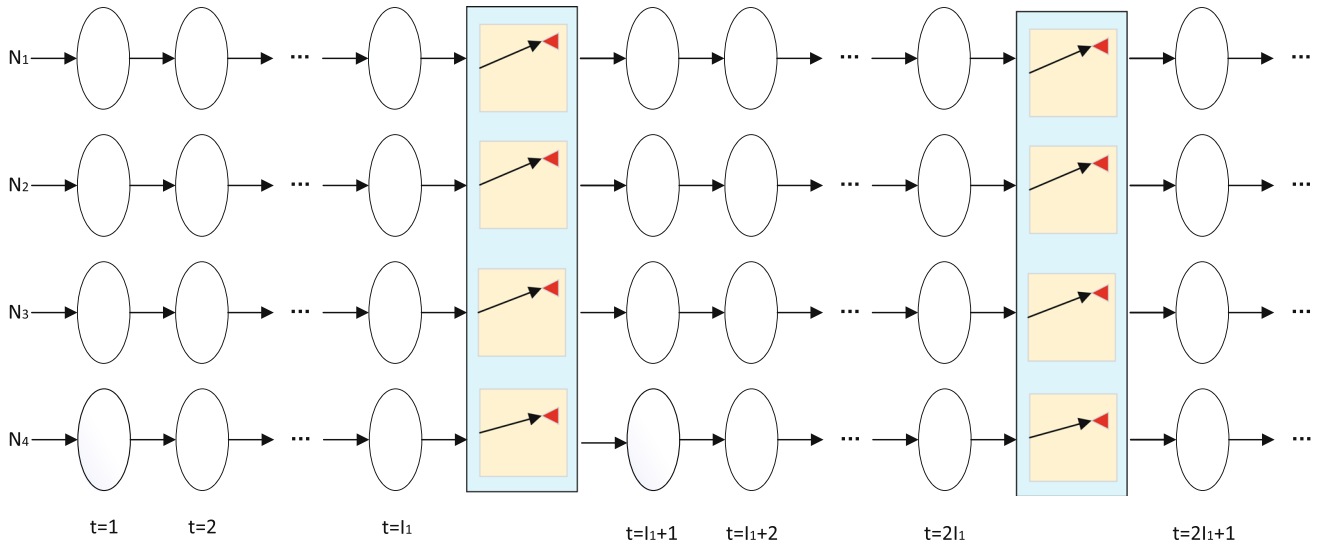


Fig. 1 The AR communication strategy

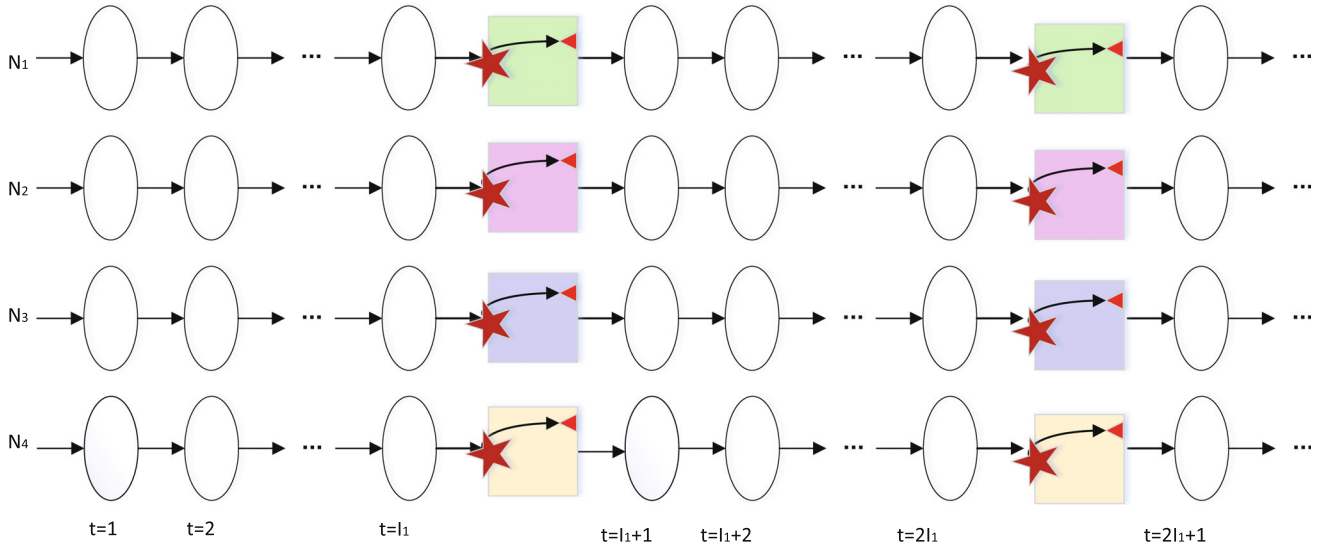


Fig. 2 The BR communication strategy

behavior based on population. This advantage is so important for some areas, such as WSN. To conveniently deploy sensor nodes, the size of sensor nodes is as possible as small. Therefore, the compute performance of the sensor nodes is so weak. It is impossible to run a population-based algorithm on a sensor node, but the compact algorithm does not require strong computing power to effectively solve the problem.

The compact algorithm uses probability representation to obtain smaller variable memory, instead of storing the solution of the whole population in the meta-heuristic algorithm. The specified probability of each element in the new candidate solution is maintained during the best process. In the compact scheme, a population comes from a probability density function (*FDP*) and can be viewed as a virtual population. In the data structure, the virtual population is encoded, which is called the perturbation vector (*PV*) to solve the probabilistic model. The *PV* is the vector of binary numbers in the binary-coded compact scheme, while in a real-valued compact scheme, the so-called perturbation vector is a matrix of $n \times 2$. *PV* probably generates the solution to be generated from the vector. The real value *PV* in the candidate solution can also be explained as the probability component within the unsolved question. After the update, *PV* will show a better solution. *PV* can be positioned as a matrix related to μ and σ, and the *PV* can be defined as:

$$PV_t = [\mu_t, \sigma_t]. \tag{5}$$

where in the interval $[-1, 1]$, the mean and standard

deviation of the Gaussian probability distribution function (*PDF*) truncated by each design variable are expressed as μ and σ. The value is in PDF for each design variable. To keep its area equal to 1, the amplitude of PDF is normalized. The vertex t in the Eq. (5) indicates generation (the number of times of comparison). Where t is time steps. To keep the algorithm stable, truncated by each design variable within the interval $[-1, 1]$. As the Gaussian probability is defined in $[-\infty, +\infty]$, so the area between -1 and 1 of *PDF* is normalized to 1. According to the expressed *PV*, corresponding to a virtual cat based on μ and σ is generate related Gaussian. *PDF* is represented by the following Eq. (6):

$$PDF_i(trunc(x)) = \frac{\sqrt{\frac{2}{\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}}{\sigma_i\left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} \tag{6}$$

where *erf* is the error function. $PDF_i(trunc(x))$ is the probability distribution of the PV associated with the μ and σ in the truncated Gaussian *PDF*. This is the value from *PDF* to variable x_i. By using the Chebyshev polynomial as described in *CDF*, *PDF* can be constructed in accordance with the cumulative distribution function, and then the corresponding cumulative remotion function is calculated by using the *PDF* in Eq. (6).

When a cat is doing a seeking mode, the first initializes the *PV*, μ and σ. The cat is copied according to the value of *SMP* in the initialization *PV*. Determine whether to copy the cat's current position based on whether the value of *SPC* is true and save the current position as a candidate position for the cat. Otherwise, $j = SMP$, the current position will not be a candidate position. For each candidate position of the cat, the *SRD* of the candidate position is randomly added or subtracted according to the *CDC*, and the old position is replaced with the new position. It calculates the fitness (*FS*) of all candidate locations and the likelihood that all candidate locations will be selected. If all FS_s are the same, the selectivity of all candidate locations is set to 1, otherwise, the selectivity of the candidate locations is calculated according to Eq. (7). The candidate position for the cat is randomly selected, and then the cat is moved to the selected position. Then, according to the current and best cat positions, these cats will proceed to the next stage. The better solution is determined by the competition between the winner and loser based on a fitness function. Then update *PV* based on its wins and losses. The update rule of the mean value μ of each element is:

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{n}(winner_i - loser_i) \tag{7}$$

where the scale of the virtual population is denoted by n. For σ values, the update strategy for each element is presented as follows:

$$(\sigma_i^{t+1})^2 = (\sigma_i^t)^2 + \mu_i^t - \mu_i^{t+1} + \frac{1}{n}(winner_i^2 - loser_i^2) \tag{8}$$

Finally, the *PV* is updated based on the new μ and σ.

3.3 Hibrid parallel and compact CSO

This section introduces the hybrid parallel and compact cat swarm algorithms. Firstly, the *PV* is initialized and the cat population is set to n subpopulation. Secondly, the compact scheme is implemented for the n subpopulation. Communication occurs when the population iterates to the I-th time. The above three communication strategies are combined with the compact scheme respectively, which are referred to as Parallel Compact CSO-Average Replacement (PCCSO-AR), Parallel Compact CSO-Best Replacement (PCCSO-BR) and Parallel Compact CSO-Weight Replacement (PCCSO-WR). Finally, the maximum iteration times of the population are checked. If the termination condition is met, the algorithm ends. Otherwise, go back to the second step to continue the algorithm. The pseudo-code of the PCCSO is shown in Algorithm 1. To better introduce the PCCSO algorithm proposed in this paper, we will analyze the theoretical computational complexity of PCCSO. As can be seen from Algorithm 1, the computational complexity of algorithm 4–14 lines is $O(g \times d)$, algorithm 15 line is O(1), and algorithm 16–33 lines is $O(g \times g)$. So the algorithm 4–33 lines computational complexity is $\max(O(g \times d), O(1), O(g \times g))=O(g \times d)$, std. $d > g$. The final 3–36 lines computational complexity is $O(time_{max} \times g \times d)$, that is, the PCCSO computational complexity is $O(time_{max} \times g \times d)$.

Algorithm 1 Parallel Compact Cat Swarm Optimization (PCCSO)

Require:
$F(x_1, x_2, x_3, ..., x_D)$: fitness function. $time_{max}$: the maximum time. g: the number of groups in parallel. d: dimension.
Ensure:
F_{best}: global optimization;
Initialization: each group is $N(i)$, $(i \leq g)$, $t = 1$, $c = 1.05$, PV, $N(i)_{gbest}$ and $N(i)_{fbest}$ of each group, $time = 0$;
Initialization: $GlobalBest$=$N(i)_{gbest}$; $GlobalFmin$ =$N(i)_{fbest}$;
while $time < time_{max}$ **do**
  for $i = 1$ *to* g **do**
    Generate x_1, x_2 via PV;
    $[winner, loser, fit_{winner}]$=compete($x_1$, x_2);
    for $i = 1$ *to* d **do**
      Update μ, σ via Eq. (7) and Eq. (8);
      if $fit_{winner} < N(i)_{fbest}$ **then**
        $N(i)_{gbest} = winner$;
        $N(i)_{fbest} = fit_{winner}$;
      end if
    end for
  end for
  Update $GlobalBest$ and $GlobalFmin$;
  for $i = 1$ *to* g **do**
    /*PCCSO-AR*/
    for $i = 1$ *to* g **do**
      replace the $worst(N(i))$ with average value;
      Disturb and update $N(i)_{gbest}$;
    end for
    /*PCCSO-BR*/
    Randomly select a group of best values $N(k)$;
    for $i = 1$ *to* g **do**
      replace the $worst(N(i))$ with $N(k)$;
      Disturb and update $N(i)_{gbest}$;
    end for
    /*PCCSO-WR*/
    for $i = 1$ *to* g **do**
      replace the $worst(N(i))$ with A of Eq. (4);
      Disturb and update $N(i)_{gbest}$;
    end for
  end for
  $time+ = 1$;
end while
Output: F_{best}

3.4 PCCSO experimental results and analysis

In this section, CEC2013 is considered classic and is used to demonstrate the performance of PCCSO. It has 28 benchmark functions. The 28 classical reference functions are distributed from unimodal functions to multimodal functions and combined functions. These three kinds of functions are considered to cover most practical problems and can effectively test the feasibility of the proposed algorithm. In order to ensure the fairness of the experiment, this paper uses the same computer for experimental training. The development environment of the computer is MatLab2015b and the processor is Intel(R) Core(TM) I7-4720HQ CPU @ 2.60ghz.

3.4.1 Parameter settings

To test the performance of the proposed PCCSO algorithm, we compare it with the classical PSO [3] and the original algorithm CSO [12] to prove that the proposed algorithm is feasible. Because the proposed algorithm uses the idea of parallel and compact, this paper compares it with parallel CSO (PCSO) and compact CSO (CCSO) respectively to verify the performance of the proposed algorithm. To test the performance of PCCSO and other algorithms fairly, the dimension (D) is set to 10, and each algorithm runs 10 times in each test function, and the maximum running time ($time_{max}$) of each algorithm is set to 25 s. Population size is one of the key factors that determine the performance of an algorithm, so we set the population number population of PSO, CSO and PCSO to 160. The key of compact scheme is to reduce the number of population, so as to improve the time efficiency of the algorithm. The traditional idea of compact scheme is to use a particle for optimization training, while the cat swarm algorithm has two behaviors. The population of the algorithm determines whether the algorithm can find the optimal solution effectively. Therefore, according to the experience, the population number (*Population*) of CCSO is set to 16. PCCSO is trained in groups, so the population number (*Population*) of PCCSO-AR, PCCSO-BR and PCCSO-WR is set to 32. Based on past experience and continuous attempts, this paper finds that the group (*Groups*) set as 4 can better and faster search for optimization. In order to ensure the fairness of the experiment, the basic parameters of CSO, PCSO, CCSO, PCCSO-AR, PCCSO-BR and PCCSO-WR are set as c=1.05, *SMP*=40, *SRD*=0.8, *CDC*=0.8, *SPC*=0.2, *MR*=0.8, and the parameters of PSO should be set as C=2.0, W=0.9 according to the traditional method. The parameter Settings of all algorithms are shown in Table 2.

3.4.2 PCCSO performance analysis

This section mainly analyzes and analyzes the experimental results of three PCCSO algorithms proposed in this paper. Table 3 shows the results of averaging 28 benchmark functions run independently 10 times. In the case of the same time, the average value of the same times of each function is used for functional evaluation. This not only ensures the fairness of the experiment, but also measures the performance of the proposed algorithm in training speed. All the algorithms are placed in the same table so that it is clear how the algorithms compare. Where, bold indicates the algorithm that performs best in the current test function. According to the data given in Table 3, it can be clearly seen which algorithm has more advantages in the optimization process.

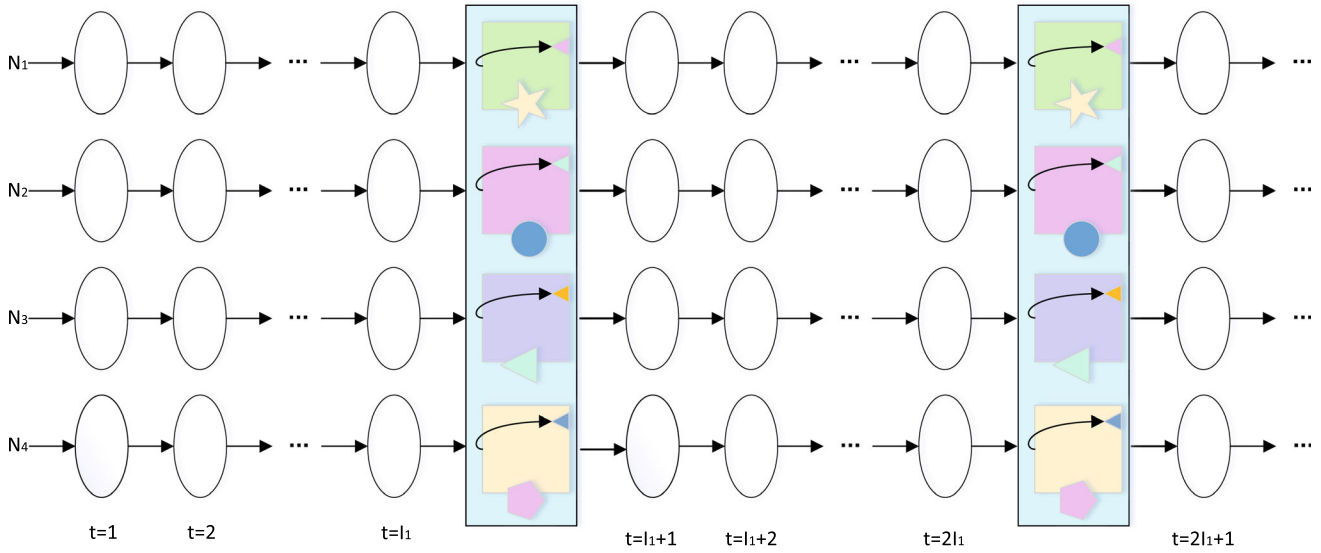


Fig. 3 The WR communication strategy

Table 2 Parameter setting

Algorithm	Parameter
PSO	$c = 2.0$, $w = 0.9$, $Population = 160$, $D = 10$, $time_{max} = 25$s
CSO	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 160$, $D = 10$, $time_{max} = 25$s
CCSO	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 16$, $D = 10$, $time_{max} = 25$s
PCSO	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 160$, $Groups = 4$, $D = 10$, $time_{max} = 25$s
PCCSO-AR	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 32$, $Groups = 4$, $D = 10$, $time_{max} = 25$s
PCCSO-BR	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 32$, $Groups = 4$, $D = 10$, $time_{max} = 25$s
PCCSO-WR	$c = 1.05$, $SMP = 40$, $SRD = 0.8$, $CDC = 0.8$, $SPC = 0.2$, $MR = 0.8$, $Population = 32$, $Groups = 4$, $D = 10$, $time_{max} = 25$s

To test the convergence of the proposed PCCSO algorithm, this paper uses Figs. 4, 5 and 6 to demonstrate the convergence performance and speed of the comparison algorithm used in 28 benchmark functions. The horizontal axis represents the time that each function takes to run. The vertical axis is a function of the current time. This paper uses a variety of test functions to test the performance of the PCCSO algorithm. The selected test functions include unimodal, multi-modal, and composition functions. The f1–f6 represents unimodal function, which can effectively test the convergence rate of the algorithm. The f7–f20 is a multi-modal function, which can test whether the algorithm can effectively avoid the local optimal and find the global optimal solution when the test function has multiple extreme values. The f21–f27 represents the composition function, which can test the algorithm's ability to deal with complex problems. Through these three functions can effectively test PCCSO optimization ability, convergence speed and robustness. To explain the performance of the proposed algorithm in detail, these three functions are divided into the following three parts for detailed explanation.

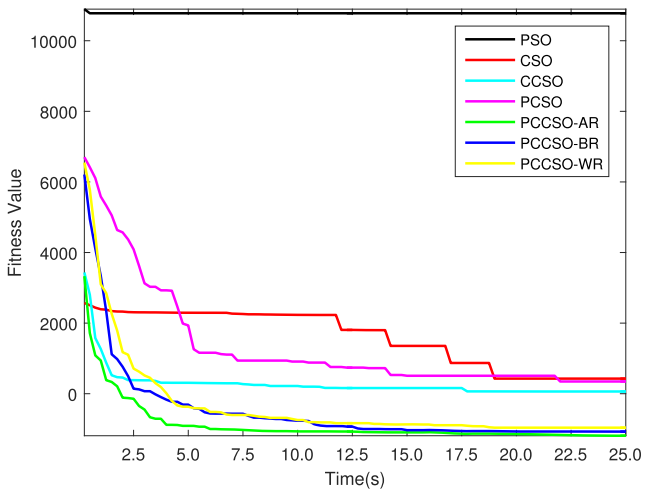
(1) Unimodal Functions: The f1–f6 of Table 3 and Fig. 4 illustrates the benchmark results of each algorithm using the unimodal function. Only one global optimum is a typical feature of unimodal functions. By analyzing the data in Fig. 4, it can be concluded that PSO and CSO are easy to fall into local optimality. For example, in f4, when the running time reaches 5 s, both PSO and CSO basically reach a stable state. However, the proposed algorithm PCCSO has always been in a state of continuous optimization. It can be seen from f4 in Table 3 that the best final optimization result is PCCSO-AR. Its result is '2.2514E+03', a little more than 10 times more than PSO, CSO, CCSO and PCSO. By observing f1–f6 in Table 3, we can see that the first six functions PCCSO-AR have shown

Table 3 The results of simulation experiment

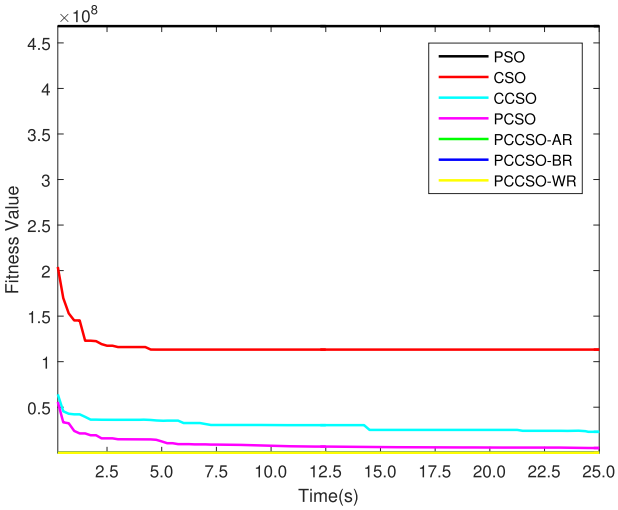
F	PSO	CSO	CCSO	PCSO	PCCSO-AR	PCCSO-BR	PCCSO-WR
f1	1.0778E+04	2.2853E+03	2.6193E+01	3.0908E+02	− **1.2972E+03**	− 1.1995E+03	− 1.1058E+03
f2	4.6842E+08	1.1325E+08	2.2968E+07	5.1958E+06	**1.0778E+04**	**1.0778E+04**	**1.0778E+04**
f3	8.2884E+14	8.2884E+14	5.5424E+12	1.5632E+10	**3.7137E+06**	5.9654E+06	1.0290E+07
f4	1.2754E+04	1.2100E+04	1.2450E+04	1.1246E+04	**2.2514E+03**	3.0668E+03	6.1460E+03
f5	2.6804E+04	2.1654E+03	− 1.6675E+02	4.7029E+02	− **9.3858E+02**	− 8.9640E+02	− 8.4480E+02
f6	− 6.3498E+01	− 4.0334E+02	− 7.7226E+02	− 7.7949E+02	− 8.4940E+02	− 8.3084E+02	− **8.5375E+02**
f7	4.7285E+04	2.2823E+03	− 5.3127E+02	− 6.1259E+02	− 7.6110E+02	− 7.4709E+02	− **7.1568E+02**
f8	− 6.7969E+02	− 6.7948E+02	− 6.7955E+02	− 6.7965E+02	− 6.7972E+02	− 6.7956E+02	− 6.7960E+02
f9	− 5.8901E+02	− 5.8649E+02	− 5.9131E+02	− 5.8998E+02	− **5.9271E+02**	− 5.9232E+02	− 5.9051E+02
f10	1.2457E+03	8.7538E+02	− 2.0395E+02	− 2.8220E+02	− **4.8461E+02**	− 4.6701E+02	− 4.5276E+02
f11	− 1.6988E+02	− 2.1053E+02	− 3.0393E+02	− 2.8112E+02	− **3.3874E+02**	− 3.4658E+02	− 3.3604E+02
f12	− 9.7270E+01	− 1.3862E+02	− 1.3573E+02	− 1.9630E+02	− **2.4567E+02**	− 2.3239E+02	− 2.1671E+02
f13	− 2.9670E-01	− 3.8522E+01	− 1.0054E+02	− 1.0680E+02	− **1.5369E+02**	− 1.4077E+02	− 1.3528E+02
f14	1.8877E+03	1.3867E+03	1.4023E+03	1.6975E+03	**1.2466E+03**	1.3986E+03	1.3140E+03
f15	1.7917E+03	1.4395E+03	1.2117E+03	1.4521E+03	1.2029E+03	**1.0715E+03**	1.1578E+03
f16	2.0030E+02	2.0169E+02	2.0098E+02	2.0124E+02	2.0076E+02	2.0135E+02	2.0130E+02
f17	4.0881E+02	4.5545E+02	3.8903E+02	4.0494E+02	3.6414E+02	**3.6362E+02**	3.6995E+02
f18	5.0660E+02	5.6772E+02	4.8925E+02	5.0314E+02	**4.5911E+02**	4.7504E+02	4.7415E+02
f19	3.0452E+04	2.1730E+03	5.6184E+02	8.8657E+02	5.0741E+02	5.1240E+02	**4.1152E+02**
f20	6.0500E+02	6.0500E+02	6.0429E+02	6.0458E+02	**6.0380E+02**	6.0421E+02	**6.0398E+02**
f21	1.3417E+03	1.2929E+03	1.1494E+03	1.1428E+03	**1.1019E+03**	1.1026E+03	1.1051E+03
f22	3.5710E+03	3.4739E+03	3.0015E+03	2.6086E+03	2.5370E+03	2.7327E+03	**2.2586E+03**
f23	3.0264E+03	2.9341E+03	2.8858E+03	2.7946E+03	**2.2218E+03**	2.4087E+03	2.7487E+03
f24	1.2850E+03	1.3554E+03	1.2269E+03	1.2318E+03	**1.2254E+03**	1.2279E+03	1.2271E+03
f25	1.3762E+03	1.3879E+03	1.3278E+03	1.3313E+03	1.3071E+03	1.3118E+03	**1.0520E+03**
f26	3.9848E+03	2.2908E+03	1.4475E+03	1.5095E+03	1.4511E+03	1.4772E+03	**1.4430E+03**
f27	2.0518E+03	2.1118E+03	1.7587E+03	1.7854E+03	**1.7144E+03**	1.7804E+03	1.7286E+03
f28	2.4609E+03	2.2534E+03	2.2629E+03	2.3921E+03	**2.0361E+03**	2.1266E+03	2.1306E+03

excellent optimization capability. This proves that using the average value to replace each set of worst values combined with the compression scheme has a very good functional effect in solving unimodal functions. As can be seen from f2 in Table 3, the optimization capabilities of the three PCCSO algorithms are almost similar. However, these three algorithms show sufficient advantages compared with PSO, CSO, CCSO and PCSO. According to the above analysis can fully prove that PCCSO has a better ability to solve the unimodal function, can quickly obtain the global optimal value and prevent the search into the local optimal value.

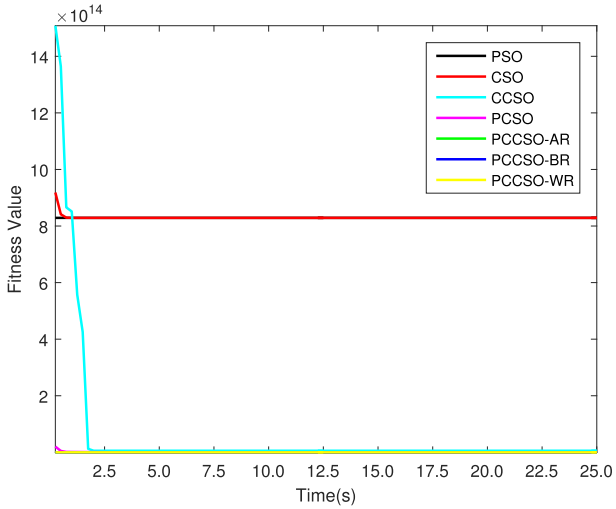
(2) Multi-modal Functions: The f7–f20 of Table 3 and Fig. 5 illustrates the benchmark results of each algorithm using the multi-modal function. Many local optima are the salient features of multimodal functions. As can be seen from Fig. 5, it is difficult for PSO to avoid falling into local optimality. It can be seen from the functions f7, f9, f10, f11, f12, f13, f19 and f20 that PSO is unable to jump out after falling into a certain extreme point. However, the three PCCSO algorithms show excellent ability to jump out of local optimal. In f16, PSO showed a good effect. But it is also clear that although the PSO final search results are good, the PSO almost reaches an optimal state after 15 s, and PCCSO is in continuous optimization. This fully demonstrates that PCCSO can effectively avoid falling into locally optimal defects. In f9, f14, and f22, the PCCSO test results are not as good as some of the other comparison algorithms. For example, in f9, the PCCSO-WR is less effective than the CCSO. This may be due to the occasional limitation of the PCCSO-WR in assigning weights. It can be seen from f7–f20 in Table 3 and Fig. 5 that the algorithm proposed in this paper has a good effect in solving most functions. In f7, f10, f11, f12, f13, f15, f17, f18, f19 and f20, PCCSO shows excellent performance in optimization. This illustrates that in most cases, PCCSO works



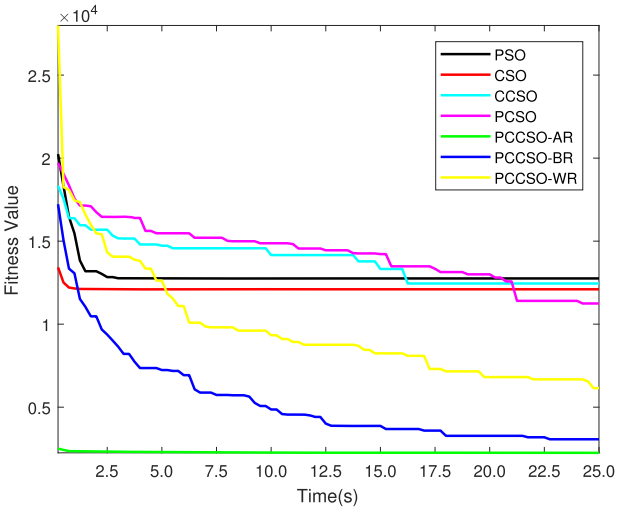
(a) f_1.



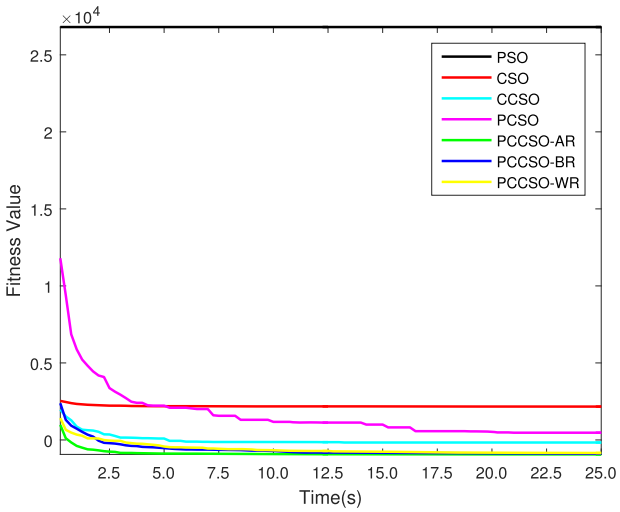
(b) f_2.



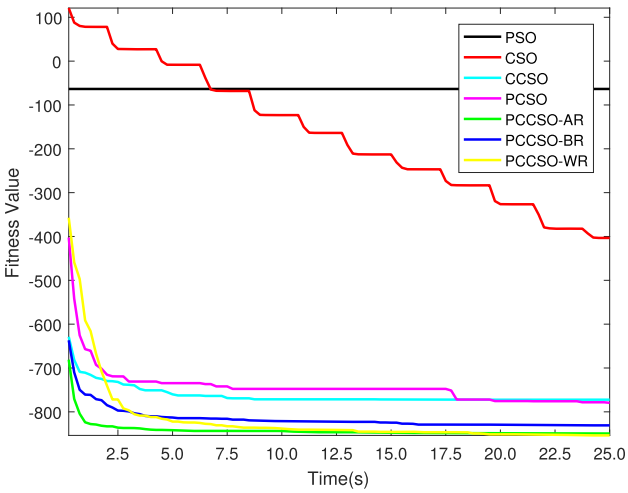
(c) f_3.



(d) f_4.



(e) f_5.



(f) f_6.

◀ **Fig. 4** Convergence tendency for unimodal benchmark functions with 10D optimization

well within a multi-modal function. This proves that PCCSO has a good performance in solving multi-modal functions problems, and can effectively solve such problems.

(3) Composition Functions: The f21–f28 of Table 3 and Fig. 6 illustrates the benchmark results of each algorithm using composition functions. As Fig. 6 shows, the experimental results of PCCSO-BR in f32 are slightly worse than those of PCSO, but PCCSO-AR and PCCSO-WR show better results. This is because PCCSO-BR uses the best worth thinking in the solution of the combined function problem has a certain limitation, is likely to make the algorithm into local optimal. As can be seen from f21–f28 in Table 3, PCCSO-AR and PCCSO-WR show excellent optimization results. In f21, f23, f24, f27, and f28, PCCSO-AR optimization results are more advantageous. In f22, f25, and f26, the winner is PCCSO-WR. This is because PCCSO-AR uses the idea of averaging and compression to effectively average the gap in the combined function problem. PCCSO-WR uses the idea of weight and compression to balance the power of combined functions. The PCCSO-BR, though slightly less effective, also performs better than most other algorithms. This suggests that using the PCCSO-AR and PCCSO-WR proposed in this article is a better way to solve the problem of combined functions.

From Table 3 and Figs. 4, 5 and 6, we conclude that the proposed algorithm improves the global search capability and quickly converges during the iteration. Therefore, PCCSO has greatly improved convergence speed and accuracy. To better highlight the memory advantages of the algorithm proposed, this paper analyzes the memory situation of PCCSO and traditional CSO. The comparison of CSO and PCCSO about memory cost is shown in Table 4. The number of variables of CSO and PCCSO algorithms can be obtained from their update equations. In Table 4, we can see that the update equations of CGBMO consist of Eqs. (1–3) and (13–16). Suppose the number of packets is m, and m is a positive integer. The population size of PCCSO is m. Therefore the computing complexity is $7 \times m \times T \times$ Iterations. In the same way, we can obtain the computing complexity of CSO is $3 \times N \times T \times$ Iterations. Although the update equations of PCCSO are greater than CSO, it has only m individual and m is much less than N. Therefore it costs less memory than the original algorithm.

In summary, the performance of PCCSO is better than PSO, CSO, CCSO, and PCSO in simple unimodal, multimodal, and composition functions. The PCCSO can jump out of the local optimum and surpass other functions. Most experimental results show that PCCSO has a good ability to converge and solve problems. Therefore, the PCCSO algorithm proposed in this paper can not only effectively solve different kinds of function problems, but also improve the convergence speed of the algorithm and has a strong convergence ability. Moreover, in terms of memory usage and time complexity, PCCSO can also save the use of memory while the time complexity is relatively small.

4 Parallel compact CSO for DV-Hop

4.1 The node localization of DV-Hop

There are three main stages in the DV-Hop method. The main processes are as follows:

1. Obtain the number of hops between anchor nodes according to the broadcast of anchor node information through network flooding.
2. Estimate the distance between anchor nodes and unknown nodes using the average hop distance. After estimation, the hops and coordinate of anchor node will be delivered to other anchor nodes, and the average hops calculated by Eq. (9) between anchor, the node is got eventually.

$$Hopsize_i = \frac{\sum_{i \neq j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum_{i \neq j} h_{ij}} \tag{9}$$

   where $Hopesize_i$ represents the average distance of hops occurred between anchor nodes. The h_{ij} and (x_1, y_1), (x_2, y_2) represent the minimum quantity of hops and coordinates of node i and j respectively.
3. Calculate the average distance between hops which are broadcast to network based on anchor nodes.

The Eq. (10) the list below is responsible for calculating the estimated distance from anchor node i to unknown node u, in which the hop_{iu} represents the number of hops. Besides, $Hopsize_i$ represents the same meaning mentioned before.

$$D_{iu} = Hopsize_i \times hop_{iu} \tag{10}$$

The unknown node location will get determined by the maximum likelihood estimation method, once the calculation of the distance from anchor nodes to unknown nodes completes. In other words, the unknown node utilizes the information of n anchor nodes to find the coordinate parameters of the unknown node.

Suppose the location of anchor nodes $B_1, B_2, \ldots, B_n$ are (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) respectively. The unknown

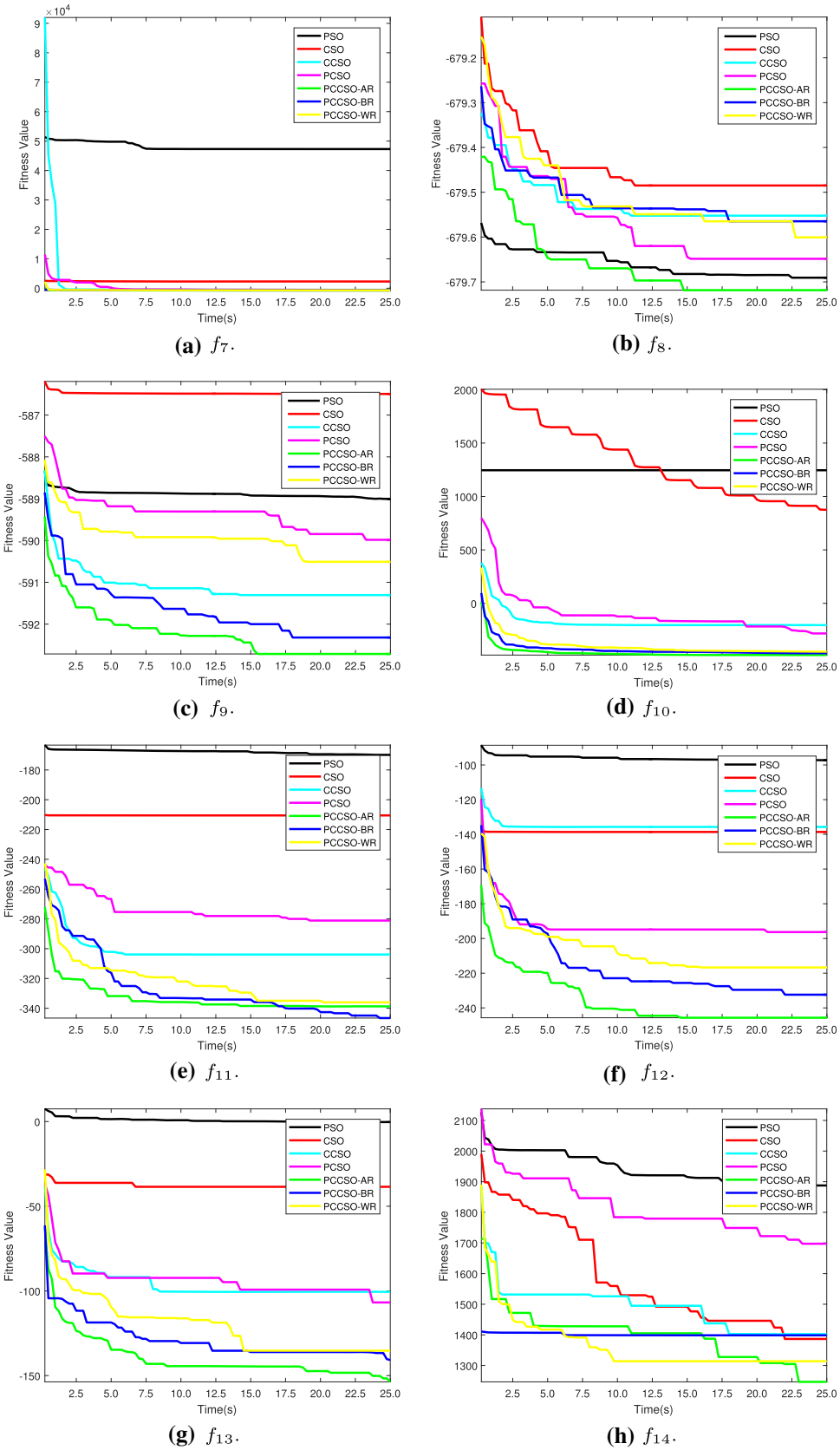
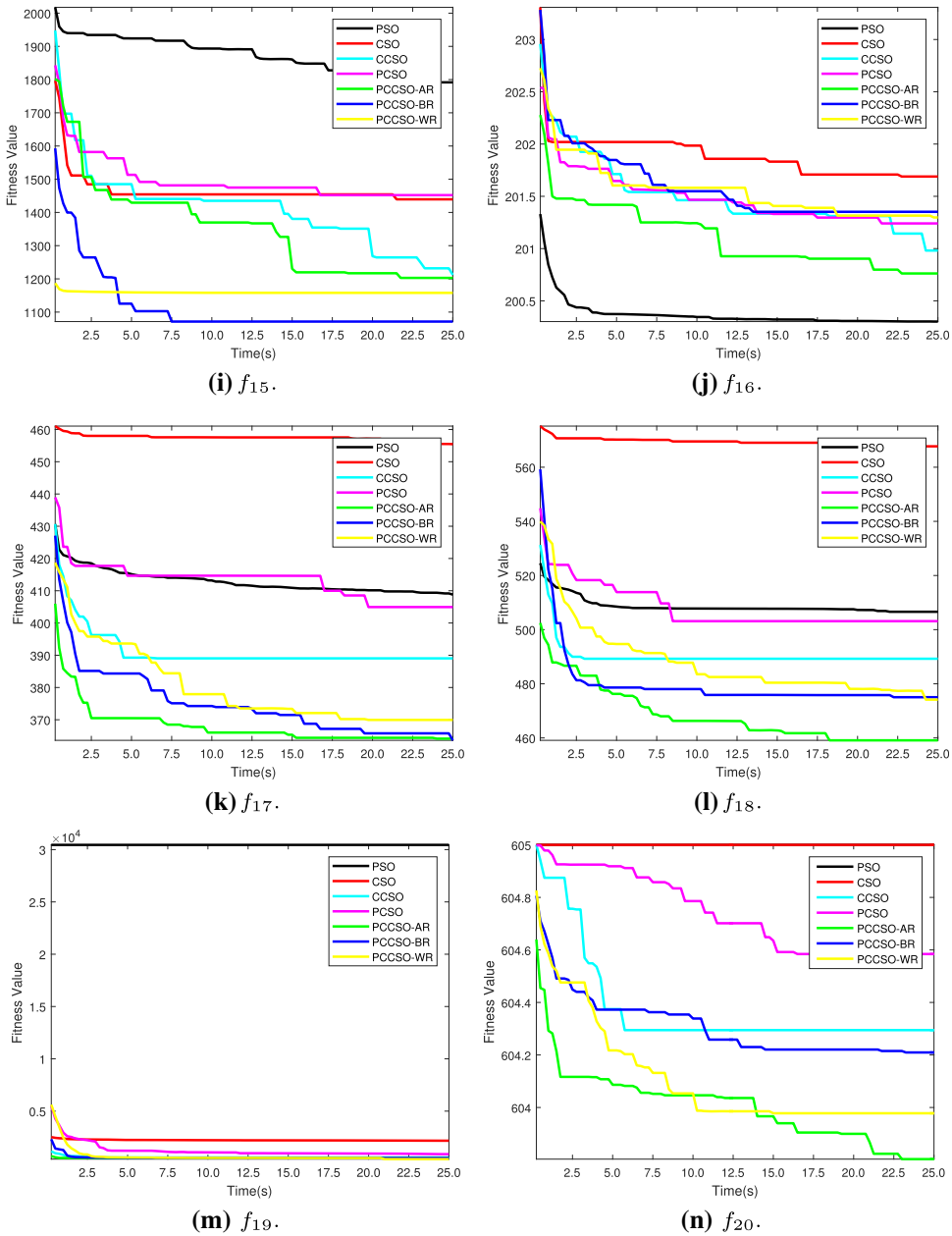


Fig. 5 Convergence tendency for multi-modal benchmark functions with 10D optimization

Fig. 5 continued



(i) f_{15}. **(j)** f_{16}.

(k) f_{17}. **(l)** f_{18}.

(m) f_{19}. **(n)** f_{20}.

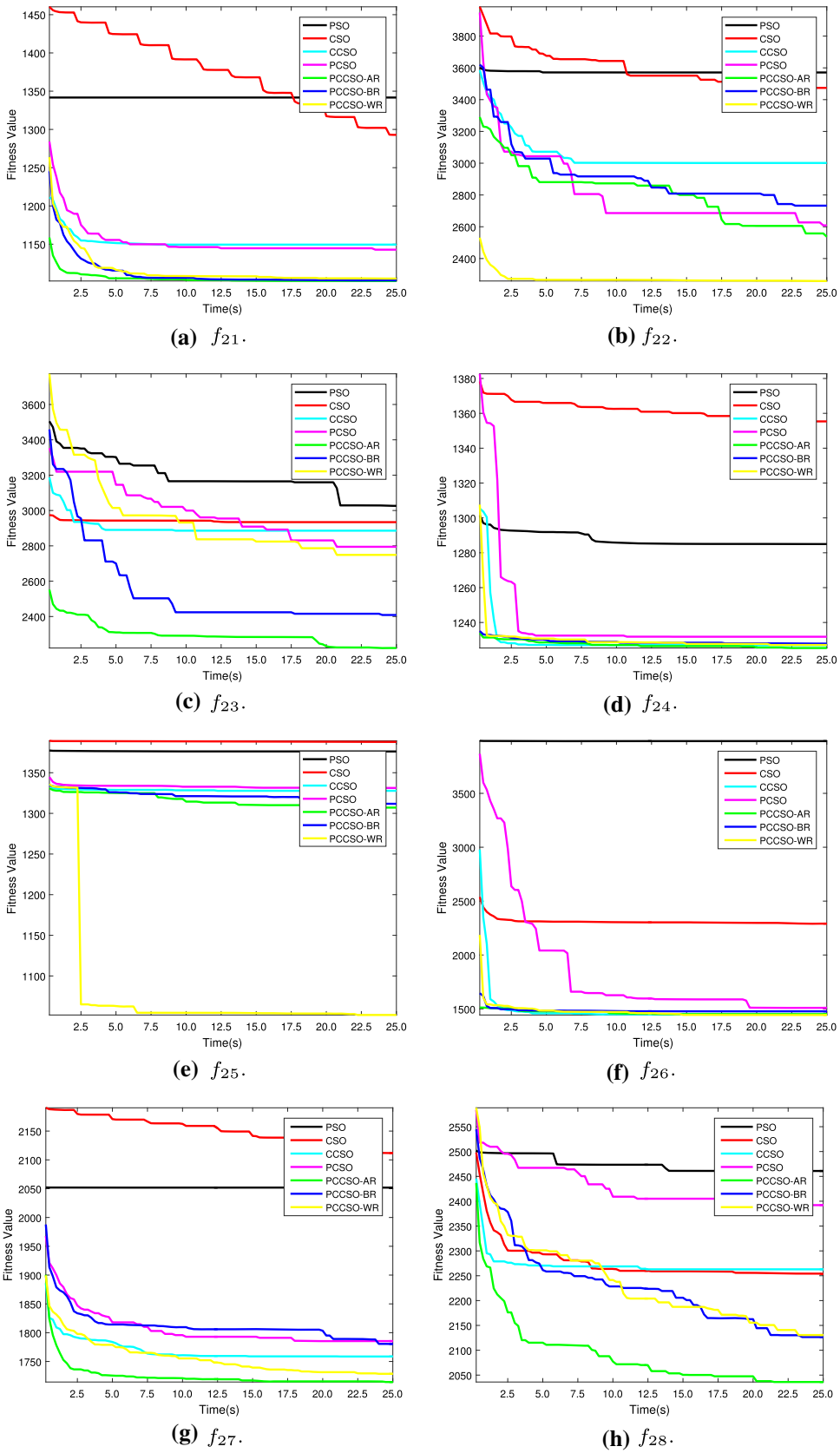
nodes' coordinate S_0 is set as (x, y), then the formula is as follows:

$$\begin{cases} \sqrt{(x-x_1)^2+(y-y_1)^2} = (d_{B_1S_0}) \\ \sqrt{(x-x_2)^2+(y-y_2)^2} = (d_{B_2S_0}) \\ \vdots \\ \sqrt{(x-x_n)^2+(y-y_n)^2} = (d_{B_nS_0}) \end{cases} \tag{11}$$

Subtracting the *n-th* formula from the previous $n-1$ formulas, the formula is as follows:

$$\begin{cases} x_1^2 - x_n^2 - 2(x_1-x_n)x + y_1^2 - y_n^2 - \\ 2(y_1-y_n)y = (d_{B_1S_0})^2 - (d_{B_nS_0})^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 - 2(x_{n-1}-x_n)x + y_{n-1}^2 - y_n^2 - \\ 2(y_{n-1}-y_n)y = (d_{B_{n-1}S_0})^2 - (d_{B_nS_0})^2 \end{cases} \tag{12}$$

The above formula can also be decomposed into a matrix multiplication form of $AX = B$, and the A, B and X are shown as follows:



(a) f_{21}. **(b)** f_{22}. **(c)** f_{23}. **(d)** f_{24}. **(e)** f_{25}. **(f)** f_{26}. **(g)** f_{27}. **(h)** f_{28}.

Fig. 6 Convergence tendency for composition benchmark functions with 10D optimization

Table 4 The space complexity of the two algorithms

Algorithm	Particle	Memory size	Computing complexity	Use equations
PCCSO	m	7 × m	7 × m × T × Iteration	(1)(2)(3)(13)(14)(15)(16)
CSO	N	3 × N	3 × T × N × Iteration	(1)(2)(3)

$$A = \begin{pmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{pmatrix} \tag{13}$$

$$B = \begin{pmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + d_{B_n S_0}^2 - d_{B_1 S_0}^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + d_{B_n S_0}^2 - d_{B_{n-1} S_0}^2 \end{pmatrix} \tag{14}$$

$$X = \begin{pmatrix} x \\ y \end{pmatrix} \tag{15}$$

The coordinates of the unknown node S_0 is the solution of (x, y)

Finally, in order to get the equation solved, the least square method is presented like Eq. (16)

$$X = (A^T A)^{-1} A^T B. \tag{16}$$

4.2 PCCSO application in DV-Hop

Many reasons affect the localization accuracy ratio of WSN, but the main reasons are the asynchronous non-uniform distribution and the factors of the DV-Hop localization algorithm itself. Here we analyze from the following perspective. In WSN, anchor nodes with exact locations cannot calculate accurate information about unknown nodes. In the traditional DV-Hop algorithm, the least square method is used to calculate the location information of unknown nodes. Since the least squares method itself has calculation errors, the unknown position iteration will be affected by the cumulative error, resulting in lower positioning accuracy of the traditional DV-Hop. The evolutionary algorithm does not need a mathematical model when searching for the optimal solution, which can effectively solve the DV-Hop positioning problem. Therefore, this paper uses the PCCSO algorithm to calculate the position of the unknown node to reduce the DV-Hop positioning error. There is a certain error in the distance between the unknown node and the anchor node, which is calculated by Eq. (17):

$$error_j = \sum_{i=1}^{m} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_{ij} \tag{17}$$

where m is the number of anchor nodes. (x_j, y_j) represents the coordinates of the j-th unknown node. (x_i, y_i) represents the coordinates of the anchor node. d_{ij} is the distance from the anchor node of i-th to the unknown node of j-th estimated by DV-Hop. $error_j$ is the sum of the distance errors from all anchor nodes to the j-th unknown node. In WSN, the greater the number of hops, the greater the positioning error, so the objective function of PCCSO uses the inverse of that number of hops to calculate in order to reduce the error [44]. In the objective function, in order to find the optimal solution more clearly, square the reciprocal of the number of hops and $error_j$. Minimizing the error equation makes the PCCSO the smallest error in the optimization process. Finally, in order to improve the positioning accuracy, we define the PCCSO fitness function as Eq. (18).

$$f(x, y) = min\left(\sum_{i=1}^{m}\left(\frac{1}{hop_{iu}}\right)^2 \times \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_{iu}\right)^2\right) \tag{18}$$

where $m(m \geq 3)$ is the number of anchor nodes. The coordinates of an unknown node optimized by the heuristic algorithms is represented by (x, y) and the (x_i, y_i) represents the anchor node coordinate. The distance calculated by DV-Hop from anchor node i to unknown node u is denoted by d_{iu}.

5 Experiment results on node localization in WSN using PCCSO

In this section, we apply PCCSO-AR, PCCSO-BR, and PCCSO-WR to DV-Hop and perform simulations, denoted as PCCSO-AR-DV-Hop, PCCSO-BR-DV-Hop, and PCCSO-WR-DV-Hop respectively. We compare the simulation results with the DV-Hop optimized by maximum likelihood estimation (DV-Hop) and compare the experimental results based on PSO (PSO-DV-Hop) and CSO (CSO-DV-Hop). And, to better illustrate the effectiveness of our proposed algorithm, we also compare with the DV-Hop optimized by CCSO and PCSO, which are recorded as CCSO-DV-Hop and PCSO-DV-Hop, respectively. The result of all experiments is an average of 30 runs to achieve a fair solution. The sensor nodes in WSN are randomly

generated in a two-dimensional fixed area. This experiment is done under the condition that the sensor area is 100 m × 100 m.

To more accurately prove that the scheme proposed in this paper is feasible and effective, we use three different forms of schemes to test the experimental results. First, the sensor nodes and anchor nodes are tested with a 10% scheme, that is, the communication radius is fixed at 40 m, the total number of sensor nodes is 150–400, and the corresponding anchor node is 15–40 m. Second, the total number of sensor nodes and the communication radius are fixed, and the number of anchor nodes is changed, that is, the communication radius is fixed to 40 m, the total number of sensor nodes is 200, and the corresponding anchor node is 15–40 m. Third, change the communication radius under the condition that the total number of sensor nodes and anchor nodes do not change, that is, the total number of sensor nodes and anchor nodes are 200 and 20, respectively, and the communication radius is 15–40 m. To better compare the performance-optimized by an evolutionary algorithm, we set the number of solutions to 8000. PSO-DV-Hop, the population number of CSO-DV-Hop and PCSO-DV-Hop is 80, and the number of iterations is 100. Because the compact solution is to reduce the memory, the population number of CCSO is set to 8, and the number of iterations is set to 1000. However, the three PCCSOs we proposed not only add the compact solution, but also the parallel idea, so the population numbers of PCCSO-AR-DV-Hop, PCCSO-BR-DV-Hop and PCCSO-WR-DV-Hop are all set to 32, and the number of iterations is set to 250. For a better description, we record the number of populations as p and the number of iterations as *Iter*, the number of groups as g, and the above parameter settings are shown in Table 5.

The *Le-Avg* which means average localization error is regard as performance metric and given as follows:

$$Le - Avg. = \frac{\sum_{i=1}^{n} \sqrt{(X - x_i)^2 + (Y - y_i)^2}}{n \times R} \tag{19}$$

There are n unknow nodes and (x_i, y_i) means the estimated coordinates of i-th unknown nodes, (X, Y) is the actually coordinates. All nodes communicate at the same distance and R represents this distance.

5.1 Train the anchor nodes by percentage

To ensure the versatility of the experiments, we distribute 10 percent of overall nodes in the 100 m × 100 m sensing area at random as anchor nodes and the number of overall nodes gradually increases from 150 to 400. Figure 7 and Table 6 show the optimization results of all localization methods under different communication distances. In Fig. 7 and Table 6, comparing with the DV-Hop, PSO-DV-Hop, CCSO-DV-Hop, and PCSO-DV-Hop, the proposed algorithm always gets the lowest average localization error that comes up to 0.2229 among the three communication strategies, and the improvements are 0.0812, 0.0546, 0.0459, 0.0295 and 0.0337 respectively. The improvement attributes to the adoption of communication between groups, which gets the most use of local optimal cats in each group. Furthermore, the error is getting slighter and smaller as the anchor nodes increase in most cases, except the number of anchor nodes is 35.

5.2 Anchor node change training

To further illustrate the localization efficiency using different algorithms comprehensively, the number of nodes is fixed as 200 and the range of communication that occurred between sensor nodes is 40 m. Figure 8 and Table 7 denote the experimental results on anchor nodes that increase from 10 to 40 gradually.

Seeing from Table 7, the PCCSO-WR-DV-Hop gets the lowest localization error that reaches 0.2106, and the decrease of average localization error ranges from 0.0375 to 0.0852 compared with others. Intuitively, the improvement of localization error is more obvious from Fig. 8 with different percentages of anchor nodes, and the larger the number of anchor nodes is, the smaller error is. The node error starts to converge when the number of anchor nodes exceeds 20.

Table 5 Experimental settings for parameters

Simulation parameters	Parameter
Sensing region area	100 m × 100 m
Total number of sensor nodes	150–400
The number of anchor nodes	15–40
Communication range	15–40 m
Number of solution test	8000
PSO-DV-Hop	$p = 80$, $Iter = 100$
CSO-DV-Hop	$p = 80$, $Iter = 100$
CCSO-DV-Hop	$p = 8$, $Iter = 1000$
PCSO-DV-Hop	$p = 80$, $Iter = 100$, $g = 4$
PCCSO-AR-DV-Hop	$p = 32$, $Iter = 250$, $g = 4$
PCCSO-BR-DV-Hop	$p = 32$, $Iter = 250$, $g = 4$
PCCSO-WR-DV-Hop	$p = 32$, $Iter = 250$, $g = 4$

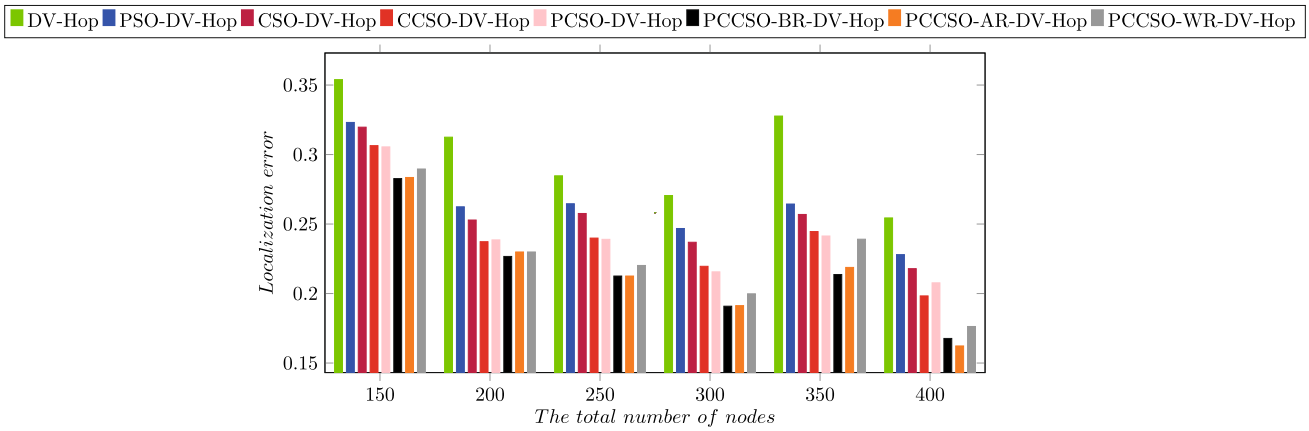


Fig. 7 Localization error results of sensor nodes and anchor nodes trained in percentage

Table 6 Experimental results using applied PCCSO and other algorithms by different number of overall nodes

Algorithm	150	200	250	300	350	400	Avg.
DV-Hop	0.3539	0.3125	0.2847	0.2705	0.3277	0.2544	0.3041
PSO-DV-Hop	0.3231	0.2624	0.2646	0.2468	0.2644	0.2280	0.2755
CSO-DV-Hop	0.3197	0.2529	0.2576	0.2369	0.2569	0.2179	0.2688
CCSO-DV-Hop	0.3064	0.2373	0.2399	0.2196	0.2446	0.1983	0.2524
PCSO-DV-Hop	0.3055	0.2386	0.2390	0.2156	0.2414	0.2077	0.2566
PCCSO-BR-DV-Hop	0.2827	0.2267	0.2126	0.1909	0.2137	0.1677	0.2252
PCCSO-AR-DV-Hop	0.2835	0.2299	0.2126	0.1913	0.2188	0.1623	0.2229
PCCSO-WR-DV-Hop	0.2896	0.2299	0.2202	0.1998	0.2391	0.1763	0.2329

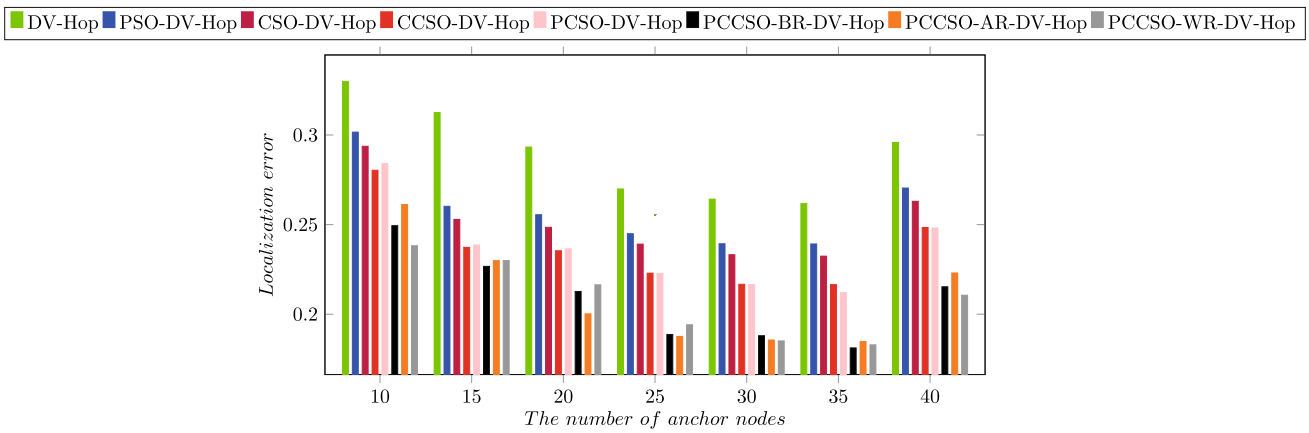


Fig. 8 Localization error results of variable anchor node

5.3 Communication radius change training

To more realistically simulate the layout environment of WSN, we randomly distribute 200 nodes in the monitoring area within 100 m × 100 m, and 20 nodes are randomly selected as anchor nodes and the rest 180 nodes as unknown nodes. The transmission range of WSN node determines the shortest distance between any two sensor nodes, which determines the coverage of WSN. The transmission power is the most important factor affecting the communication radius of sensor nodes. As the transmission power of sensor nodes increases, the communication radius will also increase. Then, the signal transmission capacity will become stronger, and the transmission

Table 7 Experimental results using applied PCCSO and other algorithms by different number of anchor nodes

Algorithm	10	15	20	25	30	35	40	Avg.
DV-Hop	0.3670	0.3298	0.3125	0.2933	0.2699	0.2642	0.2617	0.2958
PSO-DV-Hop	0.3486	0.3016	0.2624	0.2555	0.2449	0.2393	0.2392	0.2704
CSO-DV-Hop	0.3459	0.2937	0.2529	0.2485	0.2391	0.2332	0.2324	0.2630
CCSO-DV-Hop	0.3429	0.2803	0.2373	0.2354	0.2229	0.2167	0.2165	0.2484
PCSO-DV-Hop	0.3429	0.2841	0.2386	0.2365	0.2227	0.2165	0.2121	0.2481
PCCSO-BR-DV-Hop	0.3015	0.2494	0.2267	0.2127	0.1887	0.1880	0.1812	0.2153
PCCSO-AR-DV-Hop	0.3118	0.2612	0.2299	0.2002	0.1876	0.1856	0.1847	0.2230
PCCSO-WR-DV-Hop	0.3118	0.2382	0.2299	0.2164	0.1941	0.1851	0.1829	0.2106

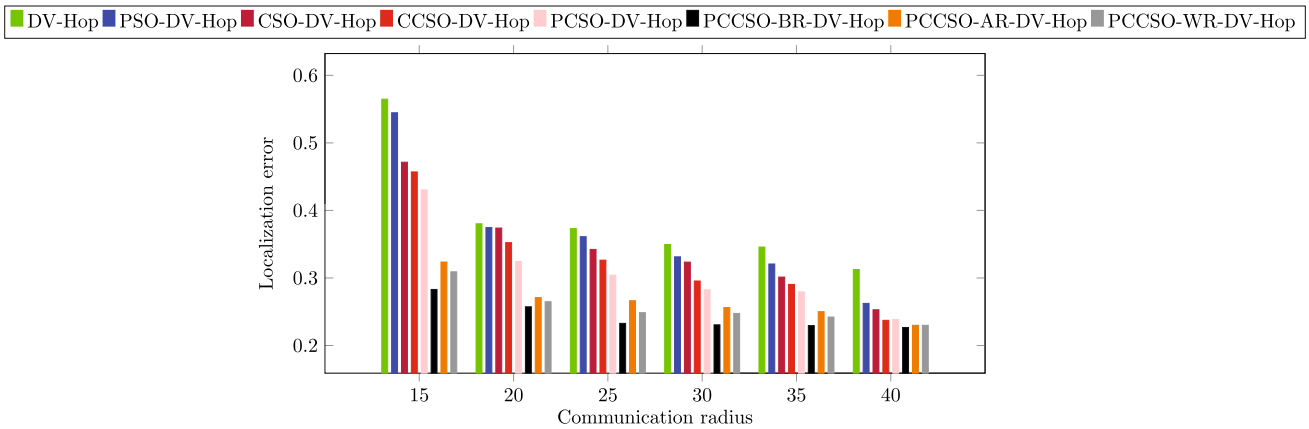


Fig. 9 Localization error results of variable communication radius

distance will also become longer. By changing the communication distance of nodes which is limited to a communication range from 15 to 40 m continuously, the results reported in Fig. 9 and Table 8 records the communication distance influence on location. The experimental results show that the proposed method can effectively achieve the desired purpose, which is to improve the positioning accuracy of WSN and reduce the complexity of DV-Hop. As can be seen from Fig. 9 and Table 8, the novel algorithm is more effective than others. As shown in Fig. 9, when arranging 200 nodes in monitor area and 10% of which is anchor nodes, the communication radius keeps changing, and increases at any time and the error becomes smaller and smaller. When the communication radius is greater than 20, the node error tends to be stable. However, when the communication radius is 15 and 40, the lowest error based on PCCSO is 0.2830 and 0.2267, which reflects that the PCCSO-based-DV-Hop is more robust and less influenced by the changes of communication radius compared to others. By increasing the transmission power, the communication radius of nodes is improved, and the ability of information transmission between nodes is enhanced. Thus, the positioning accuracy of WSN is improved. Through the simulation experiment, we confirm this point.

6 Conclusions

Aiming at optimization and localization problems, PCCSO is proposed and applied to WSN localization in this paper. Because compression algorithms use probabilistic models in spatial search to generate new candidate solutions and forward them to promising regions, compression techniques reduce running memory and reduce variables for computational optimization. In addition, we also proposed three parallel schemes to better find the best solution strategy to prevent the algorithm from falling into the local optimum. The proposed compact scheme in PCCSO indicates a significant improvement in the capacity of the global search for optimization compared to the original CSO algorithm, and it only needs a little memory in a real implement environment at the same time. Finally, this paper uses CEC2013 and WSN to locate problems to test PCCSO. All evidence shows that the proposed PCCSO test results are excellent.

Table 8 Experimental results using applied PCCSO and other algorithms by different ranges of communication

Algorithm	15 m	20 m	25 m	30 m	35 m	40 m	Avg.
DV-Hop	0.5647	0.3802	0.3730	0.3495	0.3458	0.3125	0.4386
PSO-DV-Hop	0.5446	0.3746	0.3611	0.3313	0.3207	0.2624	0.4035
CSO-DV-Hop	0.4714	0.3739	0.3423	0.3235	0.3013	0.2529	0.3622
CCSO-DV-Hop	0.4570	0.3523	0.3264	0.2955	0.2905	0.2373	0.3472
PCSO-DV-Hop	0.4304	0.3244	0.3041	0.2824	0.2793	0.2386	0.3345
PCCSO-BR-DV-Hop	0.2830	0.2573	0.2328	0.2306	0.2295	0.2267	0.2549
PCCSO-AR-DV-Hop	0.3236	0.2709	0.2663	0.2561	0.2501	0.2299	0.2768
PCCSO-WR-DV-Hop	0.3091	0.2648	0.2486	0.2475	0.2421	0.2299	0.2695

Acknowledgements This work is supported by National Natural Science Foundation of China (No. 61501106), Science and Technology Foundation of Jilin Province (Nos. 201801010-39JC, JJKH20200116KJ), and Science and Technology Foundation of Jilin City (No. 201831775).

References

1. Wang, H., Zhang, G., Mingjie, E., & Sun, N. (2011). A novel intrusion detection method based on improved SVM by combining PCA and PSO. *Wuhan University Journal of Natural Sciences, 16*(5), 409.
2. Qin, S., Sun, C., Zhang, G., He, X., & Tan, Y. (2020). A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems. *Complex & Intelligent Systems, 6,* 263274.
3. Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence, 1*(1), 33–57.
4. Wang, H., Liang, M., Sun, C., Zhang, G., & Xie, L. (2020). Multiple-strategy learning particle swarm optimization for large-scale optimization problems. *Complex & Intelligent Systems,* 1–16.
5. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69,* 46–61.
6. Pan, J. S., Hu, P., & Chu, S. C. (2019). Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power. *Processes, 7*(11), 845.
7. Hu, P., Pan, J., & Chu, S. (2020). Improved binary grey wolf optimizer and its application for feature selection. *Knowledge Based Systems*, 105746.
8. Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures, 139,* 98–112.
9. Du, Z. G., Pan, J. S., Chu, S. C., Luo, H. J., & Hu, P. (2020). Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks. *IEEE Access, 8,* 8583–8594.
10. Pan, J. S., Kong, L., Sung, T. W., Tsai, P. W., & Snášel, V. (2018). A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set. *Journal of Internet Technology, 19*(4), 1111–1118.
11. Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 2, pp. 1470–1477). IEEE.
12. Chu, S. C., Tsai, P. W., & Pan, J. S. (2006). Cat swarm optimization. In *Pacific Rim international conference on artificial intelligence* (pp. 854–858). Springer.
13. Neri, F., Mininno, E., & Iacca, G. (2013). Compact particle swarm optimization. *Information Sciences, 239,* 96–121.
14. Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation, 3*(4), 287–297.
15. Mininno, E., Neri, F., Cupertino, F., & Naso, D. (2010). Compact differential evolution. *IEEE Transactions on Evolutionary Computation, 15*(1), 32–54.
16. Tian, A. Q., Chu, S. C., Pan, J. S., Cui, H., & Zheng, W. M. (2020). A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station. *Sustainability, 12*(3), 767.
17. Pan, J. S., Dao, T. K., et al. (2019). A novel improved bat algorithm based on hybrid parallel and compact for balancing an energy consumption problem. *Information, 10*(6), 194.
18. Pan, J. S., Dao, T. K., et al. (2019). A compact bat algorithm for unequal clustering in wireless sensor networks. *Applied Sciences, 9*(10), 1973.
19. Pan, J. S., Song, P. C., Chu, S. C., & Peng, Y. J. (2020). Improved compact cuckoo search algorithm applied to location of drone logistics hub. *Mathematics, 8*(3), 333.
20. Zhao, M. (2018). A novel compact cat swarm optimization based on differential method. *Enterprise Information Systems, 14,* 1–25.
21. Xue, X., & Pan, J. S. (2018). A compact co-evolutionary algorithm for sensor ontology meta-matching. *Knowledge and Information Systems, 56*(2), 335–353.
22. Chu, S. C., Xue, X., Pan, J. S., & Wu, X. (2020). Optimizing ontology alignment in vector space. *Journal of Internet Technology, 21*(1), 15–22.
23. Song, P. C., Pan, J. S., & Chu, S. C. (2020). A parallel compact cuckoo search algorithm for three-dimensional path planning. *Applied Soft Computing*, 106443.
24. Pan, J. S., Kong, L., Sung, T. W., Tsai, P. W., & Snášel, V. (2018). α-Fraction first strategy for hierarchical model in wireless sensor networks. *Journal of Internet Technology, 19*(6), 1717–1726.
25. Chen, C. H., Lee, C. A., & Lo, C. C. (2016). Vehicle localization and velocity estimation based on mobile phone sensing. *IEEE Access, 4,* 803–817.
26. Wang, J., Gao, Y., Wang, K., Sangaiah, A. K., & Lim, S. J. (2019). An affinity propagation-based self-adaptive clustering method for wireless sensor networks. *Sensors, 19*(11), 2579.
27. Halder, S., & Ghosal, A. (2016). A survey on mobile anchor assisted localization techniques in wireless sensor networks. *Wireless Networks, 22*(7), 2317–2336.
28. Kulkarni, V. R., Desai, V., & Kulkarni, R. V. (2019). A comparative investigation of deterministic and metaheuristic algorithms for node localization in wireless sensor networks. *Wireless Networks, 25*(5), 2789–2803.

29. Zaruba, G. V., Huber, M., Kamangar, F. A., & Chlamtac, I. (2007). Indoor location tracking using RSSI readings from a single Wi-Fi access point. *Wireless Networks, 13*(2), 221–235.
30. GhasemAghaei, R., Rahman, M., Gueaieb, W. & El Saddik, A. (2007). Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. *2007 IEEE instrumentation & measurement technology conference IMTC 2007* (pp. 1–6). IEEE.
31. Shi, H. Y., Wang, W. L., Kwok, N. M., & Chen, S. Y. (2012). Game theory for wireless sensor networks: A survey. *Sensors, 12*(7), 9055–9097.
32. Sikeridis, D., Tsiropoulou, E. E., Devetsikiotis, M., & Papavassiliou, S. (2018). Wireless powered Public Safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency. *Journal of Network and Computer Applications, 123,* 69–79.
33. Fragkos, G., Apostolopoulos, P. A., & Tsiropoulou, E. E. (2018). ESCAPE: Evacuation strategy through clustering and autonomous operation in public safety systems. *Future Internet, 11*(1), 20.
34. Huang, X. L., Ma, X., & Hu, F. (2018). Machine learning and intelligent communications. *Mobile Networks and Applications, 23*(1), 68–70.
35. Kong, L., Chen, C. M., Shih, H. C., Lin, C. W., & Pan, J. S. (2014). An energy-aware routing protocol using cat swarm optimization for wireless sensor networks. *Lecture Notes in Electrical Engineering, 260,* 311–318.
36. Temel, S., Unaldi, N., & Kaynak, O. (2013). On deployment of wireless sensors on 3-D terrains to maximize sensing coverage by utilizing cat swarm optimization with wavelet transform. *IEEE Transactions on Systems Man & Cybernetics Systems, 44*(1), 111–120.
37. Kasana, R., & Kumar, S. (2017). A geographic routing algorithm based on cat swarm optimization for vehicular ad-hoc networks (pp. 86–90).
38. Kong, L., Pan, J. S., Tsai, P. W., Vaclav, S., & Ho, J. H. (2015). A balanced power consumption algorithm based on enhanced parallel cat swarm optimization for wireless sensor network. *International Journal of Distributed Sensor Networks, 11*(3), 729680.
39. Kanwar, V., & Kumar, A. (2020). DV-Hop localization methods for displaced sensor nodes in wireless. *Wireless Networks,* 1–12.
40. Gumaida, B. F., & Luo, J. (2019). Novel localization algorithm for wireless sensor network based on intelligent water drops. *Wireless Networks, 25*(2), 597–609.
41. Tsai, P. W., Pan, J. S., Chen, S. M., & Liao, B. Y. (2012). Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Systems with Applications, 39*(7), 6309–6319.
42. Chang, J. F., Chu, S. C., Roddick, J. F., & Pan, J. S. (2005). A parallel particle swarm optimization algorithm with communication strategies. *Journal of Information ENCE & Engineering, 21*(4), 809–818.
43. Liang, J., Qu, B., Suganthan, P., & Hernández-Díaz, A. G. (2013). *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization*. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report (Vol. 201212, No. 34, pp. 281–295).
44. Chen, X., & Zhang, B. (2012). Improved DV-Hop node localization algorithm in wireless sensor networks. *International Journal of Distributed Sensor Networks, 2012*(6), 1018–1020.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Jianpo Li received his B.S., M.S., and Ph.D. from the Department of Communication Engineering, Jilin University, China, in 2002, 2005, and 2008, respectively. In 2008, he joined the School of Information Engineering, Northeast Electric Power University (NEEPU). He was a visiting scholar with New York University in 2013 and University of Ottawa in 2016. Now he is the full professor and Vice-dean of the School of Computer Science, NEEPU. His research interests focus on wireless sensor networks, intelligent signal processing, 5G, wireless power transmission.

Min Gao received her B.S. degree from Nankai University Binhai College, China, in 2019. She is currently pursuing the master degree with the Northeast Electric Power University, Jinlin, China. Her research interests include intelligence algorithms and wireless sensor network.

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 National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He is currently the Professor in the College of Computer Science and Engineering, Shandong University of Science and Technology. He is also the IET Fellow, U.K., and has been the Vice Chair of the IEEE Tainan Section. He was offered Thousand Talent Program in China in 2010.

Shu-Chuan Chu received the Ph.D. degree in 2004 from the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. She joined Flinders University in December 2009 after 9 years at the Cheng Shiu University, Taiwan. She is the Research Fellow in the College of Science and Engineering of Flinders University, Australia from December 2009. Currently, She is the Research Fellow with PhD advisor in the College of Computer Science and Engineering of Shandong University of Science and Technology from September 2019. Her research interests are mainly in Swarm Intelligence, Intelligent Computing and Data Mining.