Applied Soft Computing 114 (2022) 108039

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Bifurcated particle swarm optimizer with topology learning particles

Reza Vafashoar [b,*], Hossein Morshedlou [a], Mohammad Reza Meybodi [b]

[a] Department of Computer Engineering and Information Technology, Shahrood University of Technology, Shahrood, Iran
[b] Soft Computing Laboratory, Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 24 July 2020
Received in revised form 23 August 2021
Accepted 29 October 2021
Available online 13 November 2021

Keywords:
Particle swarm optimization
Global numerical optimization
Swarm intelligence
Circular array antenna
Image segmentation

ABSTRACT

The flying speed and trajectory of particles are subject to several factors, including neighborhood structures, inertia weight, and acceleration coefficients. This paper improves particle swarm optimizer by exploiting these factors. Specifically, it proposes an approach to adjust the neighborhood structures of particles adaptively. The search task is divided between two groups of particles, termed even and uneven, to perform a vigorous in-depth search. Each particle group pursues a different objective and conducts its search in a different manner. Even particles adaptively adjust their number of attractors and neighborhood radiuses to experience various flying trajectories and paces. Each uneven particle follows a single even one for a while until it is assigned to another even particle. Uneven particles are responsible for performing fine-grained searches in the vicinity of their associated even particles as well as their previously experienced locations. A tree structure is utilized to implement the neighborhood structures of the proposed method. In the presented structure, particles can experience large neighborhoods by choosing their attractors from higher levels of the tree. The proposed method is experimentally investigated on the comprehensive CEC2013 benchmark set and two challenging real-world problems: non-uniform circular antenna array synthesis and image segmentation. The comparison results with advanced particle swarm optimization algorithms demonstrate that search bifurcation and topology adjustment can significantly improve particle swarm optimization. Experimental results also indicate that the proposed method can be successfully employed for solving challenging real-world problems with various characteristics.

1. Introduction

Particle swarm optimization (PSO) is a nature-inspired optimization method. It is inspired by the social behavior existing in bird flocking or fish schooling [1–3]. PSO systematically searches a given fitness landscape to find its optimum point. The algorithm maintains a swarm of particles to conduct its search. These particles constantly fly over the fitness landscape with specific velocities. The velocity of a particle governs its flying path over the problem search space. Particles iteratively update their velocities using their individual and social experiences to reach fitter areas. Due to its fast convergence speed and simplicity, PSO has widely been employed for solving scientific and engineering optimization problems. Some recent applications of PSO include band selection of hyperspectral images [4], feature selection [5], localization in wireless sensor networks [6], robotics [7,8], and optimal scheduling [9].

In canonical PSO, each particle is pulled by two attractors while having some tendency to continue flying along its previous direction. Accordingly, these three elements govern the flying trajectories of particles. A set of coefficients control the influence of these elements on the flying pattern of a particle. These coefficients deeply affect the flying pace of particles and consequently the convergence speed of the algorithm. In order to have more control over the flying pattern and pace of particles, various swarm topologies have also been investigated in the literature [10–12]. The swarm topology dictates which particles can influence each particle. It also governs the information spread throughout the swarm. Meanwhile, changing the number of attractors can lead to different trajectories and social behaviors. All these factors intensively affect the explorative and exploitative behavior of the algorithm.

While conical PSO uses two attractors to guide each particle, some PSO variants operate differently. In fully informed PSO (FPSO), each particle is influenced directly by all of its neighbors, rather than just the best one [13]. Hence, a particle can learn from the success of several neighbors at the same time. Like FPSO, particles of comprehensive learning PSO (CLPSO) are also allowed to follow several different exemplars simultaneously [14]. However, at each time step, a particle can follow only one exemplar in each

* Corresponding author.
E-mail addresses: vafashoar@aut.ac.ir (R. Vafashoar), morshedlou@shahroodut.ac.ir (H. Morshedlou), mmeybodi@aut.ac.ir (M.R. Meybodi).

https://doi.org/10.1016/j.asoc.2021.108039
1568-4946/

dimension. Both schemes have their advantages and drawbacks. The averaging effect that exists in FPSO would reduce its exploration ability and thus its effectiveness on complex multi-modal problems. However, FPSO can effectively exploit the local patterns of the fitness landscape. In contrast to FPSO, CLPSO is mainly developed for effective exploration of the problem landscape.

Several improved PSO variants have been developed by exploiting different neighborhood topologies. Some of these variants start with small local neighborhoods for better explorations and gradually increase the neighborhood size for better exploitation [15]. In the PSO variant presented by Bonyadi et al. [16], population topology changes over time from several small swarms at the beginning to few large swarms at the end. In another work, Xia et al. [17] suggested a dynamic strategy that periodically reduces the number of sub-swarms. Although several studies indicated that increasing the neighborhood size or connectivity can improve the performance of PSO, decreasing the neighborhood size or connectivity can be beneficial on some problems [18]. Cellular learning automata based multi-swarm optimization algorithm (CLAMS) uses a reinforcement learning approach to gradually learn the most promising connectivity degree for each particle [19]. Dynamic tournament topology PSO (DTT-PSO) utilizes a tournament strategy for repeatedly choosing an appropriate exemplar set for each particle to follow [20]. New local rules and neighborhoods have also been investigated in several works [21–25].

Maintaining a swarm with a well-balanced explorative and exploitative behavior is critical for obtaining accurate solutions. Exploration guides the swarm out of poor local optima and prevents premature convergence. Exploitation is a consequence of precise and in-depth search in promising areas, which is required for obtaining high-quality solutions. Achieving desirable explorative and exploitative abilities is the ultimate challenge of designing a robust PSO. Several adaptive mechanisms have been proposed in the literature to achieve this robustness. These mechanisms try to adjust various aspects of the algorithm, such as inertia weight, dynamically during the search process [26–29]. However, it is hard to describe the general swarm topology via few parameters controllable by the common adaptive mechanisms. Accordingly, less work has been conducted on developing adaptive methods regarding the swarm topology.

This paper presents an enhanced particle swarm optimizer called bifurcated dynamic topology PSO (BDTPSO). BDTPSO enhances PSO by diversifying the searching patterns of particles. It also adjusts the flying trajectories of particles adaptively according to the characteristics of the fitness landscape. To achieve these goals, the search process of BDTPSO is bifurcated among two types of corporative particles, called even and uneven particles. Each type of particle has its neighborhood structures and adjusts these structures according to the fitness landscape. In order to have an efficient adaptive topology mechanism, BDTPSO only considers the most significant aspects of the swarm topology. Even particles adaptively adjust their number of attractors and neighborhood radiuses. Accordingly, they can reach a well-balanced explorative and exploitative search behavior. The components of these particles tend to move in tandem, which is beneficial for solving problems with correlated and inseparable variables. Uneven particles serve a complementary and different objective. Each uneven particle can follow only a single attractor at each dimension. Accordingly, these particles can easily improve their unfit components. They are designated to perform fine-grained searches in the proximity of the fit locations discovered by even particles. These particles can adaptively change their exemplars to find the most suitable attractors.

Acceleration coefficients are the most significant parameters of the PSO algorithm. Proper settings for these parameters are very problem-dependent, and each variant of the PSO requires careful settings. Here, we use a simple technique to adjust the acceleration coefficients adaptively during the search process. The employed adaptation technique is similar to those developed in some differential evolution literature [30]. Although this adaptation mechanism is not the best choice for a PSO method, it can be employed to obtain acceptable settings. Also, this adaptation mechanism eliminates the need for an exhaustive search to find appropriate values for the acceleration coefficients.

The rest of the paper is organized as follows: Section 2 presents some background on particle swarm optimization algorithm and one of its variants called CLPSO. In Section 3, we describe the proposed algorithm. Section 4 introduces the antenna array design problem considered in this paper. Section 5 introduces another real-world application. Experimental results are presented in Section 6. Finally, some brief concluding comments are provided in Section 7.

2. PSO and CLPSO

In this section, we briefly review PSO and one of its well-known variants called CLPSO.

2.1. PSO

PSO is a nature-inspired optimization algorithm. It is inspired by the swarm social behavior existing in bird flocking or fish schooling [1,2]. PSO uses a swarm of particles to search for the optimum solution to a given problem. Each particle of the swarm records its current flying velocity and its current position in the search space. It also keeps track of the fittest position it has encountered thus far, which is called its personal best position or pbest. We use three D-dimensional vectors $x_i = \left[x_i^1, \ldots, x_i^2, \ldots, x_i^D\right]$, $p_i = \left[p_i^1, \ldots, p_i^2, \ldots, p_i^D\right]$, and $v_i = \left[v_i^1, \ldots, v_i^2, \ldots, v_i^D\right]$ to represent the current position of a particle like X_i, its pbest, and velocity, respectively. The fittest position encountered by the whole swarm is called the global best or gbest of the swarm and is denoted by $\mathrm{pg} = [pg^1, pg^2, \ldots, pg^D]$. To perform its search, each particle iteratively updates its flying velocity and current position according to the following Equations:

$$v_i^d(k+1) = w v_i^d(k) + c_1 r_1^d \left(p_i^d(k) - x_i^d(k)\right) + c_2 r_2^d \left(pg^d(k) - x_i^d(k)\right), \quad (1)$$

$$x_i^d(k+1) = v_{id}(k+1) + x_i^d(k), \quad (2)$$

where w, c_1, and c_2 are, respectively, inertia, cognitive, and social weights. r_1^d, and r_2^d are two random numbers drawn from the uniform distribution u [0,1]. After moving to a new position, the personal best position of the particle is updated, which may also affect the global best position of the swarm. Considering the above equations, PSO searches the fitness landscape by simply adjusting the trajectory of each particle towards gbest and its pbest. The inertia weight w maintains a balance between the global and local search [3].

2.2. CLPSO

The updating mechanism of canonical PSO has long been criticized as a reason for its stagnation and premature convergence. To improve the exploration ability of the algorithm, Liang et al. suggested a novel updating scheme [14]. In their proposed method, called CLPSO, each particle exclusively follows a single attractor in each dimension. However, particles are allowed to

have as many different exemplars as their number of dimensions. Consequently, the velocity updating equation of CLPSO is defined as follows:

$$v_i^d(k+1) = w v_i^d(k) + c_1 r_1^d \left(p_{f_i(d)}^d(k) - x_i^d(k)\right), \quad (3)$$

where $f_i = [f_i(1), \ldots, f_i(D)]$ represents the index of attractors for particle X_i. $f_i(d)$ is defined in the following manner. First, two particles X_j and X_l are selected randomly from the swarm. Without loss of generality, assume that X_j is fitter than X_l. Then, $f_i(d)$ is set to either i with probability Pc_i or to j with probability 1-Pc_i. Pc is called the learning probability of particles and controls their explorative and exploitative properties. Each particle X_i has its learning probability, which is defined as follows:

$$Pc_i = 0.05 - 0.45\frac{exp\left(\frac{10(i-1)}{N-1}\right) - 1}{\exp(10) - 1}, \quad (4)$$

where N is the number of particles in the swarm.

Using the defined exemplars, each particle X_i strives to find a better position in the search space. If the personal best position of the particle does not get better after a fair number of iterations, it can be assumed that the exemplar set defined by f_i is not appropriate for X_i. At this point, CLPSO re-defines f_i as described previously. CLPSO uses a parameter called refreshing gap, denoted by m to control the number of iterations required before changing the exemplars of a particle.

3. The proposed algorithm

This section introduces bifurcated dynamic topology PSO (BDTPSO). BDTPSO modifies the neighborhood structures of particles adaptively to achieve an enhanced explorative and exploitative behavior. Neighborhood structures control the information spread in the swarm. Large and fully connected topologies induce high selection pressure; meanwhile, small or loosely connected topologies can promote swarm diversity [31–33]. Swarm topology also determines which particles attract each particle of the swarm. As a result, they affect the flying patterns and trajectories of particles.

BDTPSO diversifies the search patterns of the swarm by using two types of particles called even and uneven particles. Even particles use an updating scheme similar to that of FIPS [34]. However, they can have any number of exemplars, and their neighborhood radius can change dynamically. An adaptive mechanism adjusts the neighborhood size and the number of exemplars of even particles according to their surrounding fitness landscape. Consequently, these particles can experience different degrees of exploration and exploitation. The components of an even particle tend to move in tandem. As a result, even particles can move in trajectories consistent with the existing interdependencies among the problem variables. Also, even particles have high exploitative capabilities. These features make even particles suitable for solving complex problems with inseparable and correlated variables. However, simultaneous attractions to multiple points can confound a particle [34]. Also, in highly connected topologies, when particles have many attractors, the search region will shrink to the area around the swarm centroid. To overcome these drawbacks, BDTPSO uses uneven particles that serve different and complementary objectives.

Uneven particles use an updating scheme similar to that of CLPSO. While identical attractor particles affect different components (dimensions) of an even particle, the attractors that affect different components of an uneven particle can be dissimilar. Each uneven particle has two exemplars: its pbest and an even particle. However, each of its components can be pulled towards only one of the exemplars. Accordingly, the unfit components of these particles can be improved separately and rapidly. Uneven particles change their exemplars dynamically. When an uneven particle cannot get fitter after some iterations, it redefines the exemplars for its components. BDTPSO uses mutually different attractors for uneven particles to enhance the diversity of the swarm. Experiencing different combinations of sources of attraction provides uneven particles with high exploration capabilities. Also, these particles provide a fine-grained search in the proximity of previously discovered promising regions.

BDTPSO employs an equal number of even and uneven particles. In order to control the neighborhood structures of even particles dynamically, the algorithm maintains two parameters for each one: tournament size and the number of exemplars. The number of exemplars of an even particle determines how many attractors are affecting the particle. To determine each attractor of an even particle, tournament selection is employed. The number of participants in the tournament is defined by the tournament size parameter of the particle. In order to simplify the description of the algorithm and increase its computational efficiency, a binary tree data structure is utilized in the implementation.

All particles are stored in the leaves of a binary tree, where each leaf node holds exactly one particle. The leaf nodes are labeled from 1 to N. Each leaf node with an even label holds an even particle like X_i, while its sibling node holds the uneven particle which follows X_i. The levels of the tree can be defined recursively as follows. The leaf nodes constitute the first level of the tree. If a node resides in the kth level of the tree, then its parent is considered to be in the $(k+1)$th level. Each internal node contains a pointer to the fittest particle in its rooted subtree (to ease the description of the algorithm, we will say that each internal node holds the fittest particle of its rooted subtree, instead). Fig. 1 shows the described tree structure. Additionally, Table 1 provides some useful notations.

3.1. The updating procedure of BDTPSO

BDTPSO uses the procedure given in Fig. 2 to update its particles. Uneven particles are updated using their sibling even particles. Specific acceleration coefficients are calculated for each particle dynamically during the updating step. BDTPSO maintains two archives of promising acceleration coefficients for this purpose. The set of promising acceleration coefficients for uneven particles is denoted by ψ. During the kth iteration, each uneven particle X_i selects a promising acceleration coefficient from this set. Using this acceleration coefficient, the acceleration coefficient of the particle is calculated as follows:

$$c_i(k) = Max\left[Min\left\{\psi_{ri} + \varsigma(0, 0.2), C^{MAX}\right\}, C^{MIN}\right], \quad (5)$$

where ψ_{ri} is an acceleration coefficient randomly picked from ψ. $\varsigma(0,0.2)$ is a random number generated by a Cauchy distribution with a mean of 0 and a standard deviation of 0.2. C^{MIN} and C^{MAX} represent, respectively, the minimum and maximum allowable values for an acceleration coefficient. In short, each uneven particle uses a slightly modified archived acceleration coefficient. After obtaining its acceleration coefficient, the velocity of the particle is updated according to the following Eq.:

$$v_i^d(k) = w v_i^d(k-1) + c_i(k) r_i^d \left(p_{Qi,d}^d(k-1) - x_i^d(k-1)\right), \quad (6)$$

where $Q_i = [Q_{i,1}, \ldots, Q_{i,D}]$, defines the index of attractors for each component (dimension) of X_i. Q is called attractor index, and its calculation is described in Section 3.3.

During each iteration, an even particle X_i uses its pbest along with n_i exemplars from the l_i^{th} level of the tree to calculate a new velocity. n_i and l_i define the neighborhood structure of the particle and are adapted during the search procedure (this adaptation procedure is described in the subsequent sections).

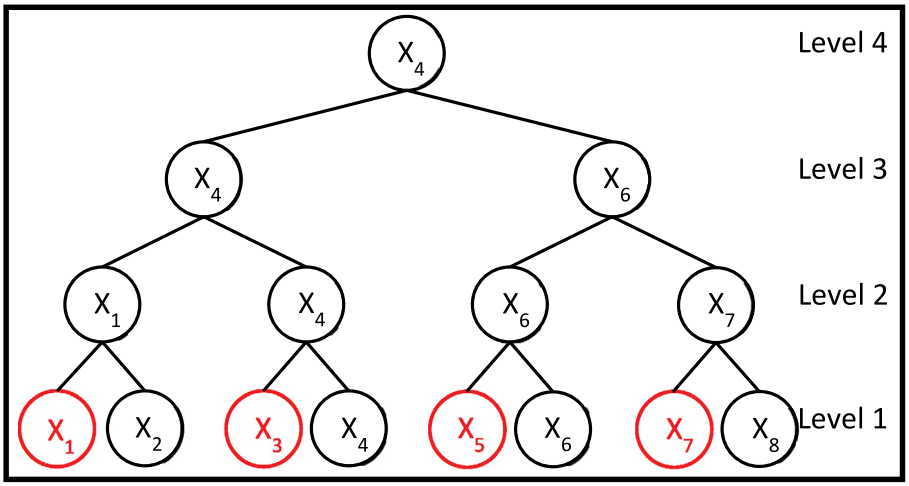

Fig. 1. The organization of particles in BDTPSO. Uneven particles and their corresponding nodes are shown in red. In this particular example, it is assumed that $f(X_4) < f(X_6) < f(X_7) < f(X_1) < f(X_2) < f(X_3) < f(X_5) < f(X_8)$.

Table 1
The notations that are used in the proposed method.

Notation	Description	Notation	Description
X_i	The ith particle of the swarm	D	Number of dimensions of the problem
p_i	The personal best position of X_i	N	Swarm size
x_i	The current position of X_i	$a(s, l)$	The ancestor of node s at the lth level
v_i	The velocity of X_i	ξ	The archive of promising acceleration coefficients for even particles
Pc_i	Learning probability for particle X_i	$\chi(s)$	The index of the particle associated with node s
$\rho(l)$	The number of nodes at level l, which is $N/(2^{l-1})$	$node(X_i)$	the node holding X_i at the first level
$Q_{i,d}$	The attractor index of the dth component (dimension) of X_i	ψ	The archive of promising acceleration coefficients for uneven particles
ϖ.	The archive of promising neighborhood structures for even particles	$e(s, n)$	If $n = 1$, $e(s, n) = s$ Otherwise $e(s, n) = s \cup e(next(s), n\text{-}1)$
m	Refreshing gap	$next(s)$	The immediate right node of node s in the tree

Two acceleration coefficients are used in the updating stage of an even particle. These acceleration coefficients are calculated for each particle at each iteration using an archive of promising acceleration coefficients. The archive of promising acceleration coefficients for even particles is denoted by ξ. During iteration k, the acceleration coefficients of each even particle X_i are calculated as follows:

$$c_i^1(k) = Max\left[Min\left\{\xi_{ri}^1 + \varsigma\left(0, 0.2\right), C^{MAX}\right\}, C^{MIN}\right], \tag{7}$$

$$c_i^2(k) = Max\left[Min\left\{\xi_{ri}^2 + \varsigma\left(0, 0.2\right), C^{MAX}\right\}, C^{MIN}\right], \tag{8}$$

where ξ_{ri} is a tuple of acceleration coefficients randomly picked from ξ. Using the calculated acceleration coefficients, the velocity of the particle is updated as follows:

$$\begin{aligned} v_i(k) = {} & w v_i(k-1) + c_i^1(k)\, r_{i1} \otimes \left(p_i(k-1) - x_i(k-1)\right) \\ & + s_i \frac{c_i^2(k)}{n_i} r_{i2} \otimes \sum_{u \in e(a(node(X_i).l_i), n_i)} \left(p_{\chi(u)}(k-1) - x_i(k-1)\right) \\ & + (1 - s_i) \frac{c_i^2(k)}{n_i} r_{i3} \otimes \sum_{u \in e(a(node(X_i).l_i), n_i)} \left(p_{\chi(u)}(k-1) - x_i(k-1)\right), \end{aligned} \tag{9}$$

where r_{i1} and r_{i2} are D-dimensional vectors whose components are sampled from u(0,1). The '⊗' symbol is the Hadamard (element-by-element matrix multiplication) operator. s_i and r_{i3} are real numbers sampled from u(0,1). It should be noted that Eq. (9) uses both linear and classical scaling schemes [35], and the random number s_i determines the weight of each scheme. Linear scaling schemes use random scalers instead of vectors of random scalers (r_1 and r_2 in Eq. (1)). Both schemes have their advantages and drawbacks [35], and Eq. (9) tries to combine their benefits.

The new position of each particle is acquired by Eq. (2) after its velocity is updated. Then, this position is evaluated using a fitness function, and the personal best position of the particle is updated accordingly (line 3.a of Fig. 2). BDTPSO maintains the number of consecutive failures of each particle in *fail_count* for later use. Moreover, considering line 3.a, BDTPSO sets the current position of each particle to its previous *pbest* upon successful updates. The intuition behind this step is as follows. Based on our empirical studies, each even particle like *Xi* is frequently updated using $n_i = 1$ and $l_i = 2$. In this case, X_i is pulled towards p_i and the *pbest* of the particle pointed by the node that $a(node(X_i), 2)$, while carrying some inertia of its old velocity. There is a good chance that $a(node(X_i), 2)$ also points to X_i itself. As a result, the particle may move in very small steps for several iterations after a successful update (considering canonical updating rules, p_i is equal to x_i after successful updates). In order to prevent this, BDTPSO changes x_i to the previous *pbest*, which is a promising position and is different from the updated *pbest*.

3.2. Archival memories

As discussed in the previous section, each particle uses an archival memory to calculate its required acceleration coefficients. BDTPSO also maintains an archival memory of promising neighborhood structures for even particles, which is denoted by ϖ. Each entry in ϖ is an ordered pair that defines the number of exemplars and the tree level of an even particle. BDTPSO updates the archival memories after updating its swarm (Fig. 3). In order to update the archival memories, BDTPSO first identifies

1. For each uneven particle X_i do:
   a. Obtain $c_i(k)$ using Eq. 5.
   b. Update v_i using Eq. 6.
   c. Update $x_i(k)$ using Eq. 2.
2. For each even particle X_i do:
   a. Obtain c_i^1(k) and c_i^2(k) using Eq. 7 and Eq. 8, respectively.
   b. Update v_i using Eq. 9.
   c. Update x_i using Eq. 2.
3. For each particle X_i do:
   a. If $f(x_i(k))<f(p_i(k\text{-}1))$ then set $p_i(k)=x_i(k)$; set $x_i(k)=p_i(k\text{-}1)$; set $fail\_count(X_i)=0$.
   b. Else set $p_i(k)=p_i(k\text{-}1)$; set $fail\_count(X_i)= fail\_count(X_i)+1$.

Fig. 2. The updating procedure of BDTPSO.

the most promising even and uneven particles. Then, it utilizes the acceleration coefficients and structural parameters (for even particles) of these particles to update the corresponding archives. The procedure of identifying promising particles is as follows. BDTPSO first determines the degree of improvement for each particle X_i as follows:

$$\delta_i = f\left(p_i\left(k-1\right)\right) - f\left(p_i\left(k\right)\right). \tag{10}$$

After calculating the degree of improvement of each particle, BDTPSO identifies the $\tau N/2$ even and $\tau N/2$ uneven particles with the highest improvement degrees. Among these particles, the ones with non-zero improvement degrees are considered promising particles. Let τ_1 and τ_2 represent, respectively, the number of promising uneven and even particles. BDTPSO replaces the τ_1 oldest entries of ψ with the acceleration coefficients of the promising uneven particles. Similarly, the τ_2 oldest entries of ξ are replaced by the acceleration coefficients of the promising even particles. Finally, BDTPSO substitutes the τ_2 oldest entries of ϖ with the structural parameters of even particles.

3.3. Adapting neighborhood structures and updating tree nodes

BDTPSO dynamically changes the neighborhood structures and exemplars of particles during the search procedure to achieve an in-depth and balanced search (Fig. 4). At the end of each iteration, BDTPSO identifies the ineffective particles. Ineffective particles are those that have failed to make any progress during the later m consecutive iterations. m is called the refreshing gap and has a similar role to the refreshing gap of CLPSO.

BDTPSO deals with ineffective uneven and even particles in different ways. Let S_u and T_u denote, respectively, the set of uneven ineffective particles and their corresponding tree nodes. First, BDTPSO randomly rearranges the ineffective uneven particles in the nodes in T_u (one particle in one node). Then, BDPTPSO redefines the attractor index (Q) for each particle X_i in S_u as follows. A D-dimensional vector r is generated randomly, where each component r_d is sampled from u(0,1). Then, $Q_{i,d}$ is set to i if $r_d < Pc_i$; otherwise, it is set to $\chi(next(node(X_i)))$.

BDTPSO redefines the structural parameters of each ineffective even particle X_i as follows. First, one element is selected randomly from ϖ. Let this element be (a, b). Then, BDTPSO slightly modifies this element with some probability by randomly increasing or decreasing a, b, or both by one. After that, the obtained parameters may be reinitialized randomly with a small probability. Finally, (n_i, l_i) is set to (a, b).

Before proceeding to the next iteration, BDTPSO updates its tree structure. This updating phase simply involves iterating each pair of sibling nodes and comparing the fitness of their associated particles. The node iterating process begins from the leaves and continues towards the root. When the process reaches a pair of nodes like s and $next(s)$ at level l of the tree, it identifies the one with the fittest particle and assigns its particle to $a(s, l+1)$. Fig. 5 shows the general building blocks of the proposed method.

3.4. Time complexity of BDTPSO

This section investigates the asymptotic time complexity of BDTPSO. In this regard, the computational cost of each iteration of BDTPSO is measured. The computational cost of fitness evaluations is ignored in the calculations as different algorithms are compared using an equal number of fitness evaluations.

The initialization phase requires o(ND) time for generating random velocity and position vectors. The binary tree data structure has $2N-1$ nodes and can be initialized in o(N) time. This data structure can be formed by only storing the index of particles in an array of size $2N-1$. Computing Q has a time complexity of o(ND). ψ, ξ, and ϖ all have a size of o(N) and are randomly initialized within the feasible range, which takes o(N) time. Considering all steps of the initialization phase, the time complexity of this phase is o(DN).

The updating phase has a time complexity of o(ND). Considering that the number of exemplars is bounded by a constant number, updating velocity and position vectors requires an o(ND) time. Also, selecting parameters from archival memories and changing their values requires just a few steps for each particle, making its overall computational cost O(N). Finally, updating personal best positions has a time complexity of o(ND).

Updating each archival memory is performed by choosing at most τN particles and updating at most $o(\tau N)$ array entries. Identifying promising particles can be performed efficiently in a linear time, O(N). Updating $o(\tau N)$ array entries takes o(τN) time. Accordingly, the overall time complexity of this phase, considering three archival memories, is o($3N$)=o(N).

Adapting neighborhood structures has a time complexity of o(ND/m+N). Identifying ineffective particles simply requires checking the *fail_count* array, which has a time complexity of o(N). Changing neighborhood structures requires at most o(D) time for a particle, which is the time required for defining the corresponding entries of Q. However, the neighborhood structures of each particle may be changed at most once every m iterations. Consequently, the time complexity of changing the neighborhood structures over m iterations is at most o(ND), making its average time complexity in each iteration o(ND/m). The employed tree data structure can be updated at the end of each iteration in o(N) time. There are $2N-1$ nodes in this tree, and each pair of sibling nodes are traversed and processed once to update their parent node.

Considering the time complexities of all phases, the time complexity of each iteration of BDTPSO is o(ND), which is similar to the time complexity of canonical PSO. While BDTPSO uses several additional steps, all these steps can be performed by simple calculations per particle.

4. Designing antenna array using BDTPSO

Antenna array design and synthesis can be modeled as a global optimization problem. According to the requirements, this problem may involve features like side lobe level, physical size, dynamic range ratio, gain, and radiation pattern. In this section, we consider the non-uniform circular antenna array design problem [36]. Such antennas are required in many communication systems. The array consists of N elements, which are placed non-uniformly on a circle of radius r (Fig. 6). Since these elements are considered to be isotropic sources, the associated array factor can be described by the following equation:

$$AF(\varphi) = \sum_{n=1}^{N} I_n e^{j(kr\cdot\cos(\varphi-\varphi_n)+\beta_n)}, \tag{11}$$

where I_n, β_n, and φ_n denote, respectively, the current amplitude, phase, and angular position of the nth element. Without loss of

1. Calculate the improvement degree of each particle using Eq. 10.
2. Let S be the set of $\tau N/2$ uneven particles with the highest improvement degrees.
3. If a particle in S has zero improvement degree, remove it from S.
4. For each particle $X_i \in S$ do:
   a. Set $\psi_{oldest\_in\_\psi} = c_i(k)$.
   b. Set $oldest\_in\_\psi = (oldest\_in\_\psi)\ \%\ (N/2) + 1$. // % represents the modulo operator
5. Let S be the set of $\tau N/2$ even particles with the highest of improvement degrees.
6. If a particle in S has zero improvement degree, remove them from S.
7. For each particle $X_i \in S$ do:
   a. Set $\xi_{oldest\_in\_\xi} = [c_i^1(k), c_i^2(k)]$.
   b. Set $\varpi_{oldest\_in\_\xi} = [n_i(k), l_i(k)]$.
   c. Set $oldest\_in\_\xi = (oldest\_in\_\xi)\ \%\ N/2 + 1$.

Fig. 3. Updating archival memories.

1. Set $S_1 = \emptyset$, $S_2 = \emptyset$.
2. Add each uneven particle X_i with $fail\_count(X_i) > m$ to S_1.
3. Add each even particle X_i with $fail\_count(X_i) > m$ to S_2.
4. For each particle $X_i \in S_1 \cup S_2$ Set $fail\_count(X_i) = 0$.
5. Let T_u be the set of tree nodes containing particles in S_1.
6. Randomly rearrange the particles in S_1 into the nodes in T_u.
7. For each particle $X_i \in S_1$ do:
   a. For d=1 to D do:
      i. With probability Pc_i set $Q_{i,d} = i$; otherwise, set $Q_{i,d} = \chi(next(node(X_i)))$.
8. For each particle $X_i \in S_2$ do:
   a. Select a random element (a, b) from ϖ.
   b. Set $(n_i, l_i) = (a, b)$.
   c. With probability 0.5:
      i. Set flag = false.
      ii. With probability 0.5 do:
         1. Randomly increase or decrease l_i by 1.
         2. If $l_i < 1$, set $l_i = b$+1. // ensuring lower limit of l_i
         3. If $l_i > \log_2(N)$+1, set $l_i = b$-1. // ensuring upper limit of l_i
         4. If $n_i > \rho(l)$, set $n_i = \rho(l)$.
         5. Set flag = true.
      iii. If $\rho(l)$>1, with probability 0.5 do:
         1. Randomly increase or decrease n_i by 1.
         2. If $n_i > \rho(l)$, set $n_i = a$-1.
         3. Set flag = true.
      iv. If flag = false, with probability 0.25
         1. Select random feasible values for n_i and l_i.
9. For each level l from 1 to $\log_2(N)$ do:
   a. For each pair of sibling nodes s and u=$next(s)$ at level l do:
      i. If $f(p_{\chi(s)}) < f(p_{\chi(u)})$, set $\chi(a(s, l+1))$ to $\chi(s)$
      ii. Else set $\chi(a(s, l+1))$ to $\chi(u)$

Fig. 4. Adapting neighborhood structures and updating tree nodes.

generality, we can assume that the peak of the radiation pattern is directed along the x-axis, i.e. $\varphi_0 = 0$. If we assume that the distance between the nth and the $(n-1)$th elements is d_n, then kr, φ_n, and β_n can be defined as follows:

$$kr = \sum_{n=1}^{N} d_n, \tag{12}$$

$$\varphi_n = \frac{2\pi}{kr} \sum_{n=1}^{N} d_n, \tag{13}$$

$$\beta_n = -kr \cdot \cos(\varphi_0 - \varphi_n). \tag{14}$$

4.1. Optimization problem for the antenna array synthesis

Here, we wish to adjust the amplitudes and the distances of the elements in order to obtain an antenna array with a specific first null bandwidth (FNBW) and the minimum side lobe level. This problem can be modeled as an optimization problem, which has terms to ensure the minimum side lobe level and the specified first null bandwidth. In order to minimize the side lobe level, two terms are considered in the optimization problem. The first term is intended to decrease the maximum existing side lobe in the radiation pattern. The second term represents the integral of the side lobes. The array factor is always minimal in the nulls of the radiation pattern. Hence, minimizing the array factor in the specified first nulls can guarantee the specified first null bandwidth. The general optimization problem can be specified as follows [36]:

$$min_{I,d}(f(I, d) = a.f_{NU} + b.f_{SLA} + c.f_{MSL}), \tag{15}$$

with

$$f_{NU} = |AFn(\varphi_{NULL1})| + |AFn(\varphi_{NULL2})|\,,$$

$$f_{SLA} = \frac{1}{\pi + \varphi_{NULL1}} \int_{-\pi}^{\varphi_{NULL1}} |AFn(\varphi)|\, d\varphi + \frac{1}{\pi - \varphi_{NULL2}} \int_{\varphi_{NULL2}}^{\pi} |AFn(\varphi)|\, d\varphi$$

,

$$f_{MSL} = |AFn(\varphi_{MSLL1})| + |AFn(\varphi_{MSLL2})|\,,$$

$$AFn = \frac{AF}{max\,|AF|},$$

where φ_{NULL1} and φ_{NULL2} are the angles at the null; φ_{MSLL1} is the angle where the maximum side lobe level resides in the lower band, i.e. $[-\pi, NULL1]$; φ_{MSLL2} is the angle where the maximum side lobe level of the upper band, i.e. $[NULL2, \pi]$ resides. f_{MSL} and f_{SLA} are incorporating, respectively, the maximum side lobe level and the average side lobe level in the fitness function. f_{NU} aims at adjusting the FNBW according to the two fixed values of nulls i.e. φ_{NULL1} and φ_{NULL2}. a, b, and c represent the weight of each objective in the fitness function. To put equal weight on each of the design objectives, we assume that $a = b = c = 1$. Moreover, to alleviate the mutual coupling between elements we consider d_n to be at least 0.5λ.

Designing circular antenna arrays with non-uniformly spaced elements, maximum directivity, and minimum side lobe level is a challenging optimization problem. Canonical approaches used for antenna array synthesis and optimization are usually stuck in poor local optima if their initial guesses are not near the global optimum of the problem. BDTPSO uses several techniques to guarantee an explorative and in-depth search. Accordingly, the quality of its obtained solutions is not affected by the initial population. Additionally, the proper exploration and exploitation balance provided by the topology adaptation mechanism of BDTPSO and its dual particle types ensures high-quality solutions.

5. Multilevel image thresholding using BDTPSO

Multi-level thresholding is a popular image segmentation approach, widely employed in medical image analysis. This approach involves finding a set of optimum thresholds (according to

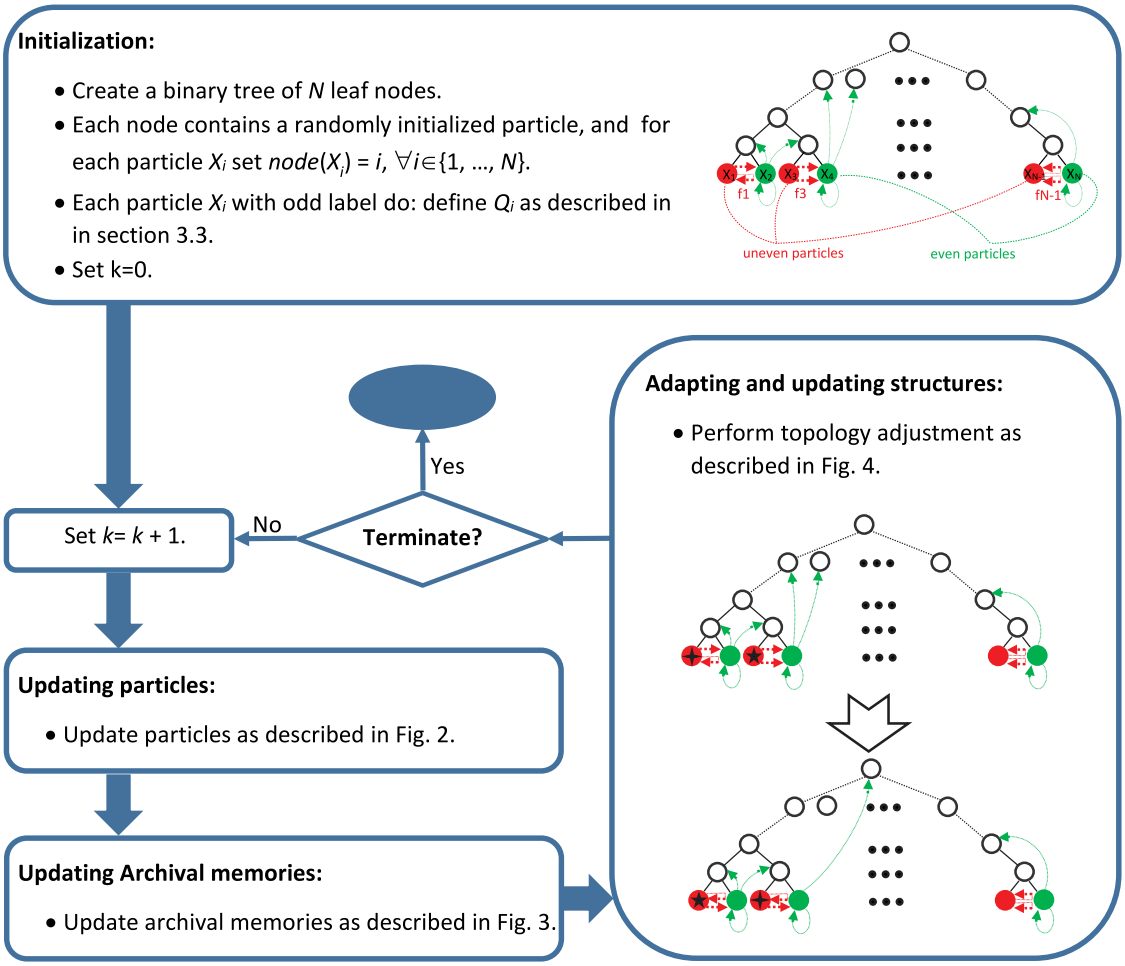

Fig. 5. General block diagram of BDTPSO.

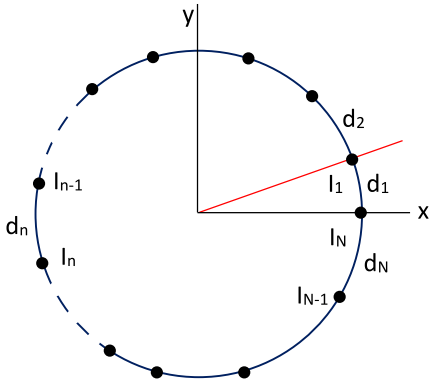

Fig. 6. A circular antenna array with N non-uniformly distributed elements [36].

some criteria) and using them to separate image pixels into different classes. Methods such as Kapur and Otsu are highly effective for bi-level thresholding. However, as the number of required thresholds increases, the computational cost of these methods grows dramatically. Accordingly, nature-inspired optimization methods are often utilized for finding suitable thresholds with manageable computational costs [37–39]. However, these methods usually suffer from a lack of robustness when dealing with a high number of thresholds. Because of the neat balance between exploration and exploitation properties, provided by the adaptive mechanism of BDTPSO, the proposed method can be effectively employed for finding even a large number of thresholds. In what follows, we formally define the problem and the criterion used for segmentation.

5.1. Problem definition

Consider a grayscale image like I, which is required to be partitioned into $K + 1$ segments. To this end, a thresholding approach finds an optimum set of K thresholds according to some criteria. Using these thresholds, each pixel of the image is segmented or classified according to its intensity as follows:

$$C_{i,j} = \begin{cases} C_1 \ if \ g_{i,j} \le T_1 \\ C_2 \quad if \ T_1 < g_{i,j} \le T_2 \\ \dots \\ C_K \quad if \ T_{K-1} < g_{i,j} \le T_K \\ C_{K+1} \quad if \ T_K < g_{i,j}, \end{cases} \tag{16}$$

where $g_{i,j}$ is the intensity or the gray level of pixel $I_{i,j}$; $C_{i,j}$ represents the segment or the class of the pixel, and $T = \{T_1, T_2, \dots, T_K\}$ is the set of the obtained thresholds.

Several criteria have been suggested for choosing suitable thresholds. Criteria based on the entropy concepts have proven to be quite effective for image thresholding. Recently, Sarkar et al. have shown that maximizing the total fuzzy entropy of the partitioned image can lead to even better solutions [40]. In what follows, we describe the concepts of their proposed method in more detail.

5.2. Multi-level Shannon entropy

Consider an image I with L gray levels, and assume that q_i is the probability distribution of the ith gray level. Let $T = \{T_1, T_2, \ldots, T_K\}$ be the set of thresholds that is used for partitioning I into $K + 1$ segments. The entropy of each obtained segment can be defined as follows [40]:

$$H_1(T) = -\sum_{i=0}^{T_1} \frac{q_i}{P_1} ln\left(\frac{q_i}{P_1}\right), \tag{17}$$

$$H_2(T) = -\sum_{i=T_1+1}^{T_2} \frac{q_i}{P_2} ln\left(\frac{q_i}{P_2}\right),$$

$$\ldots$$

$$H_{K+1}(T) = -\sum_{i=T_K+1}^{L-1} \frac{q_i}{P_{K+1}} ln\left(\frac{q_i}{P_{K+1}}\right),$$

where

$P_1 = -\sum_{i=0}^{T_1} q_i$, $P_2 = -\sum_{i=T_1+1}^{T_2} q_i$, ... $P_{K+1} = -\sum_{i=T_K+1}^{L-1} q_i$.

The optimum thresholds, according to this entropy concept, can be obtained by solving the following equation [40]:

$$[T_1, T_2, \ldots, T_K] = \text{Argmax}_T(H_1(T) + H_2(T) + \cdots + H_{K+1}(T)). \tag{18}$$

5.3. Multi-level fuzzy entropy

Consider an image I with L gray levels, which is needed to be partitioned into $K + 1$ segments. Using the ideas from the fuzzy theory, we can define each segment by a fuzzy set. The universe of discourse of this fuzzy set is the set of all gray levels of the image, and its membership function represents the membership degree of gray levels to the related segment. Here, $K + 1$ trapezoidal membership functions $\mu_1, \ldots, \mu_{K+1}$ will be used to define image segments. These set of functions can be defined by $2K$ parameters like a_i, b_i, with $0 \le i \le K$, where $0 \le a_1 \le b_1 \le \ldots \le a_K \le b_K \le L$-1 (Fig. 7) [40]. The membership functions can be defined as follows:

$$\mu_1(i) = \begin{cases} 1 & i \le a_1 \\ \frac{i-b_1}{a_1-b_1} & a_1 \le i \le b_1 \\ 0 & i > b_1, \end{cases} \tag{19}$$

$$\ldots$$

$$\mu_K(i) = \begin{cases} 0 & i \le a_{K-1} \\ \frac{i-a_{K-1}}{b_{K-1}-a_{K-1}} & a_{K-1} \le i \le b_{K-1} \\ 1 & b_{K-1} \le i \le a_{K-1} \\ \frac{i-b_K}{a_K-b_K} & a_K \le i \le b_K \\ 0 & i \ge b_K \end{cases}$$

$$\mu_{K+1}(i) = \begin{cases} 0 & i \le a_K \\ \frac{i-a_K}{b_K-a_K} & a_K \le i \le b_K \\ 0 & i > b_1 \end{cases}.$$

Similar to the previous section, we can define the fuzzy entropy of segments as follows:

$$H_1(a.c) = -\sum_{i=0}^{L-1} \frac{q_i\mu_1(i)}{P_1} ln\left(\frac{q_i\mu_1(i)}{P_1}\right), \tag{20}$$

$$H_2(a.c) = -\sum_{i=0}^{L-1} \frac{q_i\mu_2(i)}{P_2} ln\left(\frac{q_i\mu_2(i)}{P_2}\right),$$

$$\ldots$$

$$H_{K+1}(a.c) = -\sum_{i=0}^{L-1} \frac{q_i\mu_{K+1}(i)}{P_{K+1}} ln\left(\frac{q_i\mu_{K+1}(i)}{P_{K+1}}\right),$$

where

$$P_j = -\sum_{i=0}^{L-1} q_i\mu_j(i).$$

We can find the optimum parameters of these fuzzy sets by solving the following problem:

$$(a_1, \ldots, a_K, c_1, \ldots, c_K) = \text{Argmax}_{a,c}(H_1(a,c) + H_1(a,c) + \cdots + H_{K+1}(a,c)). \tag{21}$$

In this paper, BDTPSO is employed to solve the optimization problem given in Eq. (21). Finally, to obtain non fuzzy thresholds the following equation can be used:

$$T_i = (a_i + b_i)/2, i \in \{1, 2, \ldots, K\}. \tag{22}$$

6. Experimental results

This section investigates the effectiveness of the proposed approach via a comparative study on the CEC2013 benchmark set [41], image segmentation, and the antenna array design problems. CEC2013 benchmark set includes three categories of problems: unimodal (f_1–f_5), basic multimodal (f_6–f_{20}), and composition (f_{21}–f_{28}). Interested readers can refer to [41] for a thorough description of the benchmark set. Four advanced variants of PSO are chosen for comparison on this benchmark set. These algorithms include: DTT-PSO [20], MCUPS [42], HCLPSO [43], and EPSO [44]. Like the proposed approach, DTT-PSO mainly exploits the topological properties of the swarm to perform an improved search; consequently, it can be an appropriate choice for comparison. HCLPSO is one of the recent variants of CLPSO and is chosen because BDTPSO embeds some features from CLPSO. Like BDTPSO, EPSO is a multi-strategy approach. However, this algorithm tries to learn the most promising search scheme. In the experimental studies, all peer methods use the same configurations suggested in their related literature. For comparisons on the antenna array design and image segmentation problems, three approaches are selected that have been successfully employed recently on various similar problems. These approaches are CSO [45], CLPSO [46], FFO [47].

6.1. Parameter settings

Like other nature-inspired optimization methods, BDTPSO has some specific parameters. However, most of its parameters are adaptively adjusted, and it is only required to define their upper and lower bounds. These upper and lower bounds can be loosely defined using few trails. In what follows, we thoroughly investigate the impact of each parameter on the performance of the proposed approach. We also discuss the intuition behind each parameter setting. The summary of the suggested parameter settings for BDTPSO is given in Table 2. Since we use the most common setting for the inertia weight, this parameter is not investigated in this section.

6.1.1. The impact of swarm size on the performance of BDTPSO

This section empirically investigates the impact of the swarm size on the performance of the proposed approach. In this regard, BDTPSO is examined using various swarm sizes on a set of multimodal and unimodal benchmarks from the CEC2013 benchmark

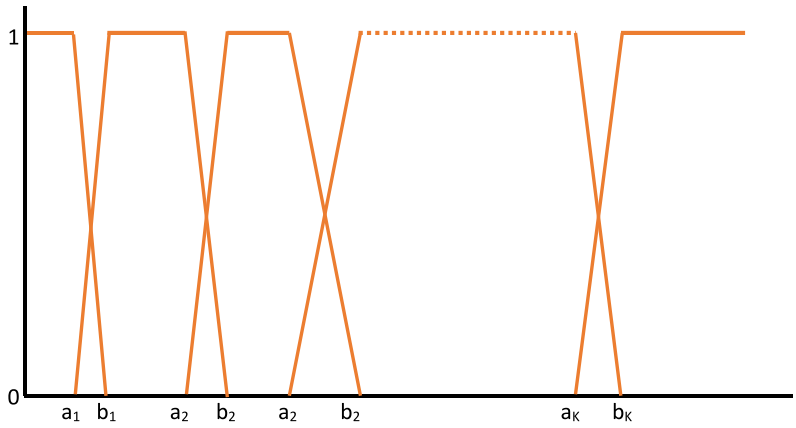

Fig. 7. Fuzzy membership functions for K+1 segments.

Table 2
Parameter settings of BDTPSO.

Parameter	Value
w	Linearly decreasing form 0.9 to 0.3
τ	0.2
m	7
C^{MIN}	0.1
C^{MAX}	1.5 for C_i and C_i^1 in Eq. (5) and Eq. (7), respectively. 2.7 for C_i^2 in Eq. (7)
N	64
$MaxE$	4

set. In this paper, BDTPSO is tested using swarm sizes which are exact powers of two. The algorithm can be extended to operate with any number of particles; However, describing this extension is unnecessary. The swarm size has a profound impact on the performance of swarm optimization algorithms. Small swarms can rapidly lose diversity, while large swarms cannot evolve sufficiently in time. Usually, a swarm of 40 particles is suggested for problems with dimensionalities tested in this paper. Accordingly, a swarm of 32 or 64 particles seems to be a reasonable choice for BDTPSO.

Fig. 8 gives the average obtained results of BDTPSO on several problems while using various swarm sizes. The performance of BDTPSO deteriorates significantly when its swarm reduces from 32 to 16. Small swarms can easily lose their diversities on the complex problems of CEC2013 and get trapped in poor local optima. The quality of results on f_2, f_{11}, and f_{15} decreases as N increases from 64 to 256. In contrast, the quality of results on f_7 and f_{12} improves by increasing the swarm size. Increasing the swarm size from 64 to 256 slightly improves the performance of BDTPSO on f_7 and f_{12} by providing a well-diversified population. These problems are multi-modal functions with lots of local optima and complex landscapes. In contrast, the quality of the results of the algorithm on f_2, f_{11}, and f_{15} decreases with N increasing from 64 to 256. BDTPSO can obtain better results on these problems by evolving a medium or small size swarm for more iterations. The proposed algorithm operates well on most benchmarks when its swarm size lies between 32 and 128. Finally, considering the overall performance of BDTPSO on several tested problems, $N = 64$ is employed for the rest of this paper.

6.1.2. The impact of MaxE on the performance of BDTPSO

This section investigates the effect of changes in $MaxE$ on the performance of BDTPSO. This parameter defines the maximum number of non-self attractors that an even particle can peruse. Each even particle can have at most $N/(2l\text{-}1)$ exemplars from the lth level. For instance, with $N = 8$, an even particle can follow not more than eight other particles from the fourth level. Using large $MaxE$ values only impedes the learning process, as the learning procedure searches for a suitable number of exemplars among infeasible values. Also, using too many exemplars have several unpleasant effects on the search process [34]. Our experimental studies indicate that the setting $MaxE = 4$ works well on most problems.

Fig. 9 shows the results of BDTPSO on several problems, with $MaxE$ changing between 2 and 7. BDTPSO obtains better results on all problems, except f_{15}, with $MaxE$ settings greater than two. With $MaxE = 2$, even particles can use at most two exemplars besides their own historical best positions. Consequently, their behavior becomes closer to canonical particles. The performance of BDTPSO deteriorates slowly on most problems as $MaxE$ increases from 5 to 7. Using too many exemplars can confound even particles. However, the adaptive mechanism slowly learns to use an appropriate number of exemplars. Hence, the performance deterioration due to increasing $MaxE$ is not significant.

6.1.3. The impact of selection rate on the performance of BDTPSO

This section investigates the sensitivity of the proposed approach to its selection rate (τ). This parameter determines the ratio of particles that can be considered promising. The value of the selection rate lies between 0 and 1. Fig. 10 shows how the performance of BDTPSO is affected by using various settings for τ. As it is clear, the proposed approach is not sensitive to changes in this parameter. In fact, we can simply omit it from the proposed method by using $\tau = 1$. This insensitiveness is mainly because of the following two reasons. First, only particles with improvement degrees higher than zero can be selected as promising particles. These particles potentially possess favorable parameters, which can be beneficial to the other particles. Second, usually, a small portion of the swarm improves at each iteration. Hence, no matter how large is the used selection rate, the number of promising particles is usually small. However, to increase the flexibility of the algorithm, we retain this parameter. Considering the obtained results, BDTPSO performs slightly better on f_2 and f_{11} when τ is between 0.1 and 0.25. Accordingly, $\tau = 0.2$ is suggested in Table 2.

6.1.4. The impact of refreshing gap on the performance of BDTPSO

This section investigates the impact of the refreshing gap on the performance of the proposed method. The value of this parameter lies between 1 and the maximum number of iterations. Increasing the refreshing gap reduces the number of neighborhood adjustments that can be performed during the search. Accordingly, the refreshing gap should be a small fraction of the maximum number of iterations. If the number of allowable fitness

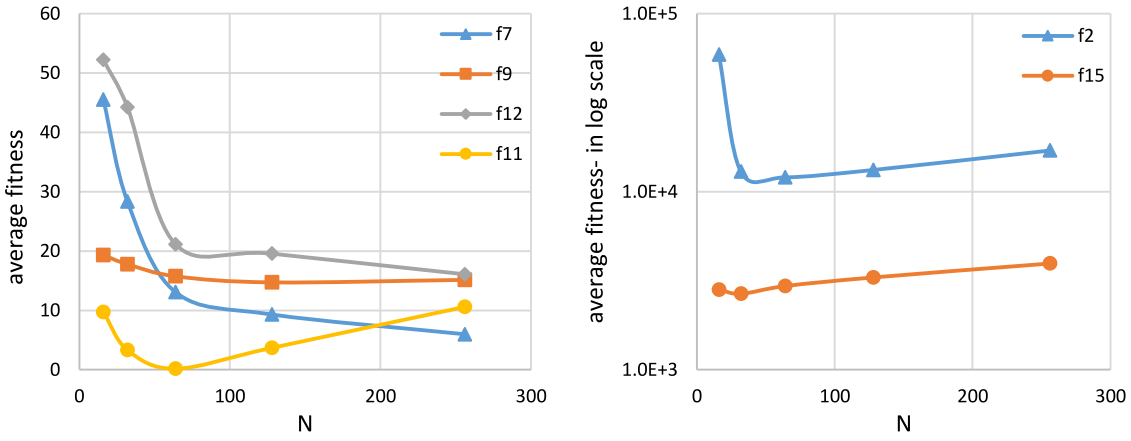

Fig. 8. The average fitness performance of BDTPSO on several problems using various swarm sizes.

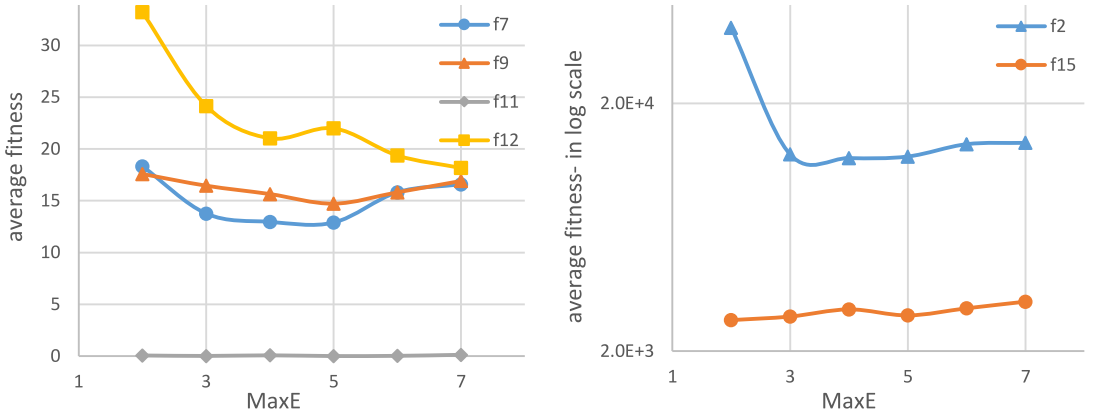

Fig. 9. The average fitness of BDTPSO on a set of benchmarks from CEC2013 with MaxE changing between 2 and 7.

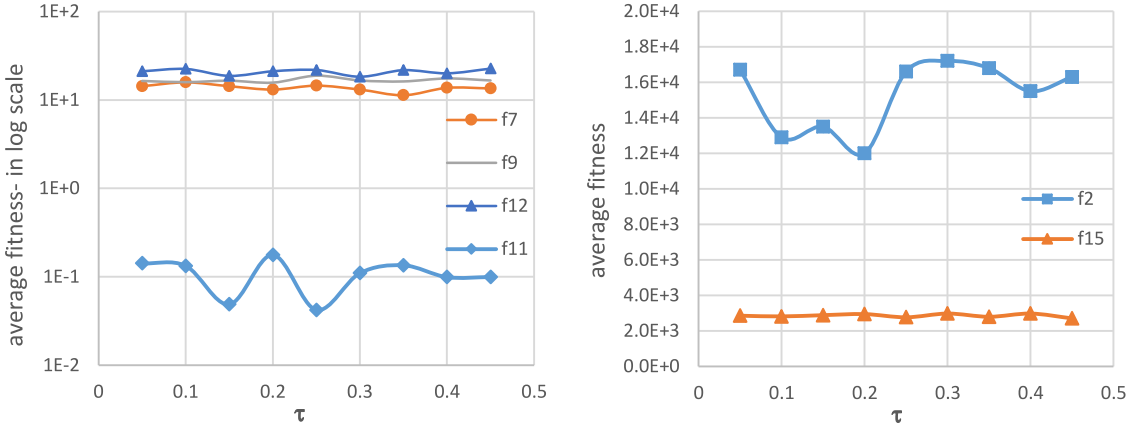

Fig. 10. The effect of selection rate on the results of BDTPSO.

evaluations, and consequently the number of iterations, is small (see Section 6.6), then the refreshing gap should be kept small as well. The suggested setting for this parameter is $m = 7$ [14]. Fig. 11 provides the results of BDTPSO on some of the benchmarks when its refreshing gap changes between 3 and 12. Considering the obtained results, the proposed algorithm is not sensitive to small changes in the refreshing gap. The performance of BDTPSO on f_2 and f_7 slightly improves with increasing the refreshing gap. However, the algorithm shows an opposite behavior on f_9. On the other three benchmarks, the changes in the results due to using various settings of the refreshing gap are neglectable.

6.1.5. The impact of acceleration coefficients on BDTPSO

Uneven particles are updated using Eq. (6), which has one acceleration coefficient. Even particles use two acceleration coefficients according to Eq. (9). All these acceleration coefficients are adaptively generated between C^{MIN} and C^{MAX} for each particle. Since c^1 in Eq. (9), like c in Eq. (6), is applied to a term with a single exemplar, we use identical C^{MAX} values for generating both of them. However, c^2, like the constriction factor of FIPS (χ in [34]), is applied to the sum of several terms and exemplars. Hence, a different upper bound is employed for generating social weights for even particles. Suggested acceleration coefficient values in

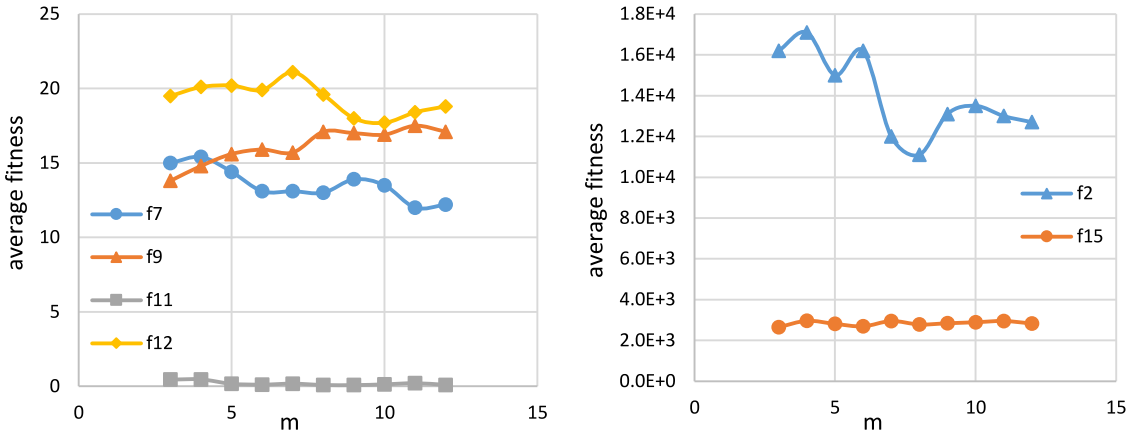

Fig. 11. The effect of refreshing gap on the results of BDTPSO.

the literature usually lie between 0 and 4.1. Accordingly, a small nonnegative C^{MIN} like 0.1 is employed in this paper. C^{MAX} can be set easily to 4.1, which allows the adaptive mechanism to search for suitable acceleration coefficients in a wide range of values. However, we can obtain better results by using better upper bounds. Two of the most popular settings for acceleration coefficients are $c = 1.5$ and $c = 2$. $c = 1.5$ ($c = 1.49445$ to be precise) is also suggested for CLPSO, which uses an updating rule similar to that of uneven particles. The recommended setting for χ in the FIPS is $\chi = 4.1$. According to the updating rules of even particles, we can assume that $\chi = c^1 + c^2$. Therefore, if we use $c^1 = 1.5$, $c^2 = 2.6$ would match the suggested settings of FIPS. These recommended settings can give some intuition in finding proper upper bounds for generating acceleration coefficients. In fact, experimental studies demonstrate that $C^{MAX} = 1.5$ is a good choice for uneven particles. Also, it is a good choice for generating cognition weights for even particles. Moreover, $C^{MAX} = 2.7$ works well for generating social weights for even particles. The rest of this section investigates the sensitivity of BDTPSO to *CMAX*.

Fig. 12 shows the results of BDTPSO when the upper bound for defining c in Eq. (6) and c^1 in Eq. (9) changes. The proposed method performs better on f_7 and f_{11} when it uses a large C^{MAX}, like $C^{MAX} = 3$. This observation indicates that using large acceleration coefficients can result in a better performance on these problems. But, the adaptive procedure cannot generate sufficiently large acceleration coefficients whenever C^{MAX} is not large enough. The performance of the proposed method slightly gets worse when C^{MAX} increases from 0.9 to 3. This decline in performance suggests that BDTPSO performs better on these benchmarks with small acceleration coefficients. With large C^{MAX}, the adaptive mechanism faces a bigger parameter space and requires more effort to find suitable settings for these problems. Moreover, changing C^{MAX} within the designated range does not affect the performance of the proposed approach on f_2 and f_9. Finally, while the best setting of C^{MAX} varies for different problems, with a large enough C^{MAX} the adaptive mechanism can always find appropriate acceleration coefficients for the particles. Hence, the decline in the performance of BDTPSO on some problems due to using large C^{MAX} values is neglectable.

Fig. 13 shows the results of the proposed method when the maximum bound for defining c^2 in Eq. (9) changes between 1 and 4. Increasing C^{MAX} for generating the social weights of even particles improves the performance of BDTPSO on problems like f_{11} and f_{12}. The adaptive mechanism can choose a large range of social weights for each even particle with large C^{MAX} values. However, with large C^{MAX} values, particles may use large social weights excessively on some benchmarks such as f_{12}. This excessive usage of large social weights can lead to fast diversity loss and entrapment in poor local optima.

Table 3
The summary of comparison results of BDTPSO with other peer methods based on Wilcoxon rank sum test.

	Unimodal	Basic Multimodal	Composition	All
MCUPS	3/2/0	14/1/0	6/2/0	23/5/0
DTTPSO	3/2/0	13/1/1	6/2/0	22/5/1
EPSO	3/2/0	10/2/3	4/2/2	17/6/5
HCLPSO	3/2/0	8/2/5	4/2/2	15/6/7

6.2. Experiments on 30-D CEC2013 problems

The experiments of this section are conducted on the 30-D instances of the CEC2013 benchmarks. Each algorithm is tested 51 times on each function for obtaining statistical results, and each run of an algorithm on a function is terminated after 3E+5 fitness evaluations. The Freidman ranks of the compared methods based on the average obtained results are provided in Figs. 15 to 18. Additionally, in order to compare different methods from a statistical point of view, the Wilcoxon rank-sum test with a confidence level of 95% [48] is employed. The summary of comparison results according to this test is provided in Table 3. The results in this table are presented in the form "$w/t/l$" meaning that BDTPSO is significantly better than, equal to, or worse than the corresponding compared methods on w, t, and l functions, respectively. The convergence curves of the compared methods on some of the benchmarks are provided in Fig. 14. Also, the detailed statistical results are given in Appendix A.

BDTPSO performs statistically better than or equal to the compared methods on the unimodal benchmarks. Both f_1 and f_5 are separable functions and easily solvable. Hence, all compared methods performed equally well on these functions. However, BDTPSO obtained better results on the remaining challenging and non-reparable unimodal benchmarks. Although these problems are unimodal, they have landscapes with complex properties, providing challenging features, especially for PSO methods.

The multimodal problems except f_{11}, f_{14}, and f_{17} are non-separable and rotated, and they usually possess a tremendous number of local optima. The proposed method outperforms MCUPS and DTTPSO on almost all multimodal problems. Besides f_8 and f_{15}, the obtained results of BDTPSO on the multimodal problems are noticeably better than those obtained by MCUPS and DTTPSO. BDTPSO also has overall better results on the multimodal benchmarks in comparison to EPSO and HCLPSO. Both EPSO and HCLPSO are able to find the global optimum of Rastrigin's function (f_{11}). BDTPSO is also capable of reaching the global optimum of this problem in most of its executed runs, and it demonstrates a very competitive performance. The performance

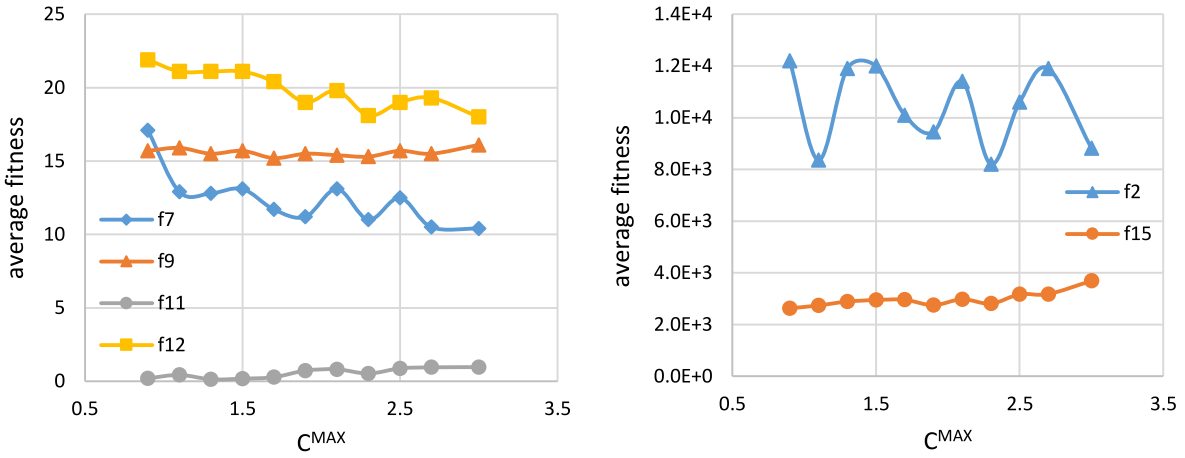

Fig. 12. The effect of C^{MAX} on the performance of BDTPSO.

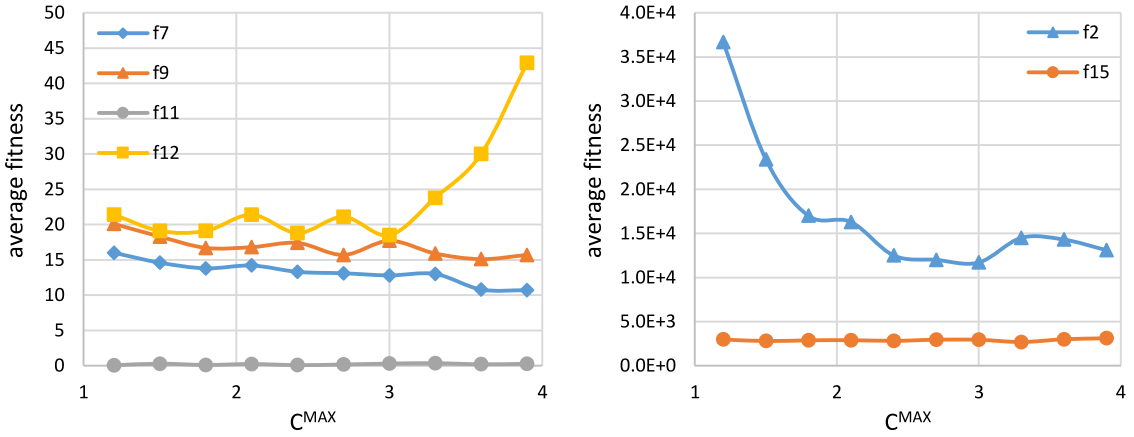

Fig. 13. The effect of CMAX for social weights of even particles on the performance of BDTPSO.

of BDTPSO on the rotated versions of f_{11}, i.e. f_{12} and f_{13}, which are remarkably harder than f_{11}, is considerably better than the other compared algorithms. Similarly, while EPSO and HCLPSO outperform BDTPSO on the Schwefel's function (f_{14}), BDTPSO is able to obtain better results on the rotated version of f_{14}, i.e. f_{15}. BDTPSO also obtained the best results on the rotated Lunacek bi-Rastrigin function (f_{18}), while HCLPSO has the best performance on its non-rotated version, i.e. f_{17}. This general superiority of BDTPSO is due to its better explorative abilities. BDTPSO can effectively exploit variable dependencies existing in inseparable problems via the adaptive mechanisms of even particles. It can reach a balanced explorative and exploitative behavior via this adaptation and the cooperation of even and uneven particles.

BDTPSO was able to yield better results on the composition problems. Except for f_{22}, all composition benchmarks are rotated and have inseparable variables. These benchmarks are formed by combining several functions and present challenging optimization tasks. Each constituent function has its landscape patterns and characteristics. Hence, the properties of each composition problem vary greatly over its landscape, and the landscape has different patterns in different areas. These varying characteristics make it hard for learning mechanisms to adapt. The proposed method was capable of achieving the best results on six out of eight composition problems while reaching comparable solutions on the reminder (see Appendix A for the detailed results). HCLPSO and EPSO obtained better results than BDTPSO on f_{21} and f_{22}. f_{21} is composed of four unimodal functions, f_1, f_3, f_4, and f_5, and one multimodal function, f_6. Accordingly, it has several local optima, with different properties around its different optima. While the proposed method is not the best algorithm for this problem, it obtained acceptable and better solutions than MCUPS and DTTPSO. f_{22} is composed of several instances of the Schwefel's function with different parameters, and, similar to f_{14}, the proposed method obtained close but weaker results than HCLPSO and EPSO on f_{22}. However, BDTPSO outperformed MCUPS and DTTPSO with a large margin on this benchmark.

Fig. 14 shows the convergence curves of the compared methods on a set of problems. These selected benchmarks include some unimodal, multimodal, and composition functions with various characteristics. On all of these benchmarks, the proposed algorithm shows an eligible convergence rate. It approaches appropriate solutions at a desirable pace. On f2, the proposed algorithm reaches solutions better than the other compared methods nearly after 1/3 of their fitness evaluation budget. The convergence rate and the solution accuracy of BDTPSO on f27 are also distinctive. It found and converged to better solutions in less than a quarter of the evolution budget designated to the other peer algorithms. A similar discussion also holds for f9. If we move along the FEs axis, the average fitness of BDTPSO at each point of the axis is usually better than the average fitness of MCUPS at that point. The same statement is true when we compare BDTPSO with other peer methods. For instance, except for some intervals in the plots related to f22 and f7, the average fitness curves of BDTPSO are generally better than the corresponding fitness

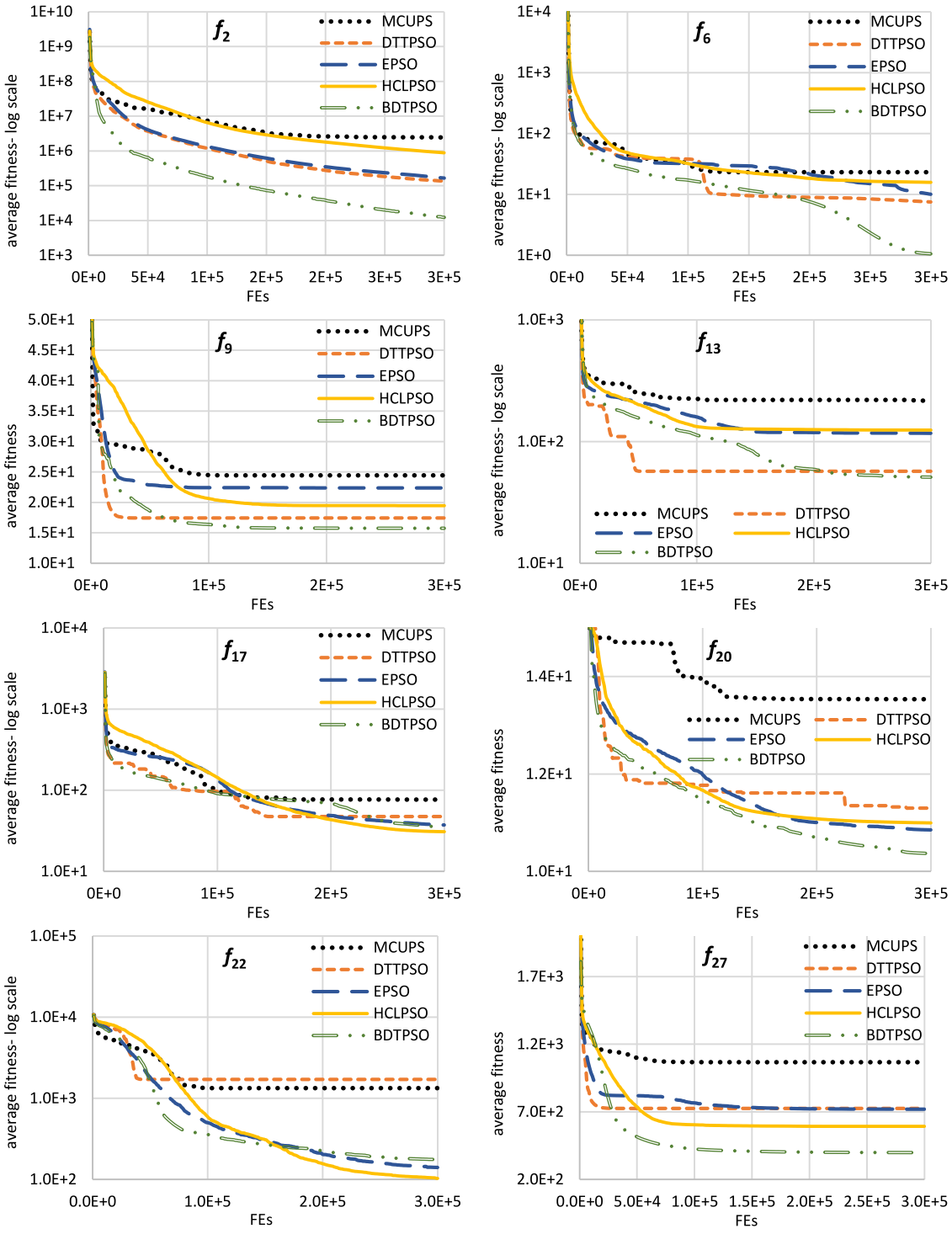

Fig. 14. Convergence curves of the compared methods on f_2, f_6, f_9, f_{13}, f_{17}, f_{20}, f_{23}, and f_{27}.

curves of EPSO at each point. Considering these observations and the statistical results from Table 3, we can postulate the following. The proposed algorithm is able to converge to better solutions at a pace better than the other compared algorithms on a majority of problems. Although BDTPSO is not the fastest compared method on all problems, it can usually reach solutions with specific fitness values using less evaluation budget than the other methods. Consequently, the proposed method can be used for solving optimization problems in cases where the evaluation budget is very limited.

6.3. Experiments on 50-D CEC2013 problems

This section investigates the performance of the proposed approach on the 50-dimensional instances of the CEC2013 benchmarks. The results of each method on each problem are obtained over 51 independent runs. Each run is terminated after 5E+5 fitness evaluations. The Freidman ranks according to the average errors of the compared methods are given in Figs. 19 to 22. The detailed results are provided in Appendix B. With few exceptions, the comparison results are similar to those reported

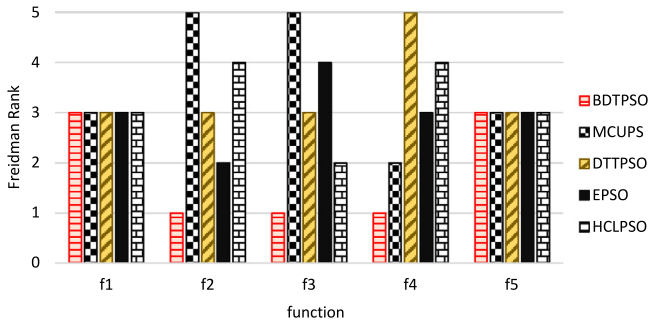

Fig. 15. The Freidman ranks of the compared methods on the unimodal functions.

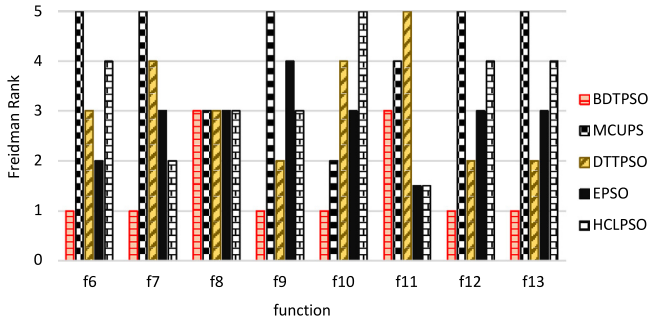

Fig. 16. The Freidman ranks of the compared methods on the Basic Multimodal Functions f_6–f_{13}.

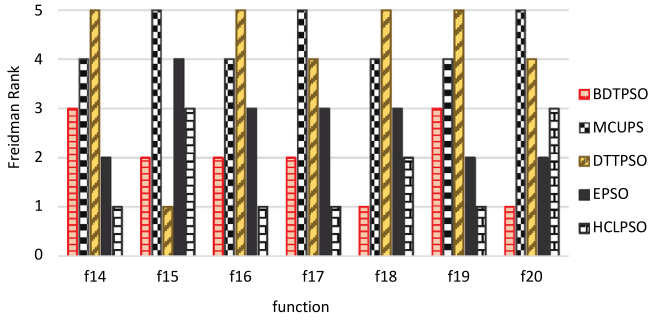

Fig. 17. The Freidman ranks of the compared methods on the Basic Multimodal Functions f_{14}–f_{20}.

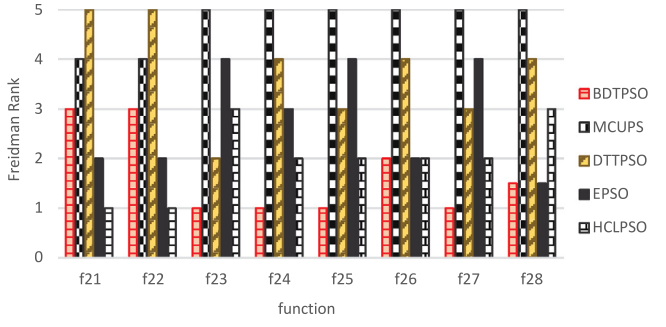

Fig. 18. The Freidman ranks of the compared methods on the Composition Functions.

on 30-dimensional problems. The proposed approach is the best performing algorithm on the unimodal problems, f_1–f_5.

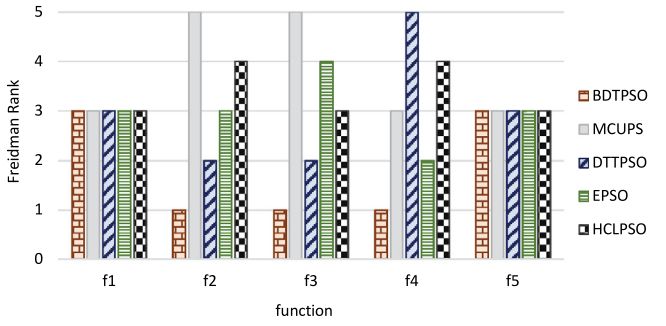

Fig. 19. The Freidman ranks of the compared methods on the 50 dimensional unimodal functions.

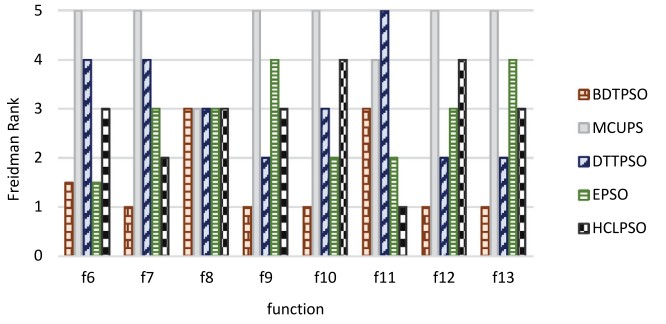

Fig. 20. The Freidman ranks of the compared methods on the 50 dimensional Basic Multimodal Functions, f_6–f_{13}.

The proposed approach is the best performing algorithm on the unimodal problems, f_1–f_5. On the multimodal benchmarks, f_6–f_{20}, the overall performance of BDTPSO is better than the other compared PSO algorithms. It has the best results on the rotated Rosenbrock's function (f_6). However, while BDTPSO is able to reach the global optimum of the 30D f_6, it is unable to escape from the local optimum of the 50D instance of f_6 in most of its runs. This benchmark has only two optima. A very narrow valley connects the local optimum to the global one, which makes it extremely hard for the algorithms to find their ways towards the global optimum. The proposed approach has the best average results on f_7–f_{10}. HCLPSO and EPSO obtained better results on the Rastrigin's function, similar to the previous section. However, BDTPSO is superior to the other peer PSO methods on the rotated versions of this function. DTTPSO has the best performance on the rotated Schwefel (f_{15}), while BDTPSO has a close performance on the benchmark. The proposed algorithm obtained the second-best results on f_{16} and f_{17}, just behind HCLPSO, and the best results on f_{18} and f_{20}.

The comparison results on the 50D composition functions are also similar to the results obtained on their 30D instances. BDTPSO obtained the best average results on five benchmarks, f_{23}-f_{27}, and fell behind HCLPSO and EPSO on three cases. HCLPSO and EPSO yielded superior results on f_{21} and f_{22} in comparison to the other compared algorithms. The result of BDTPSO on f_{21} is close to the results of MCUPS and DDTPSO on f_{21}. The proposed method is superior to the two algorithms with a large margin on f_{22}. The performance of BDTPSO is very close to EPSO and HCLPSO on f_{28}, where BDTPSO has fallen behind EPSO and HCLPSO in a few runs, only.

6.4. Investigating the running time BDTPSO

This section experimentally compares the running time of different peer PSO methods on a set of problems from the CEC2013

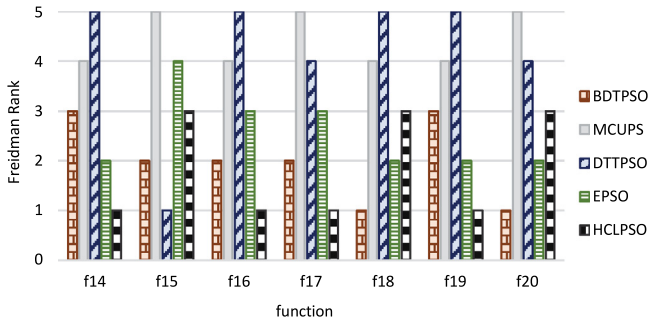

Fig. 21. The Freidman ranks of the compared methods on the 50 dimensional Basic Multimodal Functions, f_{14}–f_{20}.

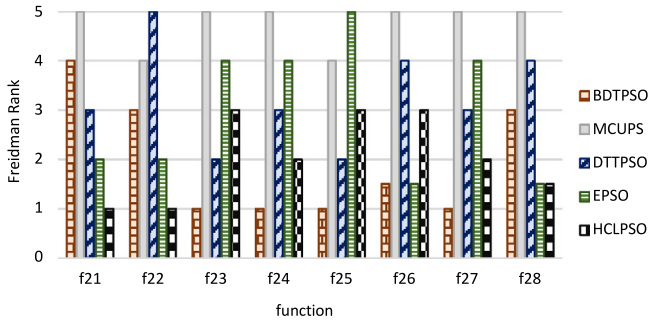

Fig. 22. The Freidman ranks of the compared methods on the 50 dimensional Composition Functions.

Table 4
The average running time of different compared methods on several benchmark problems (in seconds).

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
MCUPS	4.00	4.75	4.63	4.20	3.60	4.64	6.18	5.90	15.35	4.40
DTTPSO	9.97	10.78	10.51	10.27	10.28	10.72	12.70	11.54	20.77	10.03
BDTPSO	11.32	12.53	12.03	11.77	11.45	11.95	13.83	12.37	22.28	11.42
HCLPSO	12.13	13.32	13.26	12.45	11.17	14.52	14.80	15.33	24.03	12.52
EPSO	17.13	19.28	17.53	20.83	16.88	19.91	21.35	36.81	32.66	17.77

benchmark set. The comparison results are given in Table 4. We can arrange the compared methods from the fastest to the slowest as follows: MCUPS, DDTPSO, BDTPSO, HCLPSO, and EPSO. MCUPS has the lowest running times among the compared algorithms. The proposed method is the third fastest method, just slightly behind HCLPSO.

MCUPS has the best running times. However, the accuracy of its obtained solutions is generally lower than EPSO, HCLPSO, and BDTPSO (see Section 6.2). BDTPSO obtains better solutions compared to DTTPSO while its running times are close to BDTPSO. BDTPSO can still demonstrate better performance in comparison to EPSO and HCLPSO with lesser computation costs. Putting all this together, it can be inferred that BDTPSO is an efficient algorithm in terms of both accuracy and computational cost.

6.5. Antenna array design

In this section, BDTPSO is employed for designing two antenna arrays with 15 and 20 elements. It is assumed that the amplitudes are normalized and fall in the range [0,1]. Also, the distances between consecutive elements can change from 0.5λ to 1λ. As stated previously, each synthesized array should comply with a predefined first null bandwidth. In this experiment, the FNBW corresponding to a uniform circular antenna array of size N, with unit amplitudes and 0.5λ between element spacing, is used for designing an array with N elements.

Table 5
Statistical results of the obtained solutions of the compared methods on the antenna array synthesis problem with different number of elements.

NE	St	BDTPSO	CLPSO	CSO	FFO
15	Mean	**3.56E−01**	4.49E−01	6.65E−01	3.95E−01
	Min	**3.51E−01**	3.97E−01	4.40E−01	3.53E−01
	Std	**3.00E−03**	2.93E−02	1.07E−01	4.45E−02
	P-value		(+)3E−11	(+)3E−11	(+)6E−07
20	Mean	**4.81E−01**	5.65E−01	6.59E−01	5.09E−01
	Min	**4.63E−01**	5.22E−01	5.22E−01	4.82E−01
	Std	**6.38E−03**	2.02E−02	8.30E−02	1.65E−02
	P-value		(+)3E−11	(+)3E−11	(+)6E−10

Table 6
The average results of different methods on the antenna synthesis problems according to various design objectives.

NE	St	BDTPSO	CLPSO	CSO	FFO
15	f_{SLA}	**1.40E−01**	1.61E−01	2.52E−01	1.56E−01
	f_{MSL}	**2.17E−01**	2.68E−01	3.92E−01	2.38E−01
	FNBW	**3.70E+01**	**3.70E+01**	**3.70E+01**	**3.70E+01**
20	f_{SLA}	**1.72E−01**	1.98E−01	2.42E−01	1.83E−01
	f_{MSL}	**3.10E−01**	3.57E−01	4.00E−01	3.27E−01
	FNBW	**2.76E+01**	2.77E+01	2.78E+01	**2.76E+01**

Note. The array factor of the uniform antenna array with N elements can be obtained using Eq. (11). The minimums of this array factor on two sides of the main lobe represent the location of $NULL_1$ and $NULL_2$. The desired FNBW for $N = 15$ is 3.70E+1 and for $N = 20$ is 2.76E+1.

Note. Each position vector is a $2N$-dimensional vector with components corresponding to $d_1, d_2, \ldots, d_N, I_1, I_2, \ldots$, and I_N. This vector represents a configuration for an antenna array with N elements. The array factor of this antenna array for different angles can be calculated using Eq. (11). In order to calculate the fitness value of a position vector, the array factor is sampled along the azimuth angle. Using this sampled array factor, f_{NU}, f_{SLA}, and f_{MSL} can be computed easily. The fitness of a position vector is the sum of these three values (Eq. (15)).

The obtained results of BDTPSO are compared with those of CSO [35], FFO [37], and CLPSO [36]. The results of each method on each antenna design problem are obtained over 30 independent runs, and each run is terminated after 1.5E+5 fitness evaluations. The statistical results, in terms of the average, minimum, and standard deviation of solution costs, are given in Table 5. Also, BDTPSO is compared with each peer method using the Wilcoxon rank-sum test for a confidence level of 95%, and the obtained *P*-values are given in the table. In this table, '+' denotes the statistical superiority of BDTPSO to a compared algorithm according to the Wilcoxon test, '−' indicates the opposite case, and '=' indicates that the two compared methods are statistically indistinguishable. The cost function in Eq. (15) has three design objectives, which are formalized as f_{SLA}, f_{MSL}, and f_{NU}. f_{SLA} encodes the average side lobe level, and f_{MSL} encodes the sum of the maximum side lobes in the lower and upper bands. FNU is a term employed to enforce the desired FNBW. Table 6 provides the average f_{SLA}, f_{MSL}, and FNBW of the obtained antenna arrays. Also, Fig. 23 shows the radiation patterns of the synthesized antenna arrays with the best and median fitness values. The configurations of the antenna arrays synthesized by BDTPSO are given in Appendix C.

The statistical results demonstrate the superiority of the proposed approach for solving antenna array design problems. BDTPSO has the best average fitness values on both of the antenna design problems. It also obtained the best antennas according to the design objectives. The Wilcoxon rank-sum test results show the superiority of the proposed algorithm statistically. In summary, the proposed approach has a robust performance and can

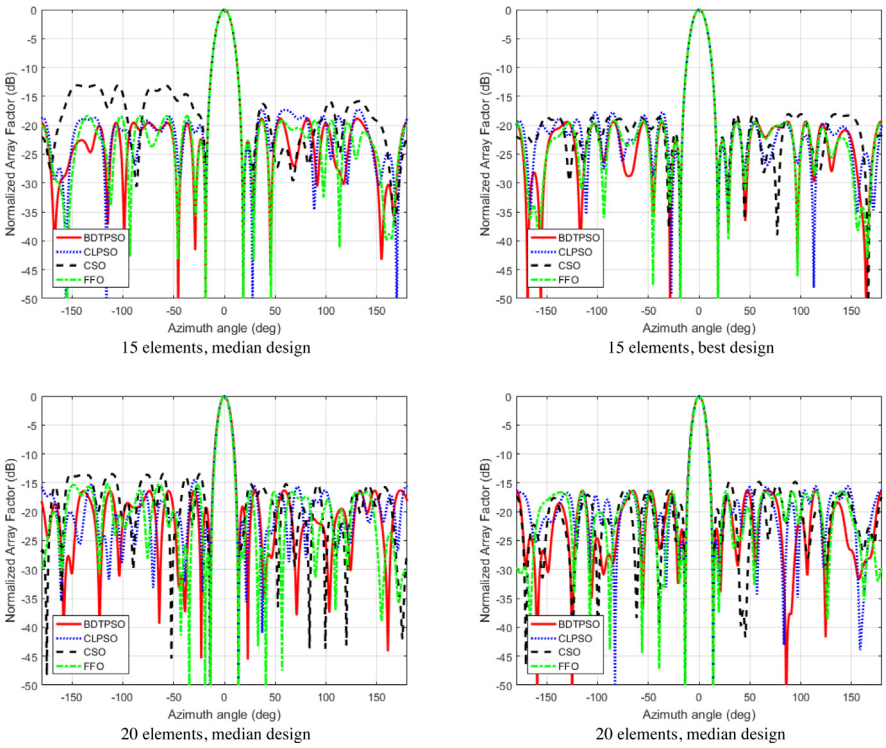

Fig. 23. Normalized radiation patterns for the circular antenna arrays with 15 and 20 elements, median and best cases.

Table 7
Statistical results of the compared algorithms on several images in terms of average and standard deviation of the fitness values along with Wilcoxon rank-sum tests and p-values. Number of thresholds = 3.

Image ID	107014	118072	15062	179084	189029	3063	326025	43033	71099	94095
BDTPSO	**1.750E+01** 2.348E−04	**1.731E+01** 6.419E−04	**1.835E+01** 3.613E−15	**1.717E+01** 7.227E−15	**1.588E+01** 5.420E−15	**1.752E+01** 1.119E−04	**1.800E+01** 7.227E−15	**1.700E+01** 6.716E−06	**1.695E+01** 3.939E−03	**1.745E+01** 3.225E−05
CLPSO	1.746E+01 2.844E−02 (+)1E−11	1.727E+01 1.960E−02 (+)3E−11	1.826E+01 4.199E−02 (+)1E−12	1.707E+01 4.622E−02 (+)1E−12	1.578E+01 3.934E−02 (+)1E−12	1.749E+01 1.061E−02 (+)1E−11	1.794E+01 3.229E−02 (+)1E−12	1.696E+01 1.279E−02 (+)1E−11	1.688E+01 2.404E−02 (+)2E−11	1.740E+01 1.945E−02 (+)2E−11
CSO	**1.750E+01** 2.318E−02 (=)1E−01	**1.731E+01** 6.398E−03 (=)4E−01	**1.835E+01** 2.870E−03 (+)1E−03	**1.717E+01** 2.884E−03 (+)2E−02	**1.588E+01** 1.129E−02 (+)6E−03	**1.752E+01** 5.013E−03 (+)2E−03	1.799E+01 4.268E−03 (+)3E−05	**1.700E+01** 1.809E−03 (+)2E−02	1.693E+01 3.901E−02 (+)4E−03	1.744E+01 6.838E−03 (+)5E−05
FFO	1.733E+01 4.980E−02 (+)1E−11	1.717E+01 4.054E−02 (+)3E−11	1.814E+01 7.542E−02 (+)1E−12	1.688E+01 4.707E−02 (+)1E−12	1.557E+01 6.285E−02 (+)1E−12	1.741E+01 4.715E−02 (+)1E−11	1.783E+01 5.842E−02 (+)1E−12	1.681E+01 5.696E−02 (+)1E−11	1.678E+01 4.789E−02 (+)2E−11	1.728E+01 5.303E−02 (+)2E−11

Table 8
Statistical results of the compared algorithms on several images in terms of average and standard deviations of the fitness values along with Wilcoxon rank-sum tests and p-values. Number of thresholds = 9.

Image	107014	118072	15062	179084	189029	3063	326025	43033	71099	94095
BDTPSO	**3.619E+01** 1.709E−02	**3.574E+01** 1.888E−02	**3.672E+01** 5.014E−02	**3.521E+01** 4.200E−02	**3.435E+01** 1.811E−02	**3.608E+01** 2.103E−02	**3.663E+01** 3.674E−02	**3.457E+01** 1.350E−02	**3.473E+01** 2.687E−02	**3.567E+01** 1.362E−02
CLPSO	3.526E+01 1.537E−01 (+)3E−11	3.484E+01 1.808E−01 (+)3E−11	3.577E+01 1.644E−01 (+)3E−11	3.422E+01 1.271E−01 (+)3E−11	3.355E+01 1.567E−01 (+)3E−11	3.498E+01 1.863E−01 (+)3E−11	3.565E+01 1.737E−01 (+)3E−11	3.362E+01 1.895E−01 (+)3E−11	3.366E+01 1.734E−01 (+)3E−11	3.470E+01 1.619E−01 (+)3E−11
CSO	3.595E+01 1.568E−01 (+)5E−10	3.550E+01 1.881E−01 (+)9E−11	3.651E+01 1.561E−01 (+)2E−07	3.489E+01 2.098E−01 (+)6E−10	3.414E+01 1.196E−01 (+)8E−10	3.581E+01 1.637E−01 (+)3E−11	3.630E+01 2.383E−01 (+)5E−09	3.433E+01 1.187E−01 (+)3E−11	3.433E+01 2.543E−01 (+)3E−11	3.543E+01 1.603E−01 (+)3E−11
FFO	3.426E+01 2.161E−01 (+)3E−11	3.396E+01 3.027E−01 (+)3E−11	3.487E+01 2.131E−01 (+)3E−11	3.321E+01 2.076E−01 (+)3E−11	3.249E+01 1.874E−01 (+)3E−11	3.411E+01 2.343E−01 (+)3E−11	3.480E+01 1.796E−01 (+)3E−11	3.246E+01 3.542E−01 (+)3E−11	3.263E+01 3.453E−01 (+)3E−11	3.370E+01 3.045E−01 (+)3E−11

be successfully employed for designing various antenna arrays. We can run each optimization algorithm just once and use the obtained solution, or we can run the algorithm several times and pick the best solution. In either case, the proposed algorithm outperforms the other peer methods and can be utilized positively.

Table 6 indicates that with optimizing Eq. (15), we can achieve all defined design objectives. All algorithms are able to find antenna designs with the defined FNBWs. However, FFO and BDTPSO can attain this objective more accurately. BDTPSO also yields the best performances according to the two other design objectives. It obtained the lowest sum of the maximum side lobe

Table 9
Statistical results of the compared algorithms on several images in terms of average and standard deviations of the fitness values along with Wilcoxon rank-sum tests and p-values. Number of thresholds = 15.

Image	107014	118072	15062	179084	189029	3063	326025	43033	71099	94095
BDTPSO	**5.075E+01**	**5.028E+01**	**5.121E+01**	**4.938E+01**	**4.904E+01**	**5.108E+01**	**5.119E+01**	**4.822E+01**	**4.844E+01**	**4.985E+01**
	2.911E−02	3.881E−02	6.028E−02	4.950E−02	3.913E−02	8.726E−02	6.395E−02	3.416E−02	6.707E−02	5.295E−02
CLPSO	4.842E+01	4.799E+01	4.903E+01	4.718E+01	4.703E+01	4.865E+01	4.900E+01	4.536E+01	4.601E+01	4.735E+01
	2.827E−01	2.986E−01	3.371E−01	2.928E−01	2.904E−01	3.064E−01	3.502E−01	3.659E−01	3.398E−01	3.042E−01
	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11
CSO	4.992E+01	4.939E+01	5.049E+01	4.871E+01	4.813E+01	5.019E+01	5.032E+01	4.719E+01	4.756E+01	4.888E+01
	3.883E−01	3.319E−01	3.124E−01	2.361E−01	3.691E−01	4.555E−01	3.859E−01	3.473E−01	3.771E−01	5.278E−01
	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)6E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11
FFO	4.638E+01	4.599E+01	4.701E+01	4.521E+01	4.492E+01	4.673E+01	4.727E+01	4.230E+01	4.372E+01	4.518E+01
	3.829E−01	4.102E−01	4.579E−01	5.200E−01	4.671E−01	4.904E−01	2.918E−01	7.387E−01	6.018E−01	5.481E−01
	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11	(+)3E−11

Table 10
The average results according to FSIM.

TH	Image	107014	118072	15062	179084	189029	3063	326025	43033	71099	94095
3	BDTPSO	5.85E−01	6.31E−01	**6.93E−01**	**7.58E−01**	**7.53E−01**	**7.04E−01**	**6.86E−01**	5.65E−01	**7.00E−01**	5.62E−01
	CLPSO	**5.88E−01**	6.33E−01	6.92E−01	**7.58E−01**	**7.53E−01**	6.97E−01	6.85E−01	**5.68E−01**	6.86E−01	**5.78E−01**
	CSO	5.86E−01	6.31E−01	**6.93E−01**	**7.58E−01**	**7.53E−01**	**7.04E−01**	**6.86E−01**	5.65E−01	6.86E−01	5.67E−01
	FFO	5.87E−01	**6.37E−01**	6.88E−01	7.54E−01	7.51E−01	7.02E−01	6.79E−01	5.59E−01	6.77E−01	5.72E−01
9	BDTPSO	**8.43E−01**	**9.00E−01**	**9.04E−01**	**8.69E−01**	**9.02E−01**	**8.23E−01**	**8.81E−01**	**8.43E−01**	**9.00E−01**	**9.06E−01**
	CLPSO	8.32E−01	8.87E−01	8.87E−01	8.62E−01	8.93E−01	8.10E−01	8.72E−01	8.27E−01	8.92E−01	8.90E−01
	CSO	8.40E−01	8.91E−01	8.91E−01	8.62E−01	8.97E−01	8.16E−01	8.72E−01	8.33E−01	8.87E−01	8.98E−01
	FFO	8.27E−01	8.70E−01	8.75E−01	8.53E−01	8.85E−01	8.01E−01	8.62E−01	8.07E−01	8.84E−01	8.73E−01
15	BDTPSO	**9.27E−01**	**9.60E−01**	**9.52E−01**	**9.19E−01**	**9.49E−01**	**8.88E−01**	**9.44E−01**	**9.26E−01**	**9.45E−01**	**9.59E−01**
	CLPSO	9.10E−01	9.40E−01	9.40E−01	9.10E−01	9.36E−01	8.74E−01	9.30E−01	9.07E−01	9.36E−01	9.39E−01
	CSO	9.17E−01	9.46E−01	9.44E−01	9.14E−01	9.42E−01	8.80E−01	9.28E−01	9.08E−01	9.35E−01	9.45E−01
	FFO	8.94E−01	9.23E−01	9.33E−01	9.02E−01	9.24E−01	8.75E−01	9.17E−01	8.83E−01	9.22E−01	9.22E−01

Fig. 24. The median images obtained using different optimization algorithms. Or = original image, BD = BDTPSO, CL = CLPSO, CS = CSO, FF = FFO.

levels in the upper and lower bands and the lowest average side lobe levels. Accordingly, the proposed method can be utilized to achieve the maximal reduction of side lobe levels, while satisfying different objectives.

6.6. Image thresholding

This section investigates the performance of BDTPSO on the image thresholding problem and compares its performance with CLPSO, CPSO, and FFO. The objective function given in Eq. (21) is employed in this section. Ten grayscale images are chosen randomly from the famous Berkeley image dataset [49] for the experiments. All algorithms are tested using three thresholds: 3, 9, and 15. Ordinarily, thresholding algorithms are tested using fewer than seven thresholds. However, we use a high number of thresholds to investigate the performance of the algorithms in complex optimization scenarios. The results of each test case are obtained over 30 independent runs, and each run is terminated after 2E+3 fitness evaluations. Image thresholding objective functions are usually computationally expensive. Hence, optimization

Table 11
Average Fitness, Standard Deviation and Wilcoxon rank-sum test on the Fitness for CEC2013 in 30D. The results are given in the mentioned order.

f	BDTPSO	MCUPS	DTT PSO	EPSO	HCLPSO	f	BDTPSO	MCUPS	DTT PSO	EPSO	HCLPSO
	0.00E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		2.95E+03	4.26E+03	**2.56E+03**	3.59E+03	3.42E+03
f_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{15}	5.84E+02	6.88E+02	**1.19E+03**	5.64E+02	5.87E+02
		(=)	(=)	(=)	(=)			(+)7E−14	(-)2E-06	(+)2E−06	(+)3E−04
	1.20E+04	2.26E+06	1.84E+05	1.65E+05	8.78E+05		1.44E+00	2.27E+00	2.47E+00	1.68E+00	**1.25E+00**
f_2	**5.25E+03**	1.14E+06	1.20E+05	7.98E+04	3.76E+05	f_{16}	3.76E−01	3.33E−01	3.03E−01	4.00E−01	**2.30E−01**
		(+)3E−18	(+)3E−18	(+)3E−18	(+)3E−18			(+)4E−14	(+)3E−16	(+)1E−03	(-)2E-02
	2.04E+04	4.74E+08	5.75E+07	1.38E+08	4.04E+07		3.52E+01	7.10E+01	4.80E+01	3.72E+01	**3.08E+01**
f_3	**3.68E+04**	4.33E+08	8.45E+07	2.71E+08	4.11E+07	f_{17}	1.43E+00	1.16E+01	5.01E+00	3.45E+00	**1.97E−01**
		(+)3E−18	(+)6E−18	(+)3E−18	(+)3E−18			(+)3E−18	(+)3E−18	(+)3E−03	(-)3E-18
	7.68E+01	1.29E+02	1.91E+04	1.78E+02	2.08E+03		**7.36E+01**	1.38E+02	1.82E+02	8.97E+01	8.19E+01
f_4	**3.23E+01**	6.76E+01	4.93E+03	1.09E+02	9.13E+02	f_{18}	**2.09E+01**	4.16E+01	9.71E+00	4.06E+01	1.45E+01
		(+)5E−05	(+)3E−18	(+)1E−08	(+)3E−18			(+)3E−14	(+)3E−18	(=)1E−01	(=)8E−02
	0.00E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		2.03E+00	3.09E+00	3.17E+00	1.78E+00	**1.37E+00**
f_5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{19}	3.06E−01	9.94E−01	6.72E−01	3.50E−01	**2.45E−01**
		(=)	(=)	(=)	(=)			(+)9E−13	(+)1E−15	(-)3E−04	(-)1E-14
	1.06E+00	3.42E+01	1.27E+01	9.97E+00	1.58E+01		**1.04E+01**	1.29E+01	1.10E+01	1.08E+01	1.10E+01
f_6	**5.17E+00**	2.48E+01	1.48E+01	5.70E+00	2.94E+00	f_{20}	**5.63E−01**	8.69E−01	7.88E−01	1.18E+00	7.12E−01
		(+)6E−17	(+)2E−15	(+)8E−15	(+)8E−16			(+)4E−18	(+)3E−07	(+)1E−02	(+)4E−06
	1.31E+01	1.38E+02	5.21E+01	3.24E+01	2.08E+01		2.97E+02	3.14E+02	3.28E+02	2.52E+02	**2.48E+02**
f_7	**5.54E+00**	4.29E+01	1.85E+01	1.52E+01	8.30E+00	f_{21}	4.92E+01	8.68E+01	7.87E+01	5.56E+01	**4.74E+01**
		(+)3E−18	(+)4E−17	(+)9E−13	(+)6E−07			(=)2E−01	(+)4E−02	(-)2E−02	(-)3E-04
	2.09E+01	**2.09E+01**	**2.10E+01**	**2.10E+01**	**2.10E+01**		1.75E+02	1.21E+03	1.44E+03	1.40E+02	**1.03E+02**
f_8	**4.36E−02**	**6.25E−02**	**3.96E−02**	**4.97E−02**	**4.60E−02**	f_{22}	5.35E+01	3.22E+02	4.05E+02	4.14E+01	**3.37E+01**
		(=)2E-01	(=)9E-02	(=)8E-02	(=)5E-02			(+)3E−18	(+)3E−18	(-)2E−05	(-)9E-17
	1.57E+01	2.76E+01	1.76E+01	2.24E+01	1.95E+01		**2.79E+03**	5.10E+03	2.86E+03	4.10E+03	3.82E+03
f_9	**3.44E+00**	3.54E+00	2.59E+00	2.82E+00	3.22E+00	f_{23}	**5.46E+02**	9.26E+02	6.37E+02	7.84E+02	5.97E+02
		(+)2E−17	(+)6E−03	(+)5E−14	(+)3E−06			(+)1E−17	(=)1E+00	(+)1E−12	(+)6E−12
	4.41E−02	1.41E−01	1.86E−01	1.62E−01	2.36E−01		**2.10E+02**	2.84E+02	2.48E+02	2.47E+02	2.28E+02
f_{10}	**2.40E−02**	7.69E−02	8.26E−02	8.63E−02	1.36E−01	f_{24}	**5.14E+00**	8.40E+00	1.03E+01	1.26E+01	8.73E+00
		(+)5E−13	(+)1E−16	(+)2E−15	(+)4E−17			(+)3E−18	(+)4E−18	(+)4E−18	(+)2E−15
	1.77E−01	2.12E+01	3.20E+01	**0.00E+00**	**0.00E+00**		**2.49E+02**	2.89E+02	2.73E+02	2.83E+02	2.70E+02
f_{11}	3.69E−01	6.59E+00	8.11E+00	**0.00E+00**	**0.00E+00**	f_{25}	**6.41E+00**	1.10E+01	8.73E+00	1.15E+01	2.15E+01
		(+)3E−18	(+)3E−18	(-)9E-07	(-)9E-07			(+)3E−18	(+)4E−17	(+)1E−16	(+)5E−11
	2.11E+01	1.75E+02	3.74E+01	5.21E+01	5.49E+01		**2.00E+02**	3.31E+02	3.06E+02	**2.00E+02**	**2.00E+02**
f_{12}	**6.07E+00**	5.06E+01	9.84E+00	1.58E+01	1.38E+01	f_{26}	**4.12E−03**	7.39E+01	5.67E+01	**7.75E−03**	**9.23E−03**
		(+)3E−18	(+)3E−13	(+)2E−17	(+)2E−17			(+)3E−18	(+)2E−17	(=)9E-02	(=)7E-02
	5.10E+01	2.14E+02	1.02E+02	1.17E+02	1.24E+02		**3.98E+02**	1.05E+03	7.01E+02	7.20E+02	5.92E+02
f_{13}	**1.55E+01**	4.49E+01	2.76E+01	3.11E+01	2.50E+01	f_{27}	**5.32E+01**	9.03E+01	9.91E+01	2.00E+02	1.23E+02
		(+)3E−18	(+)1E−14	(+)1E−16	(+)7E−18			(+)3E−18	(+)5E−18	(+)5E−12	(+)6E−13
	1.46E+02	1.29E+03	1.53E+03	2.91E+01	**1.60E+01**		**2.96E+02**	4.53E+02	4.00E+02	**2.96E+02**	3.00E+02
f_{14}	1.08E+02	2.91E+02	3.12E+02	1.02E+01	**4.52E+01**	f_{28}	**2.80E+01**	4.98E+02	4.40E+02	**2.80E+01**	2.73E−13
		(+)3E−18	(+)3E−18	(-)8E−16	(-)1E-14			(=)4E−01	(=)2E−01	(=)1E-01	(=)3E−01

algorithms utilized on image thresholding problems should be able to reach acceptable solutions with a manageable number of fitness evaluations. CLPSO, CPSO, and FFO have been employed successfully for solving various image thresholding and segmentation problems [50–52]. These algorithms are capable of obtaining high-quality solutions in a limited number of iterations.

First, algorithms are compared based on their obtained fitness values. Tables 7–9 give the average results and their standard deviations. Additionally, the Wilcoxon rank-sum test with a confidence level of 95% is used to analyze the results statistically. The tested peer methods are very competitive, and all were able to reach optimum or near optimum solutions with the designated number of fitness evaluations. BDTPSO has obtained the optimum solutions, according to the defined objective function, on all of the 3-thresholding problems. Other compared algorithms, specifically CSO, have also been capable of finding the optimum solutions. However, the precision of their obtained solutions is low in comparison to BDTPSO. The proposed method has obtained the highest average solutions (note that the results have been rounded to two decimal points) with the lowest standard deviations in all tested instances. BDTPSO is also statistically superior to the compared algorithms in most tested scenarios according to the Wilcoxon rank-sum tests. It has delivered a robust performance on the image thresholding problem when the number of thresholds is three. As the complexity of the problem increases by increasing the number of thresholds, the superiority of BDTPSO to the other peer methods becomes more apparent. The proposed method has obtained the best average results on the thresholding problems when the number of thresholds is 9 and 15. It has demonstrated a robust performance on these problems by obtaining the best solutions in almost all runs. This robustness indicates that the proposed method is suitable for solving various image segmentation problems.

The results demonstrate that the proposed method can obtain solutions with high fitness values on different images. However, other evaluation criteria are required to examine the suitability of the employed objective function. One of the widely utilized images quality assessment matrices is the feature similarity (FSIM) index [40], and Table 10 compares different methods according to this criterion. The results of the compared methods are close according to the FSIM metric when the number of thresholds is three. However, as the number of thresholds increases, the results of the proposed method become better than other peer methods. Fig. 24 shows the results of the first five images of the dataset for the segmentation problems with nine thresholds. Each given image has the median fitness among the 30 obtained results (the

Table 12
Average Fitness, Standard Deviation and Wilcoxon rank-sum test on the Fitness for CEC2013 in 30D. The results are given in the mentioned order.

f	BDTPSO	MCUPS	DTT PSO	EPSO	HCLPSO	f	BDTPSO	MCUPS	DTT PSO	EPSO	HCLPSO
f_1	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	f_{15}	6.57E+03 9.72E+02	8.32E+03 1.13E+03	**6.27E+03** **2.27E+03**	7.03E+03 6.55E+02	6.95E+03 7.65E+02
f_2	**9.68E+04** **5.47E+04**	5.21E+06 1.90E+06	4.35E+05 1.27E+05	5.80E+05 1.93E+05	1.64E+06 5.40E+05	f_{16}	1.86E+00 3.92E−01	3.05E+00 3.83E−01	3.39E+00 2.63E−01	1.98E+00 3.24E−01	**1.64E+00** **2.86E−01**
f_3	**8.99E+05** **1.24E+06**	1.81E+09 1.76E+09	1.33E+08 1.52E+08	3.02E+08 4.29E+08	1.76E+08 1.81E+08	f_{17}	6.65E+01 3.44E+00	1.49E+02 2.20E+01	1.13E+02 1.32E+01	7.15E+01 1.02E+01	**5.12E+01** **1.78E−01**
f_4	**4.23E+01** **2.94E+01**	1.18E+02 4.59E+01	2.08E+04 5.27E+03	8.71E+01 3.87E+01	1.77E+03 6.44E+02	f_{18}	**1.19E+02** **3.83E+01**	3.06E+02 8.42E+01	3.87E+02 1.71E+01	1.57E+02 4.28E+01	1.74E+02 3.93E+01
f_5	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	**0.00E+00** **0.00E+00**	f_{19}	4.24E+00 8.13E−01	6.35E+00 1.48E+00	8.98E+00 1.98E+00	3.74E+00 8.88E−01	**2.38E+00** **3.39E−01**
f_6	**4.34E+01** **6.17E−14**	6.25E+01 2.46E+01	4.57E+01 1.27E+01	**4.34E+01** **2.58E+00**	4.45E+01 8.22E−01	f_{20}	**1.86E+01** **9.24E−01**	2.25E+01 1.50E+00	2.04E+01 7.37E−01	1.98E+01 9.58E−01	2.03E+01 9.44E−01
f_7	**2.24E+01** **5.91E+00**	1.16E+02 2.88E+01	5.83E+01 1.05E+01	5.18E+01 1.30E+01	4.15E+01 1.16E+01	f_{21}	7.57E+02 3.67E+02	8.23E+02 3.43E+02	7.31E+02 3.74E+02	2.81E+02 1.55E+02	**2.69E+02** **1.58E+02**
f_8	**2.11E+01** **3.98E−02**	**2.11E+01** **4.83E−02**	**2.11E+01** **4.04E−02**	**2.11E+01** **3.83E−02**	**2.11E+01** **3.46E−02**	f_{22}	3.98E+02 1.94E+02	2.42E+03 7.27E+02	3.80E+03 1.11E+03	1.17E+02 7.34E+01	**5.30E+01** **5.59E+01**
f_9	**2.69E+01** **4.65E+00**	5.34E+01 4.71E+00	3.41E+01 3.16E+00	4.57E+01 5.05E+00	4.10E+01 6.52E+00	f_{23}	**5.93E+03** **8.68E+02**	9.99E+03 1.45E+03	6.74E+03 7.72E+02	8.66E+03 1.18E+03	7.96E+03 8.57E+02
f_{10}	**5.32E−02** **3.32E−02**	2.71E−01 3.75E−01	2.07E−01 1.01E−01	1.80E−01 9.53E−02	2.16E−01 1.09E−01	f_{24}	**2.41E+02** **9.30E+00**	3.56E+02 1.16E+01	2.97E+02 1.20E+01	3.15E+02 1.34E+01	2.85E+02 1.43E+01
f_{11}	2.02E+00 1.80E+00	5.75E+01 1.82E+01	9.42E+01 1.79E+01	1.87E−01 4.61E−01	**0.00E+00** **0.00E+00**	f_{25}	**3.02E+02** **1.03E+01**	3.75E+02 1.46E+01	3.38E+02 1.07E+01	3.76E+02 1.13E+01	3.58E+02 1.67E+01
f_{12}	**5.02E+01** **1.03E+01**	2.61E+02 5.92E+01	9.46E+01 1.46E+01	1.28E+02 3.29E+01	1.32E+02 2.23E+01	f_{26}	**2.00E+02** **1.50E−01**	4.22E+02 6.23E+01	3.68E+02 5.80E+01	**2.00E+02** **2.09E−02**	2.03E+02 2.63E+01
f_{13}	**1.31E+02** **2.40E+01**	3.65E+02 7.60E+01	2.30E+02 3.66E+01	2.66E+02 5.36E+01	2.54E+02 5.16E+01	f_{27}	**8.30E+02** **1.25E+02**	1.74E+03 1.59E+02	1.26E+03 1.18E+02	1.44E+03 2.40E+02	1.24E+03 2.08E+02
f_{14}	3.30E+02 1.83E+02	2.33E+03 5.46E+02	3.22E+03 6.23E+02	5.06E+01 1.95E+01	**1.72E+01** **4.50E+01**	f_{28}	4.02E+02 1.73E+01	1.80E+03 1.63E+03	7.28E+02 1.00E+03	**4.00E+02** **1.06E−12**	**4.00E+02** **6.98E−13**

15th best solution). Since the number of the employed thresholds is large, the obtained images are very similar to the original ones. Accordingly, recognizing differences between the results may require careful observation. Some of these differences are highlighted in the first images. Looking closely at the images, it can be observed that the results created by the proposed method are the most similar images to their original versions.

7. Conclusion

This paper presented a bifurcated particle swarm optimizer and investigated its application on various optimization problems. The searching process of the proposed method is bifurcated among two types of particles, which are called even and even particles. Each particle type has its search objectives and induces specific behavior during the search. While even particles can discover appropriate flying trajectories in the search space through topology adaptation, uneven particles can combine and exploit different solutions obtained by the even particles. The cooperation among these two types of particles enables the proposed method to effectively balance its explorative and exploitative features during the search procedure. We investigated the effectiveness of BDTPSO on three categories of problems: the comprehensive CEC2013 benchmark set, antenna array synthesis, and image segmentation. The comparisons with several advanced PSO methods indicated that the introduced mechanisms can successfully control the behavior of particles and guide them toward promising solutions. Also, BDTPSO was employed for synthesizing non-uniform circular antenna arrays. This problem had several design objectives, which includes achieving minimum side lobe levels and maximum directivity. Experimental studies demonstrated that the proposed approach can be employed successfully for synthesizing antenna arrays according to these design objectives. BDTPSO was capable of obtaining solutions that lead to a maximum reduction of side lobe levels while satisfying predefined FNBWs. The proposed algorithm was also employed for image segmentation via thresholding. On this real-world application, the evaluation budget was kept very low due to the computational cost of its optimization problem. The proposed approach was capable of reaching the best results among the compared methods, which indicates the ability of BDTPSO in solving problems with strict time limits.

CRediT authorship contribution statement

Reza Vafashoar: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing. **Hossein Morshedlou:** Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Mohammad Reza Meybodi:** Resources, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

Statistical results on the 30 dimensional CEC2013 benchmark set in terms of the average and standard deviation of the error obtained in different runs are given in Table 11. Also, the results of Wilcoxon rank sum test are provided in the table. It should be noted that on some problems like f_{26} different compared methods occasionally get trapped in some similar common local optima (for instance, local optima with the error of 200 in the case of f_{26}). On these problems, we consider obtained results closer than 1E-8 as identical and truncate them to identical values before Wilcoxon tests.

Table 13
Design variables obtained by BDTPSO for different antenna synthesis problems (results are rounded to four decimal places).

NE	d_i in terms of wavelength	Normalized I_i	d_i in terms of wavelength	Normalized I_i
15	0.6096, 1.0000, 0.6795, 0.7048, 0.6212, 1.0000, 0.5107, 0.5119, 0.5006, 0.7384, 0.7482, 0.6685, 0.6840, 0.9999, 0.6111	0.5891, 0.3812, 0.3046, 0.1835, 0.4101, 0.4091, 0.8019, 0.6966, 0.2064, 0.3684, 0.3456, 0.2021, 0.4344, 0.5706, 0.9971	0.5956, 1.0000, 0.6965, 0.6797, 0.7389, 0.6570, 0.5099, 0.5305, 0.5434, 1.0000, 0.6552, 0.6832, 0.6914, 1.0000, 0.6122,	0.6207, 0.3866, 0.2563, 0.3058, 0.3574, 0.1583, 0.6807, 0.8930, 0.4645, 0.4574, 0.1982, 0.3314, 0.4205, 0.6047, 1.0000
20	0.5985, 0.5000, 0.9836, 0.7604, 1.0000, 1.0000, 0.8024, 0.9955, 0.6048, 0.5363, 0.5062, 0.6000, 0.9999, 0.8573, 0.9386, 1.0000, 0.7372, 1.0000, 0.5000, 0.6070	1.0000, 0.8070, 0.5645, 0.5614, 0.0002, 0.3793, 0.6761, 0.6001, 0.9538, 0.9645, 0.8273, 0.6429, 0.7127, 0.3172, 0.0001, 0.5209, 0.5575, 0.8103, 0.9900, 0.9991	0.6297, 0.5154, 0.9897, 0.6521, 0.7099, 0.7055, 0.9577, 1.0000, 0.5595, 0.6062, 0.5348, 0.6204, 0.5000, 0.8148, 0.6868, 0.6511, 0.7583, 0.9656, 0.5000, 0.6386	0.9823, 0.5843, 0.7384, 0.0867, 0.4576, 0.1977, 0.6897, 0.9580, 0.8054, 0.9757, 0.7144, 0.2902, 0.5474, 0.1400, 0.5476, 0.0976, 0.7244, 0.5748, 0.9771, 1.0000

Appendix B

Statistical results on the 50 dimensional CEC2013 benchmark set in terms of the average, standard deviation of the error obtained in different runs are provided in Table 12.

Appendix C

The best design variables obtained by BDTPSO on the two antenna array synthesis problems are provided in Table 13.

References

[1] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Ieee, 1995, pp. 39–43.
[2] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[3] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings, in: IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 69–73.
[4] M. Xu, J. Shi, W. Chen, J. Shen, H. Gao, J. Zhao, A band selection method for hyperspectral image based on particle swarm optimization algorithm with dynamic sub-swarms, J. Signal Process. Syst. 90 (2018) 1269–1279.
[5] X. f. Song, Y. Zhang, D. w. Gong, X. y. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, Pattern Recognit. 112 (2021) 107804.
[6] B.F. Gumaida, J. Luo, A hybrid particle swarm optimization with a variable neighborhood search for the localization enhancement in wireless sensor networks, Appl. Intell. 49 (2019) 3539–3557.
[7] A.K. Kashyap, D.R. Parhi, Particle swarm optimization aided PID gait controller design for a humanoid robot, ISA Trans. 114 (2021) 306–330.
[8] B. Sahu, P.K. Das, M.R. Kabat, R. Kumar, Multi-robot cooperation and performance analysis with particle swarm optimization variants, Multimedia Tools Appl. (2021) 1–24.
[9] A. Verma, S. Kaushal, A hybrid multi-objective particle swarm optimization for scientific workflow scheduling, Parallel Comput. 62 (2017) 1–19.
[10] Z.-G. Chen, Z.-H. Zhan, D. Liu, S. Kwong, J. Zhang, Particle swarm optimization with hybrid ring topology for multimodal optimization problems, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2020, pp. 2044–2049.
[11] L. Han, Q. Zhou, J. Tang, X. Yang, H. Huang, Identifying Top-k influential nodes based on discrete particle swarm optimization with local neighborhood degree centrality, IEEE Access 9 (2021) 21345–21356.
[12] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, X. Liu, A dynamic neighborhood-based switching particle swarm optimization algorithm, IEEE Trans. Cybern. (2020).
[13] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (2004) 204–210.
[14] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[15] H. Liu, E. Howely, J. Duggan, Particle swarm optimization with gradually increasing directed neighbourhoods, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, 2011, pp. 29–36.
[16] M.R. Bonyadi, X. Li, Z. Michalewicz, A hybrid particle swarm with a time-adaptive topology for constrained optimization, Swarm Evol. Comput. 18 (2014) 22–37.
[17] X. Xia, L. Gui, Z.-H. Zhan, A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting, Appl. Soft Comput. 67 (2018) 126–140.
[18] M.A.M. De Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Trans. Evol. Comput. 13 (2009) 1120–1132.
[19] R. Vafashoar, M.R. Meybodi, Multi swarm optimization algorithm with adaptive connectivity degree, Appl. Intell. 48 (2018) 909–941.
[20] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Appl. Soft Comput. 48 (2016) 584–596.
[21] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Inform. Sci. 209 (2012) 16–36.
[22] B. Jiang, N. Wang, L. Wang, Particle swarm optimization with age-group topology for multimodal functions and data clustering, Commun. Nonlinear Sci. Numer. Simul. 18 (2013) 3134–3145.
[23] W. Sun, A. Lin, H. Yu, Q. Liang, G. Wu, All-dimension neighborhood based particle swarm optimization with randomly selected neighbors, Inform. Sci. 405 (2017) 141–156.
[24] A. Flori, H. Oulhadj, P. Siarry, A new neighborhood topology for Quantum particle swarm optimization QUAPSO, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 255–256.
[25] S. Chourasia, H. Sharma, M. Singh, J.C. Bansal, Global and local neighborhood based particle swarm optimization, in: Harmony Search and Nature Inspired Optimization Algorithms, Springer, 2019, pp. 449–460.
[26] S. Sengupta, S. Basak, R.A. Peters, Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives, Mach. Learn. Knowl. Extr. 1 (2019) 157–191.
[27] Y. Zhang, X. Liu, F. Bao, J. Chi, C. Zhang, P. Liu, Particle swarm optimization with adaptive learning strategy, Knowl.-Based Syst. (2020) 105789.
[28] Y. Ning, Z. Peng, Y. Dai, D. Bi, J. Wang, Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems, Appl. Intell. 49 (2019) 335–351.
[29] M. Isiet, M. Gadala, Self-adapting control parameters in particle swarm optimization, Appl. Soft Comput. 83 (2019) 105653.
[30] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[31] J. Riget, J.S. Vesterstrøm, A Diversity-Guided Particle Swarm Optimizer-the ARPSO, Tech. Rep, 2, (2002) Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, 2002.
[32] M. Giacobini, M. Tomassini, A.G. Tettamanzi, E. Alba, Selection intensity in cellular evolutionary algorithms for regular lattices, IEEE Trans. Evol. Comput. 9 (2005) 489–505.
[33] M. Tomassini, Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time, Springer, 2006.
[34] M.A. Montes de Oca, T. Stützle, Convergence behavior of the fully informed particle swarm optimization algorithm, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, 2008, pp. 71–78.
[35] D.N. Wilke, S. Kok, A.A. Groenwold, Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity, Internat. J. Numer. Methods Engrg. 70 (2007) 962–984.
[36] G.G. Roy, S. Das, P. Chakraborty, P.N. Suganthan, Design of non-uniform circular antenna arrays using a modified invasive weed optimization algorithm, IEEE Trans. Antennas and Propagation 59 (2010) 110–118.
[37] M. Abd El Aziz, A.A. Ewees, A.E. Hassanien, Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation, Expert Syst. Appl. 83 (2017) 242–256.
[38] A.A. Ewees, M. Abd Elaziz, M.A. Al-Qaness, H.A. Khalil, S. Kim, Improved artificial bee colony using sine-cosine algorithm for multi-level thresholding image segmentation, IEEE Access 8 (2020) 26304–26315.
[39] E.H. Houssein, B.E. d. Helmy, D. Oliva, A.A. Elngar, H. Shaban, A novel black widow optimization algorithm for multilevel thresholding image segmentation, Expert Syst. Appl. 167 (2021) 114159.
[40] S. Sarkar, S. Paul, R. Burman, S. Das, S.S. Chaudhuri, A fuzzy entropy based multi-level image thresholding using differential evolution, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2014, pp. 386–395.

[41] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Technical Report, 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2013, pp. 281–295.
[42] H.-C. Tsai, Unified particle swarm delivers high efficiency to particle swarm optimization, Appl. Soft Comput. 55 (2017) 371–383.
[43] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.
[44] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533–548.
[45] S. Banerjee, D. Mandal, Array pattern optimization for steerable circular isotropic antenna array using cat swarm optimization algorithm, Wirel. Pers. Commun. 99 (2018) 1169–1194.
[46] A.M. Ismaiel, E. Elsaidy, Y. Albagory, H.A. Atallah, A.B. Abdel-Rahman, T. Sallam, Performance improvement of high altitude platform using concentric circular antenna array based on particle swarm optimization, AEU-Int. J. Electron. Commun. 91 (2018) 85–90.
[47] S. Banerjee, D. Mandal, Array pattern optimization for a steerable circular isotropic antenna array using the firefly algorithm, J. Comput. Electron. 16 (2017) 952–976.
[48] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[49] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings Eighth IEEE International Conference on Computer Vision, Vol. 2001, ICCV, IEEE, 2001, pp. 416–423.
[50] A. Sharma, R. Chaturvedi, S. Kumar, U.K. Dwivedi, Multi-level image thresholding based on kapur and tsallis entropy using firefly algorithm, J. Interdiscip. Math. 23 (2020) 563–571.
[51] M. Karakoyun, N.A. Baykan, M. Hacibeyoglu, Multi-level thresholding for image segmentation with swarm optimization algorithms, Int. Res. J. Electron. Comput. Eng. 3 (2017) 1–6.
[52] L.J. Ahmed, A.E. Jeyakumar, Image segmentation using a refined comprehensive learning particle swarm optimizer for maximum tsallis entropy thresholding, Int. J. Eng. Technol. 5 (2013) 3608–3616.