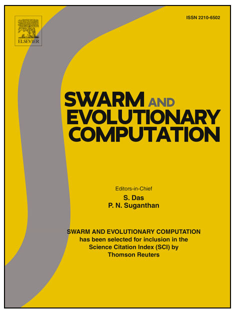

Cellular learning automata based bare bones PSO with maximum likelihood rotated mutations

Reza Vafashoar, Mohammad Reza Meybodi

PII: S2210-6502(16)30556-9

DOI: 10.1016/j.swevo.2018.08.016

Reference: SWEVO 439

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 21 December 2016

Revised Date: 5 July 2018

Accepted Date: 22 August 2018

Cellular learning automata based bare bones PSO with maximum likelihood rotated mutations

Reza Vafashoar, Mohammad Reza Meybodi

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran.

Tel: +98-21-64545120; Fax: +98-21-66495521; e-mail: (vafashoar, mmeybodi)@aut.ac.ir

Abstract

In this work, a new variant of multi-swarm bare bones PSO is presented which adaptively learns the promising alignments to re-orient its updating distributions. The idea is to maximize the likelihood of generating new particles along appropriate directions. As these alignment directions are unknown a priori, the proposed method employs a learning mechanism for adaptive learning of suitable alignments. The learning mechanism presented in this paper is based on cellular learning automata. Several alignment strategies are developed for each particle in the proposed method, and the cellular learning automata guides the particles toward the most promising directions by adjusting these strategies during the searching process. The new proposed algorithm is compared with a group of stochastic optimization approaches on the CEC2013 benchmark set. Experiential studies suggest the effectiveness of the proposed method in solving complex optimization problems.

Key words— Particle swarm optimization, Multi-swarm PSO, Global numerical optimization, learning automata, Cellular learning automata.

1. Introduction

Particle swarm optimization (PSO) is a stochastic search approach inspired from the intelligent social behavior that exists among some animals such as bird and fish [1]. It maintains a population of particles termed as a swarm. These particles, guided by their velocities, fly in the search space to locate better resources. Each particle can memories its personal best visited location in the search space. Using its personal best location and the global best location observed by the whole swarm, a particle adjusts its velocity, and accordingly, changes its current position in the search space. PSO has some interesting features such as robustness and simplicity, which makes it a desirable tool for solving many real world problems. However, it possesses low capabilities in escaping from poor local optima in complex multi modal problems and is often subjected to premature convergence [2], and due to problems such as oscillation or zigzagging, PSO suffers from a poor performance [3].

Several variants of PSO have been introduced recently to overcome its drawbacks. Most of these variants incorporate various learning schemes, updating rules, and population topologies to achieve enhanced performances. Heterogeneous comprehensive learning particle swarm optimization

(HCLPSO) employs two specific subpopulations for exploration and exploitation purposes. It uses a comprehensive learning (CL) strategy to generate the exemplars for both of these subpopulations [4]. Another interesting approach developed by Lynn and Suganthan hybridizes several efficient PSO methods into a unified optimizer called ensemble PSO (EPSO) [5]. Like HCLPSO, EPSO divides its population into two sub-swarms. A small sub-swarm searches the problem landscape based on comprehensive learning PSO (CLPSO) searching procedure, while an integrated self-adaptive mechanism learns an appropriate searching scheme in the second and larger sub-swarm. Another multi-swarm PSO with dynamic learning strategy is presented by Ye et al. [6]. In this algorithm, particles are classified into ordinary and communication particles and, based on this classification, are assigned to different tasks.

Several works have been conducted to study the behavior of particles in PSO [7-11]. In this regard, Kennedy introduced an alternative model for PSO [12], in which velocity and position updating rules are replaced by Gaussian sampling around the average of the global and personal best positions. The new model, termed as bare bones PSO (BBPSO), was mainly developed for investigating the characteristics of the PSO algorithm. BBPSO, as is demonstrated in [13], is an inferior optimization method in comparison to the ordinary PSO algorithm. Up to now, several attempts have been made to improve the performance of BBPSO; however, most of the introduced methods are less effective than the advanced optimization methods which are based on the algorithms such as differential evolution [14] or ordinary PSO. As a result, BBPSO has attracted less interest and usage in solving practical optimization problems in comparison to the other meta-heuristics.

The main drawback of BBPSO lies in its updating rules; accordingly, the majority of works on BBPSO have examined the utilization of various updating strategies and distributions. Campos et al. have developed a variant of BBPSO which employs a multivariate t-distribution for acquiring the new positions of the particles [15]. In their work, each particle is associated with its own scale matrix which is adapted during the search process to produce more promising distributions for the particle. Cooperatively coevolving particle swarms (CCPSO) is a cooperative approach which uses Cauchy and Gaussian distributions in its updating rules [16]. During each iteration of CCPSO, the coordinates of the particles are divided into *K* swarms, and each swarm is updated in a cooperative manner. The number of swarms (*K*) is adapted during the course of the execution to enhance the search. A modified BBPSO which uses adaptive updating distributions was introduced by Vafashoar and Meybodi [17]. They have developed four updating strategies based on Gaussian and multivariate Gaussian distributions. The introduced model is integrated with a CLA which adaptively selects an appropriate updating rule for each particle based on the characteristics of the fitness landscape. They also introduced a technique for the adaption of the covariance matrices of the Gaussian distributions in the proposed approach. In order to balance the exploration and exploitation characteristics of the algorithm, Chen introduced a modified BBPSO which uses a mixture of two centers in the Gaussian distribution that updates the position of a particle. For a particular particle, these centers are based on its local best position and the global best position of the swarm. The algorithm starts with bigger weights for the local best centers and gradually increases the weight of the global best one. Accordingly, it is expected to have more explorative behavior at the beginning while it mostly exploits the neighborhood of the global best position at the end [18-19]. Most of these modified updating rules still employ limited information about the fitness landscape.

Similar to the PSO methods like HCLPSO, some recently developed BBPSO approaches use different updating schemes for exploration and exploitation purposes. Entropy-based BBPSO (EBBPSO) uses entropy to measure its population diversity [20]. High diversity promotes the search guided by the global-best solutions with normal distribution; however, EBBPSO promotes exploration by means of local-best solutions and a heavy-tailed distribution in the case of low diversity. Dynamic allocation BBPSO (DABBPSO) divides its particles into two groups of equal sizes. The group termed as ancillary is responsible for exploration, while the other group exploits the promising areas of the search space. Each group uses a specific updating scheme according to its designated task [21].

Due to the characteristics of the updating rules of the classical BBPSO, the global best particle remains unchanged during the update procedure. Considering this shortcoming, BBPSO-MC updates the best particle of the swarm in a rather different way and employs the update strategies of differential evolution algorithm for this purpose [22]. Several mutation mechanisms are also introduced for bare bones PSO to enhance its exploration abilities and to cope with stagnation [23-25]. In an approach introduced by Krohling and Mendel, the particles undergo Gaussian or Cauchy jumps when they are trapped in local optima and cannot improve their finesses for a period of time [24]. Other jumping mechanisms have also been utilized in bare bones PSO by al-Rifaie and Blackwell [26]. Gao et al. introduced a new hybridized method based on bee colony and BBPSO. As the bee colony algorithm is considered to be less exploitative by the authors, they used BBPSO in the onlooker phase of the bee colony algorithm to enhance its exploitation properties [27].

The work presented in this paper is based on cellular learning automata (CLA). Incorporating the learning capabilities of learning automata [28] into a cellular automaton [29], Beigy and Meybodi introduced a new reinforcement learning approach called cellular learning automata [30]. CLA is a hybrid mathematical model for many decentralized problems. It can be considered as a network of agents with learning abilities. The interaction among the agents is guided by the CLA rules, and these agents use the learning properties of CLA to adapt their behavior. CLA has been applied in many scientific domains such as deployment and clustering in wireless sensor networks [31-34], dynamic channel assignment in cellular networks [35-36], evolutionary computing [37], and graph coloring [38].

As stated previously, the main drawback of BBPSO lies in its updating scheme, which is simply based on Gaussian distributions. Aside from the standard deviation of these Gaussian distributions and their mean points, which only contain very limited information about the fitness landscape, the updating strategies of most of the developed BBPSO methods do not employ much information about the fitness landscape. As the most practical optimization problems involve many related and correlated variables, the simple updating mechanism based on Gaussian distributions proves to be insufficient and less effective. The current work tries to alleviate this problem by introducing more depth to the updating procedure of BBPSO. Its main goal is to use the information present in the search history to improve the updating strategies of the algorithm. According to the learned characteristics of the fitness landscape, the modified updating scheme tries to increase the probability of generating favorable positions.

In order to improve the updating distributions of BBPSO, this paper introduces the concept of alignment directions. By carefully defining the updating distributions of the particles, we can align them along some promising directions. These alignments are performed to increase the likelihood of generating new positions along some promising paths and directions which are learned during the searching process. In the proposed method, the promising alignment directions are obtained based on a set of strategies. These strategies are defined in terms of two concepts named improving path and flying path. These two concepts are developed to encode the previous experiences and observations of the particles. The flying path is the accumulation of the historical paths a particle has taken previously in its successful updates. The improving path is the accumulation of the observations which would have led the particle into better areas. As the choice of the most effective alignment strategy depends on the properties of the fitness landscape, the alignment strategies are learned during the searching process. In this regard, the proposed model is embedded in a cellular learning automata [30], which has proven to be a successful model for similar problems [17, 39]. In the proposed method, each particle is associated with a learning automaton, and the learning automaton governs the learning process of the alignment strategies for the particle.

The introduced method also investigates the applicability of new learning scheme for the cellular learning automata. Most of the developed CLA models are based on the learning algorithms L_{RI}, L_{RP}, and $L_{R\varepsilon P}$ [28]. However, the CLA model in the current work is based on the generalized pursuit algorithm, which has demonstrated to be superior to many other traditional learning automata algorithms [40].

The rest of the paper is organized as follows: after a review on related works and topics, section 4 defines maximum likelihood rotation matrices for Gaussian distributions. The proposed method is described in section 5, and is validated experimentally in section 6. Finally, the main conclusions and the areas for future works are given in section 7.

2. PSO and related topics

Particle swarm optimization executes its search in a specified space through the accumulation of velocity and position information. At the beginning, each particle is initiated randomly within a D dimensional search space. It keeps information about its personal best visited position known as pbest, swarm best position known as gbest, and its current flying velocity. Canonical PSO maintains a swarm of particles; each particle X_i in the swarm maintains three D dimensional vectors: $x_i=[x_{i1},x_{i2},\dots,x_{iD}]$ representing its current position, $p_i=[p_{i1},p_{i2},\dots,p_{iD}]$ representing the best location previously experienced by the particle, and $v_i=[v_{i1},v_{i2},\dots,v_{iD}]$ representing its flying velocity. The whole swarm keeps information about its best experienced position in a D dimensional vector like $p_g=[p_{g1},p_{g2},\dots,p_{gD}]$. During the search, at step k, the velocity of the particle X_i and its current position are updated according to the following equations:

$$\begin{aligned}&v_{id}(k+1) \leftarrow \omega v_{id}(k) + c_1 r_1 (p_{id} - x_{id}(k)) \\ &+c_2 r_2 (p_{gd} - x_{id}(k)) \\ &x_{id}(k+1) \leftarrow x_{id}(k) + v_{id}(k+1)\end{aligned} \tag{1}$$

where $v_{id}(k)$ is the d^{th} dimension of the velocity vector of the particle in step k; $x_{id}(k)$ and $p_{id}(k)$ are respectively the d^{th} dimension of its position and historical best position vectors; $p_{gd}(k)$ represents the d^{th} dimension of the historical best position of the whole swarm in step k; ω is the inertia weight, which was introduced to bring a balance between the exploration and exploitation characteristics [41]; c_1 and c_2 are acceleration constants and represent cognitive and social learning weights; and, finally, r_1 and r_2 are two random numbers from the uniform distribution $u(0, 1)$.

After acquiring the new positions of the particles, the historical best position of each particle is updated, which may also affect the historical global best position of the swarm.

2.1. Bare bones particle swarm optimization

Bare bones PSO was originally developed to study the dynamics of the particle swarm optimization algorithm. As in canonical PSO the spread of the positions visited by a single particle resembles a Gaussian like distribution, BBPSO replaces the update equations of PSO with explicit probability distributions [12]. Consequently, it is suggested that the new position of a particle like X_i can be generated according to the following equation, in which the velocity term of PSO is eliminated:

$$x_{id}(k+1) \leftarrow \frac{p_{id}+p_{gd}}{2} + N(0,1)\times\left|p_{id}-p_{gd}\right| \quad (2)$$

where $N(0,1)$ denotes a random number taken from the standard normal distribution, p_g is the global best position of the swarm, and p_i is the personal best position of the particle itself.

Kennedy suggested that preserving some components of the personal best positions would speed up the search process of BBPSO. Accordingly, the following updating rule was recommended:

$$x_{id}(k+1) \leftarrow \begin{cases} \frac{p_{id}+pl_{id}}{2} + N(0,1)\times\left|p_{id}-pl_{id}\right| & \text{if } r < IP \\ p_{id} & \text{otherwise} \end{cases} \quad (3)$$

where IP is a parameter which controls the portion of a particle to be remained unchanged, r is a random number from uniform distribution $r\sim(0,1)$, and pl_i is the local best position in the neighborhood of the i^{th} particle.

3. Cellular Learning Automata

Cellular Learning Automata is an abstract model for the systems that consist of a large number of simple entities with learning capabilities [30]. It is a hybridization of a cellular automaton [29] and learning automata [28] in which each cell of the cellular automaton contains one or more learning automata. The state of the learning automata in a cell defines the state of cell. CLA is integrated with a set of local rules which describe the interactions between neighbouring cells. The neighbouring cells of a particular cell constitute its environment. Each cell of the CLA adapts during the time based on its own experience and the behaviour of its neighbouring cells.

The evolution of a CLA happens according to the following procedure. At each step, the learning automata residing in each cell select their actions based on their internal probability distributions. The selected actions are applied to the corresponding environments. Then, each performed action is evaluated by the environment; next, based on a set of CLA rules, the environment returns a reinforcement signal to the learning automaton. According to the received signals, the learning automata adjust their internal action probabilities.

3.1. Learning automata and generalized pursuit algorithm

A learning automaton (LA) is an adaptive decision making unit which operates in a stochastic environment [28]. The utilization of a learning automaton in an environment involves a repeated procedure of some steps, where it can select and perform an action out of its available actions at each step. The objective of the learning automaton is the selection of the actions that best fit with the environment. This goal is achieved through the repeated interactions that exist among the learning automaton and the environment, and a learning mechanism which updates the action probability distribution according to these interactions.

LA can be defined by a set of actions $a=\{a_1,a_2,\dots,a_r\}$, a set of inputs β which are the feedbacks from the environment, an internal probability vector q which represents the selection probability of different actions, and a learning algorithm that adjusts the probability values. One of the most effective learning automata models is the generalized pursuit algorithm developed by Agache and Oommen [40], which generalizes the pursuit algorithm introduced in [42]. The algorithm uses the history of selected actions and obtained reinforcements to estimate the reward probability of different actions. The reward estimate of the i^{th} action of the LA at step t will be denoted by $d_i(t)$. The objective of the generalized pursuit algorithm is to select the optimal action, the action with the highest reward probability, with probability one. However, as the estimated rewards could be erroneous, the algorithm gradually increases the selection probability of the most promising action. In this regard, at each step, the algorithm first decreases the selection probability of all actions. Then, the total amount which has been reduced from the probability values is equally distributed among all promising actions. Classical pursuit algorithm considers the action with the highest estimated reward as the only promising action, which receives the total discounted probability amount. However at each step, generalized pursuit considers all the actions that have higher estimates than the chosen action as the promising actions.

In order to calculate the estimated reward of each action, the algorithm maintains two vectors: $Q_i(t)$ which is the number of times the i^{th} action is rewarded up to step t, and $Z_i(t)$ which is the number of times that the i^{th} action has been selected till step t. The algorithm operates as follows: at each step t, LA selects an action, $a(t)$, according to its internal probability vector $q(t)$. Then, the action is performed in the environment and a reinforcement signal $\beta(t)$ is received from the environment in response to the selected action. Then, the LA updates its internal probability vector according to Eq. 4 and uses $a(t)$ and $\beta(t)$ to obtain new estimates according to Eq. 5.

$$q_j(t+1)=\begin{cases} q_j(t)-\lambda q_j(t) + \dfrac{\lambda}{K(t)} & \text{if } j\neq i \text{ and } d_j(t)>d_i(t) \\ q_j(t)-\lambda q_j(t) & \text{if } j\neq i \text{ and } d_j(t)\leq d_i(t) \\ 1-\sum_{k\neq i} q_k(t+1) & \text{if } j=i \end{cases}$$

(4)

where $K(t)$ represents the number of actions with higher estimates than $a(t)=a_i$ and λ is the learning rate of the LA.

$$
\begin{aligned}
&Q_i(t+1) = Q_i(t) + (1-\beta(t)) \\
&Z_i(t+1) = Z_i(t) + 1 \qquad (5) \\
&d_i(t+1) = \frac{Q_i(t+1)}{Z_i(t+1)}
\end{aligned}
$$

The detailed procedure of the generalized pursuit algorithm is given in Fig. 1. It should be noted that at the beginning, each action is chosen for a few times to obtain initial reward estimates. However, one can simply initialize *Z*(0) and *Q*(0) to some constant values like *c* (which will be used in the proposed method).

```
Algorithm 1.
Parameters and variables:
$r$          number of actions
$a(t)$       selected action at time step $t$. $a(t) \in \{a_1,a_2,\ldots,a_r\}$
$\lambda$    learning speed. $0<\lambda<1$
$Q_i(t)$     number of times the $i$th action is rewarded up to time $t$.
$Z_i(t)$     number of times that the $i$th action has been selected till now.
$\beta(t)$   reinforcement signal from the environment which represents the
             favorability of the selected action, where $\beta(t) \in \{0,1\}$
$q_i(t)$     selection probability of the $i$th action.
$d_i(t)$     current reward estimate for the $i$th action.
1.   Initialization:
     a.   Initialize $d(t)$ by choosing each action a small number of times.
     b.   Set the probability of each action to $1/r$.
2.   Until the termination condition is not satisfied repeat:
     a.   At time step $t$, select an action based on the action probability
          distribution $q$. Assume $a(t) = a_i$.
     b.   Let $K(t)$ represent the number of actions with higher estimates
          than $a_i$. update $q$ according to the following equation:
          $q_j(t+1) = \begin{cases} q_j(t) - \lambda q_j(t) + \frac{\lambda}{K(t)} & \text{if } j \neq i \text{ and } d_j(t) > d_i(t) \\ q_j(t) - \lambda q_j(t) & \text{if } j \neq i \text{ and } d_j(t) \le d_i(t) \\ 1 - \sum_{k \neq i} q_k(t+1) & \text{if } j = i \end{cases}$
     c.   Update reward estimate of the selected action i.e. $d_i(t)$ using the
          following equations:
          $Q_i(t+1) = Q_i(t) + (1-\beta(t))$
          $Z_i(t+1) = Z_i(t) + 1$
          $d_i(t+1) = \frac{Q_i(t+1)}{Z_i(t+1)}$
```

Fig. 1. A pseudo code for generalized pursuit algorithm

4. Maximum likelihood rotation matrix for multivariate Gaussian distribution

In bare bones PSO, as described in section 2.1, the new positions of the particles are generated based on the Gaussian distribution according to Eq. 2. The goal of the proposed method is to reorient the updating distributions of the particles along the most promising directions. The updating distributions as well as the promising directions for the particles will be described thoroughly in the subsequent sections. This section introduces a fundamental lemma which will be used in the proposed method and addresses the following issue: given an updating distribution and a promising direction for a particle, how can we reorient the distribution along the given direction? Accordingly, we are searching for a transformation which performs the desired reorientation. This problem can be directly solved by finding the transformation which maximizes the likelihood of generating samples along the given promising direction. However, this section introduces a general purpose theorem for the reorientation of Gaussian distributions. Then, the desired problem is stated as a special case of the obtained theorem.

In the general problem considered here, it is desired to rotate a given multivariate Gaussian distribution in order to maximize the likelihood of a given sample set like $X=\{x_1,x_2,\ldots,x_m\}$. Accordingly, it is required to find a rotation matrix like *R* which maximizes the likelihood function $L(R \mid X)$ defined as follows:

$$L(R \mid X) = L(R \mid x_1, x_2, ..., x_m) = \prod_{i=1}^{m} f(x_i \mid R) \quad (6)$$

where f equals to a multivariate density function with covariance matrix Σ and mean μ which is rotated around μ using the orthogonal matrix R.

Without loss of generality, we can assume $\mu=0$ as the distribution can be rotated around the origin and then transformed to its mean point.

Fact 1.

Let a be an $m\times1$ vector, R be an $m\times n$ matrix, and $x\sim MN(\mu,\Sigma)$. The distribution of the vector of random variables $a+Rx$ is $MN(a+R\mu, R\Sigma R^{T})$.

Theorem 1.

Let $x_i, i=1,...,m$ be n m-dimensional independent identically distributed vectors, and $S=\sum x_i x_i^{T}$; Assume $\Sigma, R, A, B \in \mathbb{R}^{n\times n}$, where Σ is symmetric positive definite, R is orthogonal, A and B are orthogonal matrices obtained from the Eigen decomposition of Σ^{-1} and S respectively in a way that $\Sigma^{-1}=A\Lambda_{\Sigma}A^{T}$ and $S=B\Lambda_{S}B^{T}$. Here, Λ_{Σ} and Λ_{S} are diagonal matrices with the diagonal entries arranged in non-decreasing order. The likelihood of the given set of samples with respect to the multivariate Gaussian distribution $MN(0,R\Sigma R^{T})$ is maximized if

$$R = BK_nA^{T} \quad (7)$$

where K_n is a reverse diagonal matrix with diagonal entries of ±1. In addition, we can always choose the sign of these entries to ensure $|R|>0$.

The proof of the theorem is provided in the Appendix A.

Corollary 1.

If Σ is diagonal and $m=1$, then R can be computed in $O(n^2)$.

As Σ is a diagonal matrix, its inverse is simply the inverse of its diagonal elements, which is also a diagonal matrix. Additionally, we have $I\Sigma^{-1}I=\Sigma^{-1}$ where I is the identity matrix. Hence, the columns of I can be considered as the eigenvectors of Σ^{-1}. We can rearrange the diagonal entities of Σ^{-1} in a non-decreasing order and their corresponding eigenvectors in $O(n^2)$. This rearrangement results in $A\Omega A^{T}=\Sigma^{-1}$, where A is a permutation matrix and Ω is a diagonal matrix. Multiplication of a diagonal matrix with a reverse diagonal one also can be performed in $O(n^2)$. As $m=1$, $S=xx^{T}$, its Eigen decomposition can be performed using the QR factorization of xx^{T} with Householder transformation in $O(n^2)$. Accordingly, The Eigen vector matrix of xx^{T} can be obtained using the following equation:

$$B = I - \frac{2}{\|x-I_1\|^2}(x-I_1)(x-I_1)^{T} \quad (8)$$

with $I_1 = (\|\mathrm{x}\|, 0, 0, ..., 0)^{T}$

It can be easily verified that B is orthogonal and $xx^{T} = B[I_1 I_1^{T}]B^{T}$.

Fig. 2 illustrates the aforementioned ideas. Each part of the figure represents a rotated multivariate

Gaussian distribution. The rotation matrix is defined in a way to maximize the likelihood of a Gaussian distribution with respect to a set of samples identified by 'o' markers in the figure. The parameters of the primary un-rotated Gaussian distributions were obtained using the points depicted by '+' markers in each part of the figure. These parameters were defined identical to classical BBPSO (Eq. 2), i.e. the mean of the distribution was set to the average of points, and its standard deviation to the distance of the points in each coordinate.

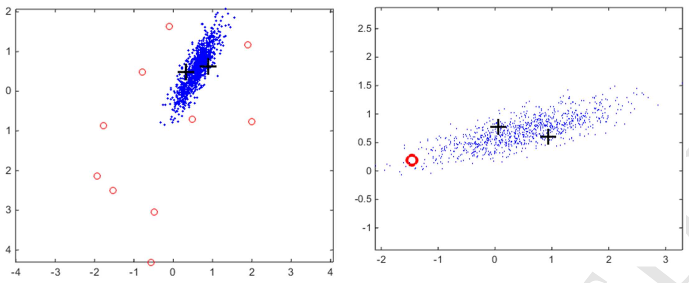

Fig. 2. Two multivariate Gaussian distributions rotated around their means to align with the samples depicted by 'o'.

5. CLA bare bones PSO with rotated mutations (CLA-BBPSO-R)

This section describes the details of the proposed new multi-swarm bare bones PSO and the ideas behind its development. The general idea of the suggested method is based on using the described theorem in section 4 to increase the likelihood of generating promising individuals in bare bones PSO. In this regard, Corollary 1 will be employed to rotate updating distributions along some promising directions. However, the difficulty with this approach lies in finding a promising rotation direction. One apparent choice of such a direction would be along the global best particle of the swarm. Our conducted experimental studies have shown that using this alignment direction only results in a premature convergence. The proposed optimization algorithm employs two techniques to deal with this problem: it uses multiple strategies and obtains promising directions through learning and incorporates multiple swarms to enhance diversity.

The formation of niches is a common phenomenon in biology. Different niches tend to have little impact on each other due to isolation; accordingly, the overall diversity of the population containing the whole niches tends to be high [43]. CLA-BBPSO-R utilizes the same concepts and uses a multi-swarm topology to enhance the diversity of its population. Like other swarm optimization approaches, it maintains a population of particles that iteratively search the problem landscape for preferable positions. This population is divided into several swarms arranged in a ring topology.

As mentioned previously, the utilization of simple alignment directions in the proposed method would result in premature convergence. In this work, we study the application of several promising alignment directions which are learned during the searching process. These directions are based on the historical paths that the particles have used to improve their previous positions as well as the paths that the particles can take to enhance their current positions.

Based on these promising paths, several strategies will be suggested for the distribution alignments. As various problems induce different fitness landscapes with different characteristics, the use of a single alignment direction would be unsuitable. The choice of an appropriate promising direction for a specific problem depends on the characteristics of its fitness landscape and the position of particles in this landscape. Accordingly, the utilization of adaptive alignment strategies would be helpful in dealing with fitness landscapes with various characteristics. Several strategy adaption mechanisms have been designed for the evolutionary approaches in the literature, which can be employed in the proposed method. Herein, we use a CLA based strategy adaption mechanism as it has already proven its effectiveness in multi-population approaches [17, 39]. In this model, each learning entity learns the best action based on local interactions and a set of local rules; accordingly, it can utilize the learned information from other nearby entities.

The block diagram of the proposed method is given in Fig. 3, and its building blocks are described in the subsequent sections. In addition, Table 1 summarizes the notations that will be used in the rest of the paper.

5.1. Action selection

As stated earlier, the selection of an appropriate alignment direction has an intense impact on the performance of the proposed method. The CLA incorporated into the model is responsible for adaptively learning the appropriate alignment directions for each particle during the search. CLA is a hybrid model based on learning automaton and cellular automaton. In this model, each LA beside its own experience can utilize the experience of other nearby entities.

CLA-BBPSO-R utilizes a CLA with *N* cells, which are arranged in a ring topology. Each cell contains a swarm of *M* particles as well as a group of *M* learning automata. There is a one to one correspondence between the learning automata and the particles within each cell, and each learning automaton is associated with a single particle. Each learning automaton controls the behavior of its associated particle by determining its updating strategies and alignment directions during the search process. The iterative process of CLA-BBPSO-R starts with the action selection phase (Fig. 3). During this phase, each learning automaton selects one of its actions according to its internal probability vector. In the proposed method, each selected action defines two terms for the associated particle: its alignment strategy and the strategy for computing the mean of its updating distribution. The particle associated with the learning automaton uses its selected action to determine its updating strategy and employs this strategy during the searching process.

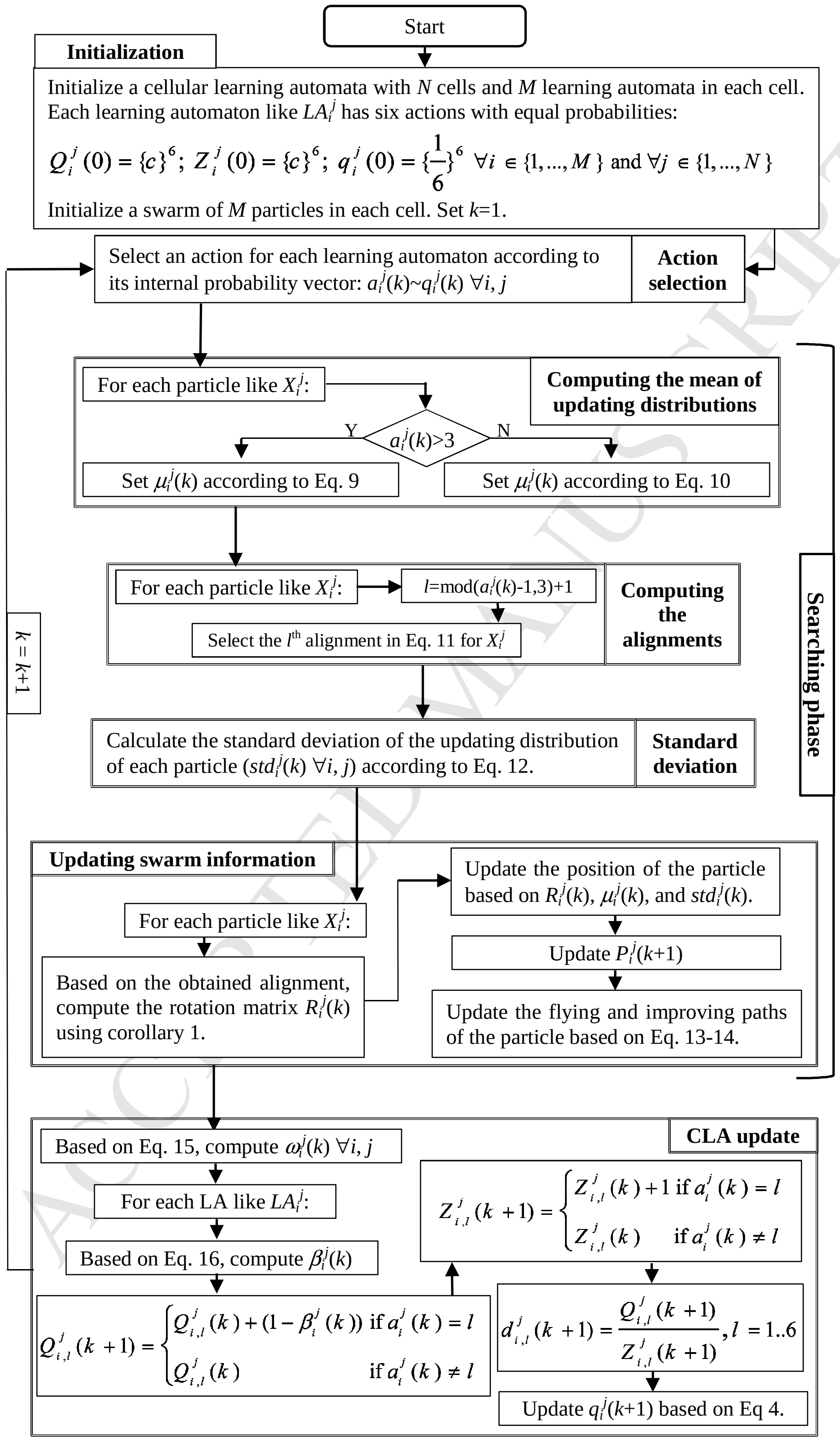
Start
Initialization
Initialize a cellular learning automata with N cells and M learning automata in each cell. Each learning automaton like LA_i^j has six actions with equal probabilities:
$Q_i^j(0)=\{c\}^6;\ Z_i^j(0)=\{c\}^6;\ q_i^j(0)=\{\frac{1}{6}\}^6\ \forall i\in\{1,...,M\}$ and $\forall j\in\{1,...,N\}$
Initialize a swarm of M particles in each cell. Set k=1.
Select an action for each learning automaton according to its internal probability vector: $a_i^j(k)\sim q_i^j(k)\ \forall i,j$
Action selection
For each particle like X_i^j:
Computing the mean of updating distributions
Y
$a_i^j(k)>3$
N
Set $\mu_i^j(k)$ according to Eq. 9
Set $\mu_i^j(k)$ according to Eq. 10
For each particle like X_i^j:
l=mod($a_i^j(k)$-1,3)+1
Select the l^{th} alignment in Eq. 11 for X_i^j
Computing the alignments
Calculate the standard deviation of the updating distribution of each particle ($std_i^j(k)\ \forall i,j$) according to Eq. 12.
Standard deviation
Searching phase
$k=k+1$
Updating swarm information
For each particle like X_i^j:
Based on the obtained alignment, compute the rotation matrix $R_i^j(k)$ using corollary 1.
Update the position of the particle based on $R_i^j(k)$, $\mu_i^j(k)$, and $std_i^j(k)$.
Update $P_i^j(k+1)$
Update the flying and improving paths of the particle based on Eq. 13-14.
CLA update
Based on Eq. 15, compute $\omega_i^j(k)\ \forall i,j$
For each LA like LA_i^j:
Based on Eq. 16, compute $\beta_i^j(k)$
$Z_{i,l}^j(k+1)=\begin{cases}Z_{i,l}^j(k)+1 & \text{if } a_i^j(k)=l\\ Z_{i,l}^j(k) & \text{if } a_i^j(k)\neq l\end{cases}$
$Q_{i,l}^j(k+1)=\begin{cases}Q_{i,l}^j(k)+(1-\beta_i^j(k)) & \text{if } a_i^j(k)=l\\ Q_{i,l}^j(k) & \text{if } a_i^j(k)\neq l\end{cases}$
$d_{i,l}^j(k+1)=\frac{Q_{i,l}^j(k+1)}{Z_{i,l}^j(k+1)},\ l=1..6$
Update $q_i^j(k+1)$ based on Eq 4.

Fig. 3. The block diagram of the proposed CLA-BBPSO-R

Table 1. Summary of the notations used in CLA-BBPSO-R

X_i^j	The i^{th} particle within the j^{th} swarm	p_i^j	Personal best position of the i^{th} particle within the j^{th} swarm
pg^j	The personal best position of the fittest particle in the j^{th} swarm	$gbi(j)$	The index of the best particle within the j^{th} swarm
v_i^j	Flying path of the i^{th} particle within the j^{th} swarm	π_i^j	The improving path of the particle X_i^j
N	Number of swarms (Number of cells in the CLA)	M	Number of particles within each swarm (Number of learning automata in each cell)
LA_i^j	The i^{th} learning automaton in the j^{th} cell	$N(j,i)$	The neighborhood function which returns the index of the i^{th} neighboring cell (swarm) to cell (swarm) j. Here $i\in\{1,2\}$.

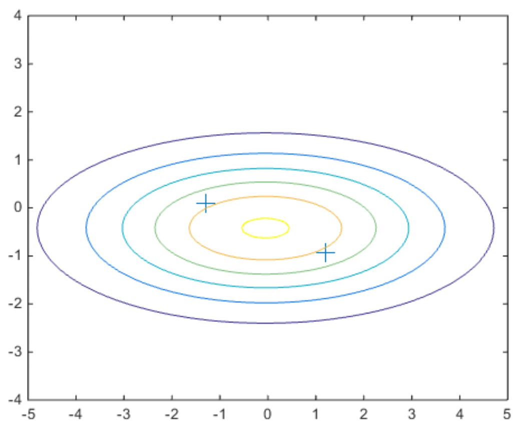

Fig. 4. The contours of the multivariate Gaussian distribution obtained by the update rule of classical bare bones PSO for two particles shown

5.2. Searching phase

After action selection phase, the algorithm enters a searching phase. During this phase, the positions of the particles are updated based on the selected actions (strategies). As depicted in Fig. 3, this phase can be divided into four related modules. In what follows these modules are discussed individually.

5.2.1. Computing the mean of updating distributions

In the classical BBPSO, the new position of a particle is randomly selected according to a Gaussian distribution with its mean being the average of two historical best positions: the best position of the particle and the best position of the whole swarm. The proposed method uses multiple swarms, which are arranged in a ring topology. Accordingly, the parameters of an updating distribution can be defined based on intra-swarm or inter-swarm information. In this regard, the mean of the updating distribution of a particle is defined based on two different schemes in the proposed method. The first scheme is based on intra-swarm information. Consequently for a particle like X_i^j, the mean of its updating distribution is defined as a weighted average of p_i^j and pg^j as follows:

$$\mu_i^j(k) = w_i p_i^j(k) + (1-w_i) p_r^j(k) \quad (9)$$

where

$$r = \begin{cases} \text{the index of the seond fittest particle in the } j^{\text{th}} \text{ swarm} & \text{if } i = gbi(j) \\ gbi(j) & \text{otherwise} \end{cases}$$

Here w_i is a decreasing weight, which linearly changes with i in the range [0.95,0.5]. Accordingly, the fitter particles have more influence on the mean of their updating distributions (here, we implicitly assumed that the particles within each swarm are sorted according to their fitness values). This definition of the mean point helps to preserve the promising information in the fitter particles. In Some developed BBPSO methods, μ_i is defined as μ_i=pg or

$\mu_i=p_i$ for each particle like X_i [16, 26]. Each one of these definitions has its benefits and drawbacks. The former favors the convergence speed of the algorithm, while the latter favors its exploration abilities. Eq. 9 defines the mean of the updating distributions as a combination of the classical definition and the one that defines $\mu_i(k) = p_i(k)$.

In the second scheme, a particle can employ information from its adjacent swarms. To this end, the mean point of an updating distribution is defined as follows:

$$\mu_i^j(k) = 0.5w_i\left(p_i^j(k) + pg^j(k)\right) + (1-w_i)X \quad (10)$$

where X is the fittest personal best position in the adjacent swarms.

During the searching phase, the mean point of the updating distribution of each particle is obtained based on one of the defined schemes. The selected action for a particle determines its choice of scheme in this module (as demonstrated in Fig. 3).

5.2.2. Computing the alignment directions

Classical BBPSO updates the position of the particles according to Eq. 2. Fig. 4 shows the contours of a typical distribution according to this updating rule. As it can be observed from the figure, the distribution always spreads along the axis. However, we can re-orient this distribution in a promising direction which facilitates the generation of fitter positions in the updating steps. The most promising alignment direction for an updating distribution is toward the optimum solution of the problem. However, this direction is unknown; accordingly, we will experiment other promising alignment directions for the distributions. The employed alignment directions in this work are based on two concepts: flying path and improving path. The flying path is the historical path a particle has taken during its search process. As this path has already improved the particle's position, it can define a promising direction for the particle in the consequent steps. The improving path for a particle is based on the position of fitter particles present in its beholding swarm. A particle can always fly toward a better particle to experience a better location. The flying path and the improving path of a particle are learned incrementally during the search process (as will be demonstrated in the consequent sections). The improving path and flying path associated with the ith particle within the jth swarm during step k will be denoted by $\pi_i^j(k)$ and $v_i^j(k)$ respectively. Based on these defined terms, we suggest three alignment strategies as follows:

$$\begin{aligned} D1_i^j &= p_i^j(k) + \kappa_1\pi_i^j(k) + \tau_1 v_i^j(k) \\ D2_i^j &= pg^j(k) + \kappa_2\pi_i^j(k) + \tau_2 v_{gbi(j)}^j(k) \\ D3_i^j &= \mu_i^j(k) + \kappa_3\pi_i^j(k) + \tau_3 v_i^j(k) \qquad (11) \\ &\text{and} \\ \kappa_l &= r_1;\ \tau_l = r_2 \end{aligned}$$

where r_1 and r_2 are two uniform random numbers in the interval [0,1]. κ_l and τ_l (l=1,2,3) are two parameters weighting the flying path and improving path in each of the alignment strategies. They are defined as uniform random numbers in the interval [0,1]. pg^j is the position of the global best particle of the jth swarm, and $gbi(j)$ represents the index of this particle within the respective swarm. $\mu_i^j(k)$ is the mean point of the Gaussian distribution which is used for updating the position of the particle at step k (as defined in Eq. 9-10). In these strategies it is assumed that by moving from $p_{ij}(k)$ ($pg^j(k)$ or $\mu_i^j(k)$) along its flying and improving paths, we can obtain new promising locations. These promising locations can be used to align the updating distributions using corollary 1.

In each step of the algorithm, a particle obtains the mean of its updating distribution according to one of the Eq. 9 or Eq. 10; then, uses one of the alignment strategies in Eq. 11 to obtain its alignment direction. As the most suitable alignment strategy is unknown a priori, the proposed method learns the most effective one based on a learning mechanism. Also it should be noted that, each one of the schemes for defining the mean points (Eq. 9-10) may lead to different behavior in the swarms, which may affect the learning process of the alignment directions. Therefore, the proposed method learns the direction alignment strategies alongside the two mechanisms for obtaining the mean values. By combining the three alignment strategies with the two defined equations for the mean values, six combined strategies can be considered in the proposed method. Consequently, each learning automaton is equipped with six actions, each one corresponding to one of the six choices to be made.

5.2.3. Computing the standard deviation of the updating distributions

Using the selected alignment strategy, a particle obtains a rotation matrix like $R_i^j(k)$ according to corollary 1 to align its updating distribution. The standard deviation of the updating distribution for a particle like X_i^j is defined similar to classical BBPSO (Eq. 2) with a slight modification as follows:

$$\sigma_i^j(k) = \left| x - y \right| \qquad (12)$$

where

$$x = R_i^j(k-1)^{\mathrm{T}}\left(p_i^j(k)\right)$$
$$y = R_i^j(k-1)^{\mathrm{T}}\left(pg^j(k)\right)$$

According to this equation, the standard deviation for a particle is calculated along its previously obtained promising alignment direction. This way, we can examine the spread of particles along various promising directions instead of a single direction which is aligned with axis. Also, it should be noted that for the best particle of each swarm, the second fittest particle of the swarm is used instead of pg^j in Eq. 12 (In order to avoid zero standard deviations).

5.2.4. Updating swarm information

After acquiring the parameters of the updating distributions and the related rotation matrices, the new position of each particle is obtained based on its calculated rotated Gaussian distribution. Then, the new positions are evaluated by the fitness function, and the historical best positions of the particles and the swarms are updated accordingly. In addition, the associated improving path and flying path of each particle like X_i^j is updated for the next iteration as follows:

$$\pi_i^j(k+1) = \alpha\pi_i^j(k) + \left(p_r^j(k) - p_i^j(k)\right) \qquad (13)$$

$$v_i^j(k+1) = \begin{cases} \alpha v_i^j(k) + \left(p_i^j(k) - p_i^j(k-1)\right) & \text{if } p_i^j(k) \neq p_i^j(k-1) \\ v_i^j(k) & \text{otherwise} \end{cases}$$
(14)

Here, π_i^j is the improving path associated with the ith particle within the jth swarm. p_r^j is the personal best location of a random intra-swarm particle fitter than p_i^j. p_r^j is randomly selected from the half fitter portion of the jth swarm. As there is no fitter particle than the global best particle of each swarm, the second fittest particle of each swarm is employed for updating the improving path of the best particle in the swarm. v_i^j is the flying path of the ith particle within the jth swarm. It is updated in the same manner as the improving path; however, a particle updates its flying path when it's personal best position changes. α is the averaging weight associated with the defined paths.

5.3. Updating cellular learning automata

After the search phase of the algorithm, the selected actions of the learning automata are evaluated, and the CLA will be updated accordingly (Fig. 3). CLA adjusts the selection probabilities of the considered actions (six combined strategies) for each particle. It accomplishes this goal through the repeated phrases of: action (strategy) selection, action execution (searching process), reinforcement signal generation, and learning automaton adaption. The generalized pursuit algorithm, which is illustrated in Fig. 1, will be used as the learning algorithm of each learning automaton.

In order to proceed, we first define the "improvement" criteria for the performance evaluation of particles as follows:

$$\omega_i^j(k) = f\left(p_i^j(k-1)\right) - f\left(x_i^j(k)\right) \qquad (15)$$

where $\omega_i^j(k)$ is a performance measure for the particle X_i^j during step k which determines the amount of fitness improvement in the particle. $x_i^j(k)$ is the new position obtained for the particle at step k according to the updating distribution.

CLA adapts its learning automata by observing the operation of their selected actions. After the action selection for a learning automaton, the selected action (updating strategy) is carried out by the corresponding particle. Then, reinforcement signals should be generated for the learning automaton in order to update its internal probability vector. In the proposed method, we evaluate the selected action of a learning automaton based on the performance of the particle employing the selected action. For this purpose, we employ the defined improvement criteria and the fitness of the particles. Based on this idea, a reinforcement signal is generated for a learning automaton like LA_i^j as follows:

$$\beta_i^j(k)=\begin{cases} 0 \text{ if } \begin{matrix}\left[\omega_i^j(k)>mean(\Omega^j(k))\right] \text{ or} \\ \left[f\ (x_i^j(k))>mean(\Psi^j(k))\right]\end{cases} \\ 1 \text{ otherwise}\end{cases} \quad (16)$$

where

$$\Omega^j(k)=\{\omega_k^l(k)\,|\,l=N(j,1) \text{ or } l=N(j,2)\}$$
$$\Psi^j(k)=\{f\ (x_k^l(k))\,|\,l=N(j,1) \text{ or } l=N(j,2)\}$$

Here $\beta_i^j(k)$ is the reinforcement signal generated for the i^{th} LA within the j^{th} cell. $\Omega^j(k)$ is the average improvement of the particles in the neighbouring cells of the j^{th} cell. $\Psi^j(k)$ is the average fitness of the particles in the neighbouring cells of the j^{th} cell. The reinforcement signal compares the improvement of the particle and the fitness of its acquired position with the corresponding ones in the neighbouring cells. Based on this generated signal, the LA will be updated according to Eq. 4-5. As demonstrated in Algorithm 1 and Fig. 3, each learning automata like LA_i^j has four 6-dimensional components, $q_i^j(k)$, $Q_i^j(k)$, $Z_i(k)$, and $d_i(k)$, that are updated in each step. A pseudo code for the overall proposed method is given in Fig. 5.

```
Algorithm 2: CLA-BBPSO-R
0.  Initialization:
      •  Set the values of the parameters $M$, $N$, $\alpha$, and $\gamma$.
      •  Initialize a cellular learning automata with $N$ cells and $M$ learning
         automata in each cell. Each learning automaton has six actions with
         equal probabilities. The learning speed of each LA is $\gamma$.
      •  Initialize $N$ swarm with $M$ particles in each swarm. Each swarm
         resides in one cell of the CLA.
1.  Set $k$=0 and $R_i^j=I$.
2.  While stopping condition is not satisfied do:
      a.  Set $k$=$k$+1.
      b.  Each learning automaton like $LA_i^j$ selects an action like $a_i^j(k)$.
      c.  For each particle like $X_i^j$, do:
            i.    Based on $a_i^j(k)$, obtain $\mu_i^j(k)$ and the alignment direction according
                  to the Eq. 9-11.
            ii.   Obtain the standard deviation of the updating distribution, $\sigma_i^j(k)$,
                  using Eq. 12.
            iii.  Using Corollary 1, obtain the Rotation matrix $R_i^j(k)$.
            iv.   Generate a new sample $x_i^j(k)$:
                  $x_i^j(k)=R_i^j(k)\left(\sigma_i^j(k)N(0,1)\right)+\mu_i^j(k)$.
            v.    Evaluate the newly generated sample and update $p_i^j$ and $pg^j$.
            vi.   update $v_i^j(k+1)$ and $\pi_i^j(k+1)$ according to Eq. 13-14.
      d.  For each LA like $LA_i^j$ do:
            i.    Compute $\omega_i^j(k)$ according to Eq. 15.
            ii.   Compute $\beta_i^j(k)$ according to Eq. 16.
            iii.  Update $LA_i^j$ based on Eq. 4 and Eq. 5.
```

Fig. 5. A pseudo code for the proposed method.

5.4. A modified version of the proposed method

As we intend to keep the number of parameters as small as possible, we didn’t considered the weights of the improving and flying paths (κ and τ) in Eq. 11 by setting κ and τ to random numbers. However, our empirical studies have shown that appropriate tuning of these parameters can significantly enhance the performance of the proposed method. In this section, we introduce an alternative approach which doesn’t require any specific parameter tunings. The values of κ and τ can be adaptively tuned during the searching process. To this extent, each particle uses its own settings of κ and τ for each one of its updating strategies, which are learned during the evolutionary process. A particle maintains a pool of different promising values of these parameters for each one of its six updating strategies. At each updating step, a particle selects one pair of these parameters from its parameter pool of the selected strategy. Then, slightly perturbs the selected values of κ and τ, and employs the obtained settings in its updating rule. If the newly generated settings improve the personal best position of the particle, the particle replaces the newly generated pair with a randomly selected old one from its parameter pool of the updating strategy. In the proposed method, the perturbed τ and κ are obtained according to the following equation:

$$\begin{aligned}\tau(new) &= \begin{cases} u(0,1) & \text{with probability } 0.1 \\ \max(\tau + 0.1N(0,1), 0.01) & \text{otherwise} \end{cases} \\ \kappa(new) &= \begin{cases} u(0,1) & \text{with probability } 0.1 \\ \max(\kappa + 0.1N(0,1), 0.01) & \text{otherwise} \end{cases}\end{aligned} \tag{17}$$

where $u(0,1)$ generates a uniform random number in the interval [0,1].

6. Experimental Study

In this section, the effectiveness of the proposed approach is investigated via a comparative study on the CEC2013 benchmark set [44]. The problems in this benchmarks set can be divided into three groups: unimodal (f_1-f_5), basic multimodal (f_6-f_{20}), and composition (f_{21}-f_{28}). As the definitions and the characteristics of these benchmarks are completely discussed in [44], they are not repeated further here. In order to demonstrate the impact of the proposed updating scheme on BBPSO, CLA-BBPSO-R is compared with two recent variants of bare bones PSO with classical updating rules. In addition, four recent variants of the ordinary PSO algorithm, namely DTT-PSO [45], MCUPS [46], HCLPSO [4], and EPSO [5] as well as CMAES [47] are also considered in the comparative studies. In the experimental studies of this section, all peer methods are configured with the same settings as the ones that are suggested in the related references. We have used the recommended settings for the 30-D search spaces on both 30-D and 50-D instances of the tested problems when there is no suggested settings for the 50-D search spaces in the related literature. A brief overview of the compared peer methods is given in Table 2.

Like other stochastic search methods, the proposed algorithm has some specific parameters. We can empirically analyze these parameters, and adjust them accordingly. In this regard, an investigation on the sensitivity of the proposed method to the different settings of the averaging weight (α) is carried out in section 6.4. Other parameters of the proposed method can be tuned in the same manner, and according to our experiments, the settings in Table 3 will be used for the rest of the paper.

Table 2. A brief description of the compared methods

Method	description
Cognitive BBJ (cBBJ) [26]	A variant of BBPSO with jumps, in which the current position of a particle is also involved in its updating rules.
PBBPSO [48]	PBBPSO uses a pairwise strategy to enhance the diversity of its swarm. The particles compete in random tournaments and, based on these competitions, are categorized as leaders or followers. Then, different search paradigms are used by the particles in each category.
DTT-PSO [45]	A dynamic tournament topology strategy is introduced in

	DTT-PSO to improve the performance of the PSO algorithm. In this method, each particle is guided by several better solutions, chosen from the entire population. The selection of the better particles is stochastic, but still favors particles with better solutions.
MCUPS [46]	A unification factor is integrated into particle swarm optimization to balance the effects of cognitive and social terms. The new model is hybridized with two other PSO methods, namely MPSO and CPSO to achieve better solutions.
CMA-ES [47]	CMA-ES stands for Covariance Matrix Adaptation Evolution Strategy. Evolution strategies (ES) are stochastic, derivative-free methods for numerical optimization.
HCLPSO [4]	HCLPSO uses two specific subpopulations for exploration and exploitation purposes and employs a comprehensive learning strategy to generate the exemplars for both of these subpopulations.
EPSO [5]	EPSO divides its population into two sub-swarms. A small sub-swarm searches the problem landscape based on comprehensive learning PSO (CLPSO) searching procedure, while an integrated self-adaptive mechanism learns an appropriate searching method for the second and larger sub-swarm.

Table 3. Parameter settings for the proposed method

Parameter	Value
N: Number of swarms (cells)	10
M: Number of particles (LA) in each swarm (cell)	10
α: averaging weight	0.8
λ: learning rate of LA	0.15
Parameter pool size of each particle in variant II	5

6.1. Comparison of different variants of the proposed method

In this section, we experimentally compare the two variants of the proposed method to identify the benefits of the adaptive parameter control component in CLA-BBPSO-R. The experiments in this section are carried out on the 30-D instances of the CEC2013 benchmarks. The results are obtained over 51 independent runs, and each run of the algorithm on each benchmark is terminated after 3E5 fitness evaluations. The obtained results are given in Table 4 in terms of the average error values of the final solutions and their standard deviations. The original variant of the proposed method will be denoted by CLA-BBPSO-R, and the variant with the adaptive parameter control component, which was introduced in section 5.4, will be denoted by CLA-BBPSO-R-A.

Table 4. Comparison results of the two variants of the proposed method on the CEC2013 benchmark set in terms of average and standard deviation of the obtained results.

f	CLA-BBPSO-R	CLA-BBPSO-R-A	f	CLA-BBPSO-R	CLA-BBPSO-R-A	f	CLA-BBPSO-R	CLA-BBPSO-R-A
f_1	**0.00E+00**	**0.00E+00**	f_{11}	**3.17E+01**	**3.17E+01**	f_{21}	2.75E+02	**2.59E+02**
	0.00E+00	0.00E+00		7.10E+00	4.91E+00		4.40E+01	4.97E+01
f_2	5.12E+04	**4.33E+04**	f_{12}	3.02E+01	**2.68E+01**	f_{22}	**2.36E+03**	2.60E+03
	2.12E+04	1.74E+04		6.49E+00	4.71E+00		3.68E+02	3.40E+02
f_3	3.43E+05	**4.90E+04**	f_{13}	6.62E+01	**6.08E+01**	f_{23}	2.93E+03	**2.85E+03**
	1.08E+06	9.35E+04		1.17E+01	1.20E+01		4.48E+02	3.70E+02
f_4	4.80E+02	**4.58E+02**	f_{14}	**2.76E+03**	2.78E+03	f_{24}	2.23E+02	**2.20E+02**
	1.56E+02	1.60E+02		5.10E+02	3.46E+02		6.96E+00	5.35E+00
f_5	**0.00E+00**	**0.00E+00**	f_{15}	2.87E+03	**2.77E+03**	f_{25}	2.67E+02	**2.64E+02**
	0.00E+00	0.00E+00		4.18E+02	3.90E+02		7.28E+00	1.18E+01
f_6	2.14E-04	**1.55E-05**	f_{16}	2.43E+00	**2.37E+00**	f_{26}	**2.00E+02**	**2.00E+02**
	1.02E-03	6.65E-05		2.58E-01	4.00E-01		1.17E-03	1.63E-03
f_7	1.38E+01	**1.14E+01**	f_{17}	5.52E+01	**5.36E+01**	f_{27}	5.80E+02	**5.48E+02**
	6.17E+00	4.51E+00		6.46E+00	5.18E+00		8.80E+01	7.48E+01
f_8	**2.09E+01**	**2.09E+01**	f_{18}	1.15E+02	**6.33E+01**	f_{28}	**2.96E+02**	3.00E+02
	6.63E-02	4.99E-02		5.63E+01	1.75E+01		2.80E+01	2.87E-13
f_9	1.85E+01	**1.82E+01**	f_{19}	2.23E+00	**1.64E+00**			
	2.37E+00	2.05E+00		3.43E-01	1.55E-01			
f_{10}	3.22E-02	**2.80E-02**	f_{20}	1.05E+01	**1.01E+01**			
	2.26E-02	1.44E-02		6.59E-01	5.48E-01			

The overall results of Table 4 show that the performances of the two variants of the introduced algorithm are very close, and their obtained results are similar on most of the tested problems. However considering the obtained results on f_3, f_{12}, f_{13}, f_{18}, f_{19}, f_{21}, and f_{27}, CLA-BBPSO-R-A demonstrates a robust performance in comparison to CLA-BBPSO-R. In the proposed method, the weights of the improving and flying paths (κ and τ) in the alignment strategies significantly affect the movements of a particle in the fitness landscape. Consequently, appropriate settings of these parameters can lead the algorithm to better solutions on different problems. In addition, the parameters κ and τ are changed randomly in CLA-BBPSO-R, which can result in a less steady behavior in the particles; however in CLA-BBPSO-R-A, these two parameters are adapted and changed smoothly, resulting in a stable behavior. Also, drastic changes in position of particles in CLA-BBPSO-R can severely affect the incremental learning process of the flying and improving paths.

6.2. Experiments on 30-D test problems

In this experiment, the performances of the peer methods are evaluated on the 30-D instances of the CEC2013 benchmarks. The results of the compared methods on each benchmark are obtained over 51 independent runs, and each run is terminated after 3E+5 fitness evaluations. To determine the superiority of the proposed approach from a statistical viewpoint, Wilcoxon rank sum test for a confidence level of 95% [49] has been conducted on each benchmark. Wilcoxon test is a nonparametric alternative to the paired t-test [49] and is based on the null hypothesis that the two compared algorithms are statistically equivalent. Table 5 summarizes the overall results of the tests as "*w*/*t*/*l*", which means that CLA-BBPSO-R-A is significantly better than, equal to and worse than the corresponding compared methods on *w*, *t*, and *l* functions, respectively. Also, the Freidman ranks of the compared methods based on the average obtained results are given in Table 6. The detailed results as well as the convergence curves of the compared methods are provided in the supplementary material.

Table 5. Summary of comparison results of CLA-BBPSO-R-A with other peer methods based on Wilcoxon rank sum test.

	Unimodal	Basic Multimodal	Composition	All
CBBJ	3/2/0	9/1/5	7/0/1	19/3/6
PBBPSO	3/2/0	12/2/1	6/1/1	21/5/2
MCUPS	2/2/1	11/1/3	6/1/1	19/4/5
DTT PSO	3/2/0	8/5/2	5/2/1	16/9/3
CMA-ES	0/2/3	13/0/2	7/1/0	20/3/5
HCLPSO	3/2/0	9/1/5	4/3/1	16/6/6
EPSO	2/2/1	10/1/4	4/3/1	16/6/6

Table 6. Friedman ranks of the compared methods on CEC2013 benchmarks at 30D. (CLA-BBPSO-R-A=CLA, PBBPSO=PBB, DTT PSO=DTT, HCLPSO=HCL)

f	CLA	CBBJ	PBB	MCUPS	DTT	CMAES	HCL	EPSO
f_1	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**
f_2	2	6	7	8	3	**1**	5	4
f_3	2	8	4	7	5	**1**	3	6
f_4	4	8	6	2	7	**1**	5	3
f_5	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**
f_6	**1**	7	6	8	4	2	5	3
f_7	**1**	8	5	7	6	2	3	4
f_8	**4**	**4**	**4**	**4**	**4**	8	**4**	**4**
f_9	**1**	8	6	7	2	4	3	5
f_{10}	2	8	4	3	6	**1**	7	5
f_{11}	5	**2**	7	4	6	8	**2**	**2**

***f*₁₂**	**1**	8	6	7	2	5	4	3
***f*₁₃**	**1**	8	6	7	2	5	4	3
***f*₁₄**	7	**1**	4	5	6	8	2	3
***f*₁₅**	**1**	7	8	6	2	3	4	5
***f*₁₆**	6	4	7.5	5	7.5	**1**	2	3
***f*₁₇**	5	**1**	8	6	4	7	2	3
***f*₁₈**	**1**	8	7	5	6	2	3	4
***f*₁₉**	3	**1**	8	6	5	7	2	4
***f*₂₀**	**1**	7	5	6	4	8	3	2
***f*₂₁**	3	6	8	7	4.5	4.5	**1**	2
***f*₂₂**	7	2	4	5	6	8	**1**	3
***f*₂₃**	2	7	8	6	**1**	3	4	5
***f*₂₄**	**1**	8	6	7	3	5	2	4
***f*₂₅**	**1**	8	6	7	4	2	3	5
***f*₂₆**	**2**	8	4	7	6	5	**2**	**2**
***f*₂₇**	**1**	8	6	7	3	5	2	4
***f*₂₈**	3.5	8	**1**	7	5	6	3.5	2
Avg	**2.76**	6	5.73	5.89	4.40	4.34	3.23	3.64

The overall results of Table 5-6 clearly demonstrate the effectiveness of the proposed method in solving optimization problems with different characteristics. CLA-BBPSO-R-A outperforms the other compared variants of BBPSO on most of the tested problems and obtains the lowest average rank according to the Freidman test. The main difference of the proposed approach with the other compared BBPSO methods lies in its updating scheme, which is based on rotated and re-aligned Gaussian distributions. Accordingly, it can be inferred that the proposed updating rule can generate new positions for the particles more effectively in comparison to the common updating rules of BBPSO. When compared with CBBJ, CLA-BBPSO-R has weaker performances on f_{11}, f_{14}, f_{17}, and f_{22}. Several compared methods have obtained better ranks than the proposed method on these problems. These problems enjoy some degree of separability among their variables, meaning that they can be solved by simple coordinate search approaches. The proposed method evolves all the coordinates of a particle in a same time based on a single unified probability distribution; accordingly, it cannot perform independent searches in different dimensions appropriately. There are various approaches to improve the proposed method in this regard; the simplest method would be integration of a jumping mechanism, similar to that of CBBJ, into CLA-BBPSO-R. This jumping mechanism is the main reason for the superior performances of CBBJ on f_{11}, f_{14}, f_{17}, and f_{22} in comparison to CLA-BBPSO-R and PBBPSO.

The proposed method also exhibits better performances in comparison with the variants of the canonical PSO algorithm. In general, advanced PSO methods with common updating rules involving velocity terms are superior to most of the BBPSO methods. This is due to the fact that common PSO updating rules are based on more effective information about the fitness landscape, which is encoded in the velocity terms. In the proposed method, the flying and improving paths of each particle play comparable rules to the velocity of particles in PSO and enhance the acquisition of promising information about the fitness landscapes.

MCUPS has a superior performance in comparison to the proposed method on f_4; however, it has achieved weaker or similar results on the other unimodal benchmarks. f_{11}, f_{14}, f_{16}, f_{22} are the multi modal and composition problems on which MCUPS, like CBBJ, has obtained better results in comparison to CLA-BBPSO-R, where the obtained results of the two methods are very close on f_{16}. DTT PSO, like MCUPS and CBBJ, has better performances in comparison to CLA-BBPSO-R on f_{14} and f_{22}, while it achieved weaker or very close performances on the reminder of the problems. The average ranks of HCLPSO and EPSO are very close to that of CLA-BBPSO-R-A. These two approaches obtained high Freidman ranks on the separable problems and demonstrated superior performances on some the tested multimodal and composition problems. However, considering pairwise comparisons based on Wilcoxon rank sum test, CLA-BBPSO-R-A demonstrated superior or equal performances in most cases.

The achieved results of CMA-ES are rather distinguishable from other compared methods. In this algorithm, the new individuals are generated by a single multivariate Gaussian distribution. Then, the covariance matrix of the distribution is updated incrementally in way to increase the likelihood of previously successful search steps. The algorithm evolves the parameters of the Gaussian distribution based on its whole population. Accordingly, CMA-ES exhibits very high convergence rates, which is noticeable from its achieved results on the unimodal problems. The proposed re-alignment scheme for updating distributions has a significant difference with the covariance adaptation scheme of CMA-ES: rather than approximating a single covariance matrix for the whole population, each individual adaptively learns its promising updating directions and based on these directions obtains a rotation matrix to adjust its updating distributions. As the computed directions are different for the various individuals in the population, the proposed method possesses better exploration capabilities in comparison to CMA-ES. Accordingly, the proposed method is superior in comparison to CMA-ES on the multimodal and composition problems. Considering the convergence curves provided in supplementary material, the higher convergence rate of CMA-ES is clearly evident. However, on problems like f_{11}, f_{13}, and f_{20} this higher convergence rate of CMA-ES results in a fast entrapment in local optima. CLA-BBPSO-R-A, also, exhibits appropriate convergence properties. Aside from CMA-ES, the proposed method reaches better solutions on problems like f_3, f_6, f_7, f_{10}, f_{13} faster than the other peer methods, and on problems f_{18} and f_{20} its convergence speed is comparable to DTT PSO, which exhibits high convergence rates on these problems.

6.3 Experiments on 50-D test problems

The summary of comparison results on the 50 dimensional instances of the CEC2013 benchmarks are given in Table 7. Additionally, the detailed statistical results are provided in the supplementary material. As it can be seen from the obtained results, the relative performances of the compared methods are much similar to the 30-D instances of the problems. The introduced method obtains relatively better results on f_4 due to the increased number of fitness evaluations. Also, the relative performance of HCLPSO and EPSO on the composition problems is improved in comparison to the 50-D search spaces, which demonstrates their efficiency on higher dimensional spaces. However based on the overall obtained results, it can be inferred that CLA-BBPSO-R is still an effective optimization method on higher dimensionalities, and its obtained results are better than the other compared methods on most of tested problems.

6.4. The effect of averaging weight (α) on the performance of CLA-BBPSO-R-A

In this section, we examine the effect of averaging weight on the performance of the proposed method. In this regard, the influence of different values of this parameter on a set of unimodal and multimodal problems is investigated. As it is clear from the obtained results in Fig. 6, various settings of this parameter exhibit different effects on the performance of CLA-BBPSO-R-A on different problems. For problems like f_3 and f_6, α=0.8 seems to be an appropriate choice, while for the others different settings exhibit better performances. The proposed method achieved better results on the unimodal problems f_2 and f_4 using relatively lower values of α, while on the multimodal benchmarks including f_7, f_9, and f_{12} CLA-BBPSO-R has distinctively better performance when α=0.95. On the two multimodal problems f_6 and f_{17}, the algorithm can obtain better results with the settings of α in the range [0.65,0.85].It should be noted that f_6 is a problem with the both unimodal and multimodal characteristics.

Table 7. Summary of comparison results of CLA-BBPSO-R with other peer methods based on Wilcoxon rank sum test on the 50D problems.

benchmarks	Unimodal	Basic Multimodal	Composition	All
CBBJ	3/2/0	9/1/5	7/0/1	19/3/6
PBBPSO	3/2/0	13/1/1	7/0/1	23/3/2
MCUPS	2/3/0	11/2/2	7/0/1	20/4/4
DTT PSO	3/2/0	10/3/2	4/3/1	17/8/3
CMA-ES	0/2/3	13/0/2	6/2/0	19/4/5
HCLPSO	3/2/0	9/1/5	3/2/3	15/5/8
EPSO	2/2/1	8/3/4	4/1/3	14/6/8

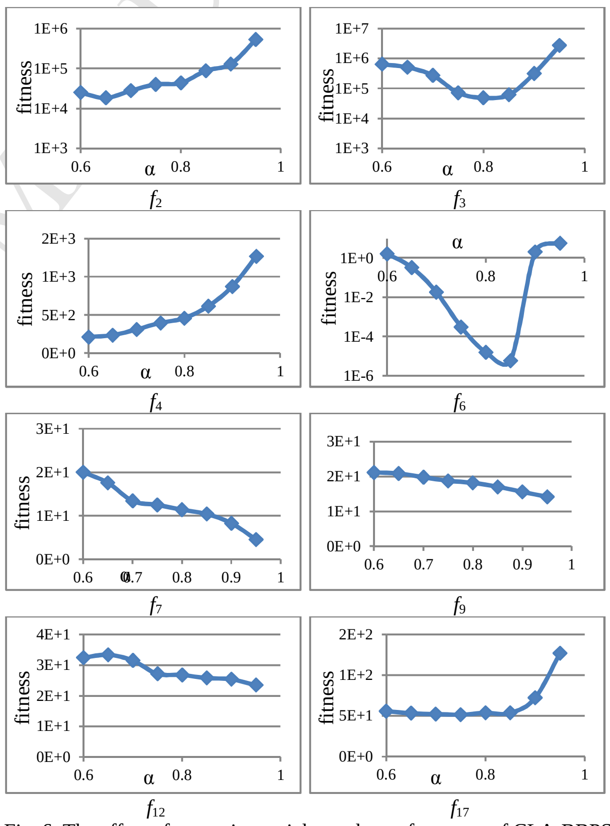

Fig. 6. The effect of averaging weight on the performance of CLA-BBPSO-R

From these observations, it could be inferred that decreasing α occasionally results in an increase in the convergence rate of the proposed algorithm;

accordingly, CLA-BBPSO-R can achieve better solutions on the unimodal problems with lower values of α. However, as α increases, the particles employ more information from their search histories in the generation of new positions, which can be helpful in escaping from local optima of multimodal problems. Consequently, adaptive and appropriate tuning of this parameter can significantly improve the performance of the proposed approach.

6.5. Investigating the alignment strategies of the CLA-BBPSO-R-A

This section investigates the effects of different proposed alignment strategies on the behaviour of CLA-BBPSO-R-A. To this end, two categories of experiments are conducted on the alignment strategies presented in Eq. 11. In the first category of experiments, the proposed method is modified to use only one of the presented strategies, and the rotation matrices are calculated based on that strategy. In the second category of experiments, a single alignment strategy is disabled each time, and the performance of the algorithm is evaluated in the absence of that strategy. In order to represent different variants of the proposed method, the notation BB-R:{S} will be employed in this section, where S represents the set of involved alignment strategies in an experiment. Fig. 7 illustrates the effectiveness of different alignment strategies.

Considering Eq. 11, the alignment strategies ‘1’, ‘2’, and ‘3’ are based on ‘pbest’, ‘gbest’, and ‘mean’ points respectively. It can be perceived from Fig. 7 that various alignment strategies have different impacts on the performance of CLA-BBPSO-R-A under different conditions. Alignment strategy ‘2’ is the most important and effective alignment among the three developed strategies. It is the best alignment strategy of CLA-BBPSO-R for the unimodal benchmarks, and the performance of CLA-BBPSO drops significantly on these benchmarks when this alignment strategy is disabled. Additionally, successful alignment strategy sets for the multimodal benchmark f_{17}, all, include strategy ‘2’. The successful performances of the proposed method on f_7 are achieved with ‘{1}’, ‘{1, 3}’, ‘{3}’, and {1, 2, 3}, which demonstrates the effectiveness of two alignment strategies ‘1’ and ‘3’ on this benchmark. The successful performances of CLA-BBPSO-R on f_{12} are with respect to the alignment sets {3}, {1, 3}, and {1, 2, 3}, which shows that alignment direction ‘3’ is the most suitable choice for this problem. On the benchmark f_9, all of the alignment directions lead to almost similar performances. Additionally, the algorithm works well on f_6 when it employs multiple strategies in combination.

Based on these observations, it could be inferred that different landscapes require different types of alignment strategies, and the performance of the proposed method significantly depends on its incorporated alignment strategies. Unimodal problems can be solved effectively by using schemes with high convergence and local search characteristics, and as the alignment strategy ‘2’ is based on ‘gbest’, it is the most effective alignment direction for the algorithm on the unimodal problems. However, multimodal benchmarks require various combinations of alignment strategies, which depend on the characteristics of the corresponding fitness landscapes.

6.6. Time complexity of CLA-BBPSO-R

In this section, we briefly investigate the time complexity of the proposed approach. The most time consuming operation in the proposed method is related to the rotation matrix of each particle. The rotation matrix is calculated in every updating step separately for each particle. As discussed in section 4, this operation can be implemented efficiently with a time complexity of O(D^2) where D represents the problem dimensionality. The other operations like action selection or LA adaption are less time consuming. Each LA has a constant number of actions; consequently, action selection step can be performed with a time complexity of O(1) for each particle. Also, the action adaption step imposes a time complexity of O(1) on each particle as the reinforcements signals for the learning automata of a cell with M learning automata can be generated in O(M) and, using these reinforcement signals, each LA can be updated in O(1).

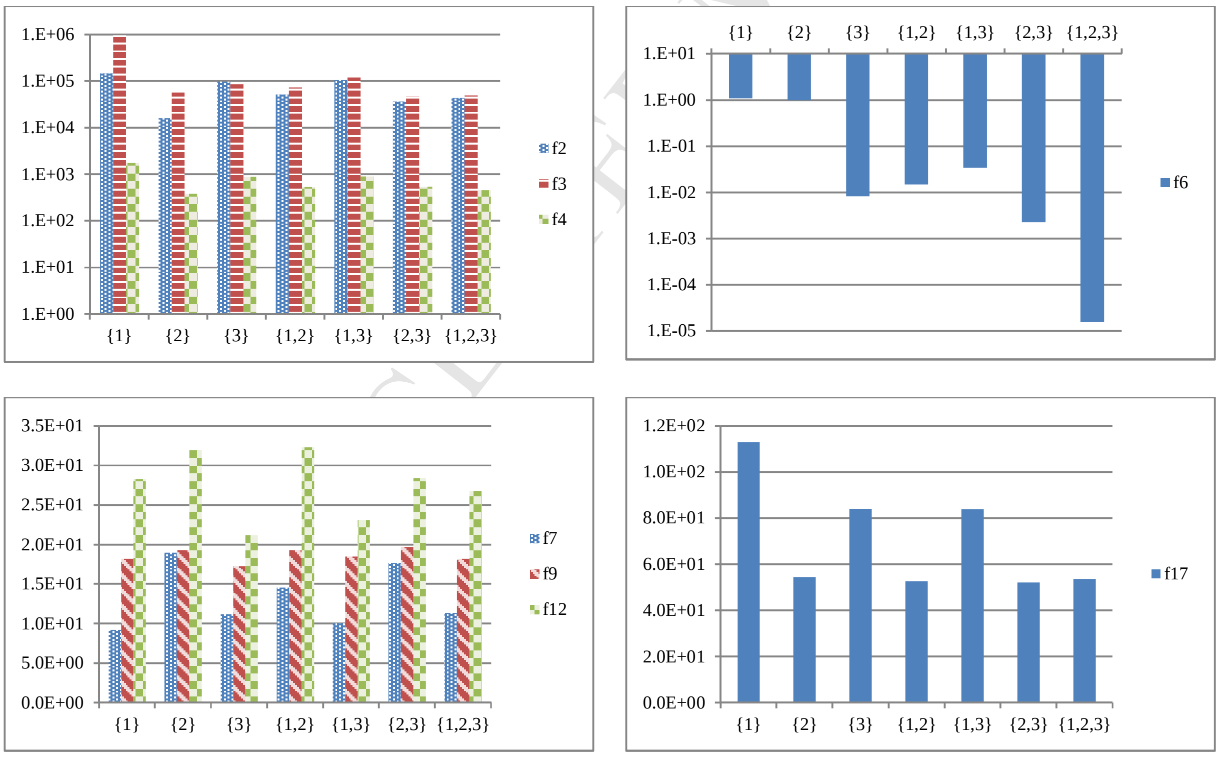

Fig. 7. The average obtained results of CLA-BBPSO-R under different integrated strategy sets.

In order to experimentally analyze the running time of the proposed method, the 30-*D* dimensional instances of the unimodal functions from the CEC2013 benchmark set are considered in this section. The running time of the proposed method is compared with two recent variants of the PSO algorithm, namely EPSO and HCLPSO. Table 8 gives the running time of the three compared methods after 3E+5 fitness evaluations. Looking at Table 8, it is obvious that the proposed method is slower than HCLPSO as HCLPSO has less overhead in comparison to the proposed method. However, CLA-BBPSO-R requires less running time in comparison to EPSO which uses an ensemble of several variants of the PSO algorithm. It should be noted that fitness evaluation is considered as the most time consuming operation in the optimization literature. Accordingly as the objective function becomes more time consuming, the running time of different optimization methods become closer.

Table 8. The run time of different methods on the 30-*D* unimodal benchmark problems (in seconds).

f	CLA-BBPSO-R-A	HCLPSO	EPSO
f_1	2.27e+01	**1.79e+01**	3.00e+01
f_2	2.45e+01	**1.95e+01**	3.36e+01
f_3	2.44e+01	**1.99e+01**	3.06e+01
f_4	2.37e+01	**1.90e+01**	3.36e+01
f_5	2.22e+01	**1.82e+01**	2.92e+01

7. Conclusion

This paper investigated a mechanism based on cellular learning automata and maximum likelihood principle for enhancing the updating distributions of BBPSO. To this end, a set of alignment strategies are developed and integrated into the classical bare bones PSO method. Based on this mechanism, the particles can utilize additional and promising information about the fitness landscape into their updating rules. The proposed method is based on the idea of re-aligning the updating distributions along some promising directions. Accordingly, the algorithm can generate new positions for the particles effectively in comparison to the classical updating rules of the bare bones PSO. These alignment directions are based on the two introduced concepts of flying and improving paths that each particle experiences during its searching process. The flying and improving paths for a particle are learned in the run time based on an incremental mechanism. The promising directions associated with these paths are learned for each particle separately via cellular learning automata. CLA achieves this goal through reinforcement learning and a set of local rules for evaluating the performance of different alignment strategies. The introduced new approach for the distribution alignments in bare bones PSO can be easily extended to other optimization methods, and is not specific to the proposed algorithm. Also, there are several ways to enhance the introduced method in the future works such as investigating and developing new alignment strategies. Comparative studies on the CEC2013 benchmark set demonstrated the effectiveness of the proposed approach in solving complex optimization problems. Also, comparison with other recent variants of BBPSO highlighted the superiority of the re-orientation accumulated updating rules to the classical updating rules of BBPSO.

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, 1995, pp. 1942–1948.

[2] P.J. Angeline, Using selection to improve particle swarm optimization, in: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, IEEE, 1998, pp. 84-89.

[3] L.T. Al-Bahrani, J.C. Patra, A novel orthogonal PSO algorithm based on orthogonal diagonalization, Swarm and Evolutionary Computation, 40 (2018) 1-23.

[4] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm and Evolutionary Computation, 24 (2015) 11-24.

[5] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Applied Soft Computing, 55 (2017) 533-548.

[6] W. Ye, W. Feng, S. Fan, A novel multi-swarm particle swarm optimization with dynamic learning strategy, Applied Soft Computing, 61 (2017) 832-843.

[7] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE transactions on Evolutionary Computation, 6 (2002) 58-73.

[8] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information processing letters, 85 (2003) 317-325.

[9] F. Van Den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information sciences, 176 (2006) 937-971.

[10] R. Poli, Mean and variance of the sampling distribution of particle swarm optimizers during stagnation, IEEE Transactions on Evolutionary Computation, 13 (2009) 712-721.

[11] F. Van den Bergh, A.P. Engelbrecht, A convergence proof for the particle swarm optimiser, Fundamenta Informaticae, 105 (2010) 341-374.

[12] J. Kennedy, Bare bones particle swarms, in: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, 2003, pp. 80-87.

[13] T.J. Richer, T.M. Blackwell, The Lévy particle swarm, in: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, IEEE, 2006, pp. 808-815.

[14] S.M. Elsayed, R.A. Sarker, T. Ray, Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2013, pp. 1932-1937.

[15] M. Campos, R.A. Krohling, I. Enriquez, Bare bones particle swarm optimization with scale matrix adaptation, IEEE transactions on cybernetics, 44 (2014) 1567-1578.

[16] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation, 16 (2012) 210-224.

[17] R. Vafashoar, M.R. Meybodi, Multi swarm bare bones particle swarm optimization with distribution adaption, Applied Soft Computing, 47 (2016) 534-552.

[18] C.-H. Chen, A variant of unified bare bone particle swarm optimizer, in: Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2013 International Conference on, IEEE, 2013, pp. 18-22.

[19] C.-H. Chen, Bare bone particle swarm optimization with integration of global and local learning strategies, in: Machine Learning and Cybernetics (ICMLC), 2011 International Conference on, IEEE, 2011, pp. 692-698.

[20] M. Campos, R.A. Krohling, Entropy-based bare bones particle swarm for dynamic

constrained optimization, Knowledge-Based Systems, 97 (2016) 203-223.

[21] J. Guo, Y. Sato, A dynamic allocation bare bones particle swarm optimization algorithm and its application, Artificial Life and Robotics, (2018) 1-6.

[22] H. Zhang, D.D. Kennedy, G.P. Rangaiah, A. Bonilla-Petriciolet, Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data, Fluid Phase Equilibria, 301 (2011) 33-45.

[23] H.-I. Hsieh, T.-S. Lee, A modified algorithm of bare bones particle swarm optimization, IJCSI, (2010) 11.

[24] R.A. Krohling, E. Mendel, Bare bones particle swarm optimization with Gaussian or Cauchy jumps, in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE, 2009, pp. 3285-3291.

[25] H. Liu, G. Ding, B. Wang, Bare-bones particle swarm optimization with disruption operator, Applied Mathematics and Computation, 238 (2014) 106-122.

[26] M.M. al-Rifaie, T. Blackwell, Cognitive Bare Bones Particle Swarm Optimisation with Jumps, International Journal of Swarm Intelligence Research (IJSIR), 7 (2016) 1-31.

[27] W. Gao, F.T. Chan, L. Huang, S. Liu, Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood, Information Sciences, 316 (2015) 180-200.

[28] K.S. Narendra, M.A. Thathachar, Learning automata: an introduction, Courier Corporation, 2012.

[29] S. Wolfram, A new kind of science, Wolfram media Champaign, 2002.

[30] H. Beigy, M.R. Meybodi, A mathematical framework for cellular learning automata, Advances in Complex Systems, 7 (2004) 295-319.

[31] M. Esnaashari, M. Meybodi, A cellular learning automata based clustering algorithm for wireless sensor networks, Sensor Letters, 6 (2008) 723-735.

[32] M. Esnaashari, M.R. Meybodi, Dynamic Point Coverage Problem in Wireless Sensor Networks: A Cellular Learning Automata Approach, Ad hoc & Sensor wireless networks, 10 (2010) 193-234.

[33] M. Esnaashari, M.R. Meybodi, A cellular learning automata-based deployment strategy for mobile wireless sensor networks, Journal of Parallel and Distributed Computing, 71 (2011) 988-1001.

[34] M. Esnaashari, M.R. Meybodi, Deployment of a mobile wireless sensor network with k-coverage constraint: a cellular learning automata approach, Wireless networks, 19 (2013) 945-968.

[35] H. Beigy, M.R. Meybodi, A self-organizing channel assignment algorithm: A cellular learning automata approach, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2003, pp. 119-126.

[36] H. Beigy, M.R. Meybodi, Cellular learning automata based dynamic channel assignment algorithms, International Journal of Computational Intelligence and Applications, 8 (2009) 287-314.

[37] R. Vafashoar, M. Meybodi, A.M. Azandaryani, CLA-DE: a hybrid model based on cellular learning automata for numerical optimization, Applied Intelligence, 36 (2012) 735-748.

[38] J.A. Torkestani, M.R. Meybodi, A cellular learning automata-based algorithm for solving the vertex coloring problem, Expert Systems with Applications, 38 (2011) 9237-9247.

[39] R. Vafashoar, M.R. Meybodi, Multi swarm optimization algorithm with adaptive connectivity degree, Applied Intelligence, 48 (2018) 909-941.

[40] M. Agache, B.J. Oommen, Generalized pursuit learning schemes: New families of continuous and discretized learning automata, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 32 (2002) 738-749.

[41] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Evolutionary Computation Proceedings, 1998. IEEE World Congress on

Computational Intelligence., The 1998 IEEE International Conference on, IEEE, 1998, pp. 69-73.

[42] M.A. Thathachar, P.S. Sastry, Estimator algorithms for learning automata, (1986).

[43] M. Tomassini, Spatially structured evolutionary algorithms: artificial evolution in space and time, Springer, 2006.

[44] 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, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, 201212 (2013) 3-18.

[45] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Applied Soft Computing, 48 (2016) 584-596.

[46] H.-C. Tsai, Unified particle swarm delivers high efficiency to particle swarm optimization, Applied Soft Computing, 55 (2017) 371-383.

[47] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary computation, 11 (2003) 1-18.

[48] J. Guo, Y. Sato, A pair-wise bare bones particle swarm optimization algorithm, in: Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on, IEEE, 2017, pp. 353-358.

[49] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research, 7 (2006) 1-30.

[50] J.R. Magnus, H. Neudecker, Matrix differential calculus with applications in statistics and econometrics, Wiley series in probability and mathematical statistics, (1988).

[51] D.S. Bernstein, Matrix Mathematics: Theory, Facts, and Formulas . Princeton reference, in, Princeton University Press, 2009.

[52] R. Bhatia, Matrix analysis, Springer Science & Business Media, 2013.

[53] D. Bayard, An Optimization Approach to Orienting Three Spacecraft Reaction Wheel Actuators with Application to the Europa Orbiter, (2000).

Appendix A. Proof of theorem 1.

This section describes the details of a proof for theorem 1. More information concerning matrix differentiation, calculus, useful relations and inequalities can be found in [50-51], theorems about the trace of matrices are also present in [52]. A useful trace optimization problems is considered in [53].

Theorem 1.

Let $x_i, i=1,..,m$ be n m-dimensional independent identically distributed vectors, and $S=\sum x_i x_i^{\mathrm{T}}$; $\Sigma, R, A, B \in \mathbb{R}^{n\times n}$, where Σ is symmetric positive definite, *R* is orthogonal, *A* and *B* are orthogonal matrices obtained from Eigen decomposition of Σ^{-1} and *S* respectively in a way that $\Sigma^{-1}=A\Lambda_\Sigma A^{\mathrm{T}}$, $S=B\Lambda_S B^{\mathrm{T}}$, Λ_Σ and Λ_S are diagonal matrices with the diagonal entries arranged in non-increasing (or non-decreasing) order. The likelihood of the given set of samples with respect to the multivariate Gaussian distribution $MN(0,R\Sigma R^{\mathrm{T}})$ is maximized if

$$R = BK_n A^{\mathrm{T}}$$

where K_n is a reverse diagonal matrix with entries ±1. In addition, we can always choose the sign of these entries to ensure $|R|>0$.

Proof. The likelihood function is

$$L = L(R \mid X) = L(R \mid x_1, x_2, \ldots, x_m) =$$
$$\prod_{i=1}^{m} f(x_i \mid R) =$$
$$\prod_{i=1}^{m} \left[\frac{1}{\sqrt{(2\pi)^n \mid \mathrm{R}\Sigma R^{\mathrm{T}} \mid}} \exp(-\frac{1}{2} x_i^{\mathrm{T}} (R\Sigma R^{\mathrm{T}})^{-1} x_i) \right]$$

$$\ln(L) = -\frac{1}{2} mn \ln(2\pi) - \frac{1}{2} m \ln \mid R\Sigma R^{\mathrm{T}} \mid - \frac{1}{2} \sum_{i=1}^{m} x_i^{\mathrm{T}} R \Sigma^{-1} R^{\mathrm{T}} x_i$$

Using $|R\Sigma R^{\mathrm{T}}| = |R| \times |\Sigma| \times |R^{\mathrm{T}}|$, $|R| = |R^{\mathrm{T}}|^{-1}$, and $\sum_{i=1}^{m} x_i^{\mathrm{T}} R \Sigma^{-1} R^{\mathrm{T}} x_i = \mathrm{trace}(R\Sigma^{-1} R^{\mathrm{T}} S)$ the log likelihood function can be written in the following form:

$$\ln(L) = -\frac{1}{2} mn \ln(2\pi) - \frac{1}{2} m \ln |\Sigma| - \frac{1}{2} trace(R\Sigma^{-1} R^{\mathrm{T}} S)$$

Accordingly, the maximization of ln(L) results in the following problem:

$$\begin{aligned} &\min\ \mathrm{trace}(R\Sigma^{-1} R^{\mathrm{T}} S) \\ &\text{subject to}: R \text{ is orhtogonal, i.e. } RR^{\mathrm{T}} = R^{\mathrm{T}} R = I \\ &\text{and } |R| = 1 \end{aligned}$$

The value of $\mathrm{trace}(R\Sigma^{-1} R^{\mathrm{T}} S)$ can be obtained in the following way:

$$\mathrm{trace}(R\Sigma^{-1} R^{\mathrm{T}} S) = \mathrm{trace}(R\Omega R^{\mathrm{T}} S) \text{ where } \Omega = \Sigma^{-1}$$
$$\mathrm{trace}(R\Omega R^{\mathrm{T}} S) = \mathrm{trace}(RA\Lambda_\Omega A^{\mathrm{T}} R^{\mathrm{T}} B \Lambda_S B^{\mathrm{T}})$$

where Λ_Ω and Λ_S are diagonal matrices consisting of Eigen values of Ω and S respectively which are arranged in non-decreasing order; A and B are Eigen vector matrices associated with Λ_Ω and Λ_S.

$$\mathrm{trace}(RA\Lambda_\Omega A^{\mathrm{T}} R^{\mathrm{T}} B \Lambda_S B^{\mathrm{T}}) =$$
$$\mathrm{trace}(\Lambda_\Omega A^{\mathrm{T}} R^{\mathrm{T}} B \Lambda_S B^{\mathrm{T}} RA) = \mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M)$$

where $M=B^{\mathrm{T}}RA$. ($MM^{\mathrm{T}}=B^{\mathrm{T}}RAA^{\mathrm{T}}R^{\mathrm{T}}B=M^{\mathrm{T}}M=I$), using equation $\mathrm{trace}(AX^{\mathrm{T}}B) = \sum_i \sum_j \sum_k A_{ij} X_{kj} B_{ki}$, the value of the above trace can be obtained as:

$$\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \sum_i \sum_j \sum_k [\Lambda_\Omega]_{ij} [M]_{kj} [\Lambda_S M]_{ki}$$
$$\text{as } [\Lambda_S M]_{ki} = [\Lambda_S]_{kk} [M]_{ki}$$
$$\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \sum_i \sum_j \sum_k [\Lambda_\Omega]_{ij} [M]_{kj} [\Lambda_S]_{kk} [M]_{ki}$$
$$\text{as } [\Lambda_\Omega]_{ij} = 0 \text{ for } i \neq j$$
$$\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \sum_i \sum_k [\Lambda_\Omega]_{ii} [M]_{ki}^2 [\Lambda_S]_{kk}$$

defining N as $N=M\odot M$ where $\odot$ stands for the Hadamard operator, we arrive at:

$$\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \sum_i \sum_k [\Lambda_\Omega]_{ii} N_{ki} [\Lambda_S]_{kk}$$
$$\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \mathrm{diag}(\Lambda_S)^{\mathrm{T}} N \mathrm{diag}(\Lambda_\Omega)$$

It can be shown that N is doubly stochastic. If we adopt $a^{\mathrm{T}}=\mathrm{diag}(\Lambda_S)^{\mathrm{T}}$, and $b = \mathrm{diag}(\Lambda_\Omega)$ the value of the trace equals:

$$\begin{aligned} &\mathrm{trace}(\Lambda_\Omega M^{\mathrm{T}} \Lambda_S M) = \\ &a_1(N_{11}b_1 + N_{12}b_2 + \ldots + N_{1n}b_n) + \\ &a_2(N_{21}b_1 + N_{22}b_2 + \ldots + N_{2n}b_n) + \\ &\ldots \\ &a_n(N_{n1}b_1 + N_{n2}b_2 + \ldots + N_{nn}b_n) = \\ &b_1(N_{11}a_1 + N_{21}a_2 + \ldots + N_{n1}a_n) + \\ &b_2(N_{12}a_1 + N_{22}a_2 + \ldots + N_{n2}a_n) + \\ &\ldots \\ &b_n(N_{1n}a_1 + N_{2n}a_2 + \ldots + N_{nn}a_n). \end{aligned}$$

with $\sum_i N_{ij} = \sum_j N_{ij} = 1,\ and\ N_{ij} \geq 0,\ \forall i, j$.

In the above equation, each a_i is multiplied with the weighted average of b_js (j=1…n) and also each b_i is multiplied with the weighted average of a_js (j=1…n). Accordingly, their sum is minimized if each element in a is paired with its respective element in $\bar{b}$, where $\bar{b}$ consists of entities in b sorted in the non-decreasing order. Hence N=J, where J is the inverse identity matrix, minimizes the summation given in the previous expression. More specifically, if χ_n represents the set of n×n doubly stochastic matrices, χ_n is a convex compact set whose extreme points are the permutation matrices (Birkhoff), since $a^{\mathrm{T}}Nb$ is linear, its minimum is reached in an extreme point of χ_n, i.e. in a permutation matrix. Then, using Hardy's theorem, we can arrive at J. So, if the value of R can be chosen in a way to make N=J, the summation in the previous equation attains its minimum. By setting $R = BK_nA^{\mathrm{T}}$ we have:

$$N = M \square\ M = B^{\mathrm{T}}RA \square\ B^{\mathrm{T}}RA = \\ B^{\mathrm{T}}BK_nA^{\mathrm{T}}A \square\ B^{\mathrm{T}}BK_nA^{\mathrm{T}}A = K_n \square\ K_n = J$$

Considering RR^{T}=I as a Lagrangian, we can observe that $R = BK_nA^{\mathrm{T}}$ is an extreme point of the following Lagrangian function:

$$\psi(R) = trace(R\Sigma^{-1}R^{\mathrm{T}}S) - trace\left[L'(RR^{\mathrm{T}} - I)\right]$$

where L is the matrix of Lagrangian multipliers. Differentiating this Lagrangian function yields:

$$\frac{\partial\psi(R)}{R} = S^{\mathrm{T}}R(\Sigma^{-1})^{\mathrm{T}} + SR\Sigma^{-1} - (L+L')R$$
$$\frac{\partial\psi(R)}{R} = 2SR\Sigma^{-1} - (L+L')R$$

Accordingly, the first-order conditions are

$$2SR\Sigma^{-1} - (L+L')R = 0$$
$$RR^{\mathrm{T}} = I$$

Right multiplying the first equation with R^{T} and subtracting from its transpose yields:

$$2SR\Sigma^{-1} - 2\Sigma^{-1}R^{\mathrm{T}}S = 0 \\ RR^{\mathrm{T}} = I \qquad (31)$$

For which $R = BK_nA^{\mathrm{T}}$ is an answer.

- This paper develops a new multi-swarm Bare Bones PSO which utilizes a cellular learning automata to adjust the alignment direction of updating distributions.
- A theorem for the maximum likelihood estimate of rotation matrix in multivariate Gaussian distributions is developed.
- Based on the estimated rotation matrix, novel updating rules are developed for Bare Bones PSO with the aim of maximizing the likelihood of promising directions.
- By adaptively controlling the effect of each rule on each particle, the proposed method guides the updating direction of the particles.