Composite Particle Swarm Optimizer With Historical Memory for Function Optimization

Jie Li, JunQi Zhang, *Senior Member, IEEE*, ChangJun Jiang, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Particle swarm optimization (PSO) algorithm is a population-based stochastic optimization technique. It is characterized by the collaborative search in which each particle is attracted toward the global best position (gbest) in the swarm and its own best position (pbest). However, all of particles' historical promising pbests in PSO are lost except their current pbests. In order to solve this problem, this paper proposes a novel composite PSO algorithm, called historical memory-based PSO (HMPSO), which uses an estimation of distribution algorithm to estimate and preserve the distribution information of particles' historical promising pbests. Each particle has three candidate positions, which are generated from the historical memory, particles' current pbests, and the swarm's gbest. Then the best candidate position is adopted. Experiments on 28 CEC2013 benchmark functions demonstrate the superiority of HMPSO over other algorithms.**

***Index Terms*—Estimation of distribution algorithm (EDA), historical memory, particle swarm optimization (PSO).**

I. Introduction

PARTICLE swarm optimization (PSO) [1], [2] is a population-based stochastic optimization technique, and has been used to solve a wide variety of continuous and discrete optimization problems. PSO contains a swarm of particles and each particle that corresponds to a potential solution can fly in a search space with its velocity. Its significant characteristic is that all particles follow collaborative search in which each particle is attracted toward the global best position (gbest) in the swarm and its own best position (pbest). In recent years, improving PSO's performance by designing different methods has been an active research topic. Reference [3] presents a cooperative particle swarm optimizer (CPSO) that uses multiple swarms to optimize different components of the solution vector cooperatively. Reference [4] presents a fully informed particle swarm (FIPS) that uses all the neighbors to influence the flying velocity. Reference [5] presents a self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients (HPSO–TVAC) that uses time-varying acceleration constants. Reference [6] presents a comprehensive learning particle swarm optimizer (CLPSO) that uses all particles' current pbests to update the velocity of any one particle. Reference [7] presents a PSO with an aging leader and challengers (ALCPSO) by using an aging mechanism. Reference [8] presents a self-government PSO in which each particle updating depends on local best information searched at the last iteration as well as pbest and gbest. It is applied to solve a texaco gasification problem. Reference [9] presents two PSO-based multiobjective feature selection algorithms in classification. Reference [10] presents a co-evolutionary PSO algorithm associating with the artificial immune principle. It is verified in multiparameter estimation of permanent magnet synchronous machines. Reference [11] presents a hybrid PSO and genetic algorithm to solve a multiple unmanned aerial vehicle formation reconfiguration problem. Reference [12] presents a novel PSO method using swarm intelligence to solve an optimal power flow problem with distributed generator failures in power networks.

Although the aforementioned algorithms have obtained satisfactory results, they tend to strike in the local optimum. A key reason behind these PSOs is that they use only the information of gbest and a particle's own current best position (pbest), while they fail to utilize all of particles' historical promising pbests except their current pbests. In this case, a particle will move in a local optimum and it may be impossible to jump out of the local optimum area once its pbest falls into the same local optimum region where the gbest locates.

Intuitively, the historical memory of particles can help PSO to overcome the drawback and improve performance. The reason is that the implicit or explicit historical memory allows PSO to store promising solutions and reuse their information later so as to improve the search process. On one hand,

Manuscript received January 27, 2015; accepted April 9, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61272271, Grant 61332008, and Grant 91218301, in part by the NSF of USA under Grant CMMI-1162482, in part by the National Basic Research Program of China (973 Program) under Grant 2014CB340404, in part by the Natural Science Foundation Program of Shanghai under Grant 12ZR1434000, and in part by the International Cooperation Project of Chinese Ministry of Science and Technology under Grant 2012DFG11580. This paper was recommended by Associate Editor Jun Zhang. *(Corresponding authors: JunQi Zhang and ChangJun Jiang.)*

J. Li, J. Q. Zhang, C. J. Jiang, and M. C. Zhou are with the Department of Computer Science and Technology, Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China (e-mail: lijietjsh@gmail.com; zhangjunqi@tongji.edu.cn; cjjiang@tongji.edu.cn).

M. C. Zhou is also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2424836

the historical memory preserves the information of optimum solution space that particles have searched, which enable particles to return to these optimum regions if they fall into local optima. On the other hand, the historical memory that derives from different particles can also maintain the diversity of population to some extent.

However, traditional historical memory methods store a great number of promising solutions, thereby requiring a large memory space and a complex memory management scheme. Estimation of distribution algorithm (EDA) introduced in [13] has good potential to solve this problem. EDA belongs to a class of probability model-based evolutionary algorithms (EAs), where the processes of learning and sampling the probability model replace the crossover and mutation operations in a conventional genetic algorithm. A probability model indicates the joint probability distribution of high-performance solutions, i.e., it characterizes the set of promising solutions. If the historical information could be stored as probability models, we would not only save the memory space drastically but also simplify the memory management scheme. Consequently, EDA may play the role of historical memory in PSO.

Next, we consider how can one dig out and use the historical information to advance PSO? To answer it, this paper proposes a novel composite PSO algorithm, called historical memory-based PSO (HMPSO), which uses EDA to estimate and preserve distribution information of particles' historical promising pbests, then forms historical memory information, and finally integrates it into PSO to guide the search. Each particle is expected to have more comprehensive learning and search ability with the help of historical memory information. So far, there are three types of useful information existing in PSO, i.e., historical memory, particles' current pbests, and the swarm's gbest. Thus, how to systematically and best use them becomes a thorny and interesting problem. To solve it, we define a new particle generation mechanism whose primary idea is to generate a new particle through competing among three candidate particles that are generated by the information at each iteration.

In order to show the advantages of HMPSO, we compare it with CPSO-H6 [3], CLPSO [6], ALCPSO [7], FIPS [4], HPSO–TVAC [5], EDA–PSO [14], and particle swarm EDA (PSEDA) [15]. These algorithms are popular and their performance is better than most PSO algorithms. Moreover, two HMPSO variants are proposed and compared. They build the historical memory via a local distribution model for each particle and a multivariate Gaussian distribution for the population, respectively. Experiments on 28 CEC2013 benchmark functions demonstrate the superiority of HMPSO over them due to the use of historical memory and the related particle generation mechanism.

This paper is organized as follows. In Section II, traditional PSO, CLPSO, and EDA algorithms are introduced together with the similar state-of-the-art techniques using memory-based EAs and the hybridization of PSO and EDA. The proposed HMPSO is presented in Section III. Extensive simulation results are presented in Section IV. Finally, the conclusion is given in Section V.

II. Related Works

A. PSO and Comprehensive Learning PSO

PSO [1], [2] has been in existence as an EA for roughly twenty years, and has shown excellent performance in many application domains since it owns a simple social learning model. In PSO, a particle represents a potential solution in a search space, and many particles form a swarm. Each particle adjusts dynamically its flying direction according to the gbest of the swarm and the experience of personal best position (pbest).

Initially, PSO is equipped with a swarm of Q possible particles randomly generated from a search space. The position and velocity of the ith particle are updated at each iteration by the following rules for $d = 1, 2, \ldots, D$:

$$V_i^d = \omega \times V_i^d + c_1 \times r1_i^d \times \left(\text{pbest}_i^d - X_i^d\right) + c_2 \times r2_i^d \times \left(\text{gbest}^d - X_i^d\right) \quad (1)$$

$$X_i^d = X_i^d + V_i^d \quad (2)$$

where $\mathbf{X}_i = (X_i^1, X_i^2, \ldots, X_i^D)$ is the position of the ith particle, $\mathbf{V}_i = (V_i^1, V_i^2, \ldots, V_i^D)$ is its velocity, $\mathbf{pbest}_i^d = (\text{pbest}_i^1, \text{pbest}_i^2, \ldots, \text{pbest}_i^D)$ is its personal best position, $\mathbf{gbest}^d = (\text{gbest}^1, \text{gbest}^2, \ldots, \text{gbest}^D)$ is the global best position discovered by the whole swarm, D is the dimension of the problem to be optimized, $r1_i^d$ and $r2_i^d$ are two random numbers in the range [0, 1] for particle i, c_1 and c_2 are acceleration constants, and ω is inertia weight that is used to balance the global and local search.

PSO has a serious defect, i.e., it is easy to be trapped into a local optimum when it solves multimodal problems [6], [16]. CLPSO [6] is proposed to solve such premature convergence problem. It ensures the diversity of swarm because all particles' pbests are used to update the velocity of any one particle. The velocity updating equation is

$$V_i^d = \omega \times V_i^d + c \times r_i^d \times \left(\text{pbest}_{f_i(d)}^d - X_i^d\right) \quad (3)$$

$$X_i^d = X_i^d + V_i^d \quad (4)$$

where $\text{pbest}_{f_i(d)}^d$ is the dth component of any particle's pbest in the swarm, $\mathbf{f}_i = \{f_i(1), f_i(2), \ldots, f_i(D)\}$ is the function that decides which particle's pbest in the swarm particle i should follow, r_i is a random number in the range [0, 1] for particle i, and c is an acceleration constant. CLPSO is described as follows.

1) Select two particles from the current swarm randomly excluding particle i whose velocity is to be updated.
2) Compare the fitness values of two selected particles' pbests through a tournament selection procedure and select the better one.
3) Use the winner's pbest as an exemplar to update velocity (3) for the corresponding component.

B. Estimation of Distribution Algorithm

EDA [13], [17] is a new class of evolutionary computation methods. The main difference between EDA and traditional evolutionary computation algorithms is that the interrelations

in EDA are expressed explicitly through the joint probability distribution associated with the individuals selected by a certain selection procedure from the previous generation, while the interactions in the latter are implicitly kept. New offspring in EDA are generated by sampling the probability distribution that is estimated from the selected individuals of the previous generation. The most difficult work in EDA is the estimation of the joint probability distribution associated with selected individuals. The following is a description of EDA.

1) Generate the initial population randomly.
2) Select promising individuals from the population according to their fitness values by using a certain selection procedure.
3) Estimate the joint probability distribution of the selected promising individuals.
4) Sample new offspring individuals according to the joint probability distribution.
5) Use new sampled offspring individuals to replace some worse individuals in the previous population.

C. Memory-Based Evolutionary Algorithms

Memory is an essential feature of all living systems and a significant part of physical, chemical, and engineering systems. EAs are inspired by a biological evolutionary process in nature. From the evolutionism point of view, the effects of memory can be used in the process of EAs.

In the past few decades, there was a great deal of meaningful work in this field. For example, the work in [18] presents a memory mechanism by introducing memory modules into a membrane algorithm for solving knapsack problems. Reference [19] presents a stochastic evolution algorithm for solving multiobjective shortest path problems by using memory efficiently. Reference [20] presents a modified harmony search algorithm together with a new memory consideration scheme based on a roulette wheel mechanism. It is applied to solve economic load dispatch and combined economic and emission load dispatch problems. Reference [21] investigates the evolving ability of a cellular automaton with a type of memory based on the least mean square algorithm. Reference [22] presents a memory-efficient stochastic evolution-based algorithm for solving multiobjective shortest path problems. Reference [23] presents a clonal selection subpixel mapping framework by building a memory cell population. Reference [24] presents a directional feature in standard covariance matrix adaptation and evolution strategy by utilizing potentially useful memory information from the previous generation. Reference [25] presents several memory-based multiobjective EAs and applies them to solve multiobjective dynamic optimization problems. Reference [26] presents a model of an evolutionary game with a memory mechanism. Reference [27] presents a new model of memory in cellular learning automata-based evolutionary computing, which is used to address those time-varying optimization problems.

Although these studies have achieved great success, a challenge still remains, i.e., their methods require a large memory space and a complex memory management scheme. In order to solve this problem to some extent, EDA is a potential algorithm to store historical information by using its probability models, which can not only save the memory space effectively, but also simplify the memory management scheme greatly. Among the attempts to use EDA for this purpose, the work in [28] presents a reactive and EA, which modifies tentative solutions by local search with memory guided by EDA. Reference [29] presents an environment identification-based memory management scheme by EDAs for solving dynamic optimization problems.

D. Hybridization of PSO and EDA

Memory is a very effective mechanism to enhance the performance of EAs. EDA is an algorithm to store and build historical information by its probability model. Next, we will introduce the existing hybrid algorithms of PSO and EDA to show how EDA is used in PSO.

Liu *et al.* [30] presented hybrid PSO–EDA to solve a permutation flowshop scheduling problem through the sharing of information from the collective experience of EDA and PSO. Zhou *et al.* [31] developed a discrete estimation of distribution PSO algorithm to solve combinatorial optimization problems by utilizing the statistical information collected from particles' current pbests. Wang *et al.* [32] presented a discrete PSO algorithm based on EDA to solve polygonal approximation problems by incorporating the global statistical information collected from EDA into PSO, and generating a new particle via the random use of the information from PSO or EDA. Wang *et al.* [33] proposed a novel discrete quantum behaved PSO based on EDA for the combinatorial optimization problem in which EDA is used to extract global statistical information of current promising solutions. Wang [34] presented a modified genetic PSO based on EDA for combinatorial optimization problems where EDA is used to collect the global statistical information from particles' local best solutions. Alguliev *et al.* [35] proposed a discrete PSO based on EDA that models the distribution of promising solutions for document summarization.

Besides, the hybridization of PSO and EDA has been performed to solve some continuous optimization problems. A significant work is EDA–PSO [14] where the population is split into chunks, and the update of particles within a chunk follows PSO rules, while the global update is based on EDA. EDA is used to estimate the current selected promising solutions and some new offspring are from both EDA and PSO. Reference [15] presents PSEDA in which the position update process is simulated as a mixture of Gaussian distribution, which is in turn used to generate new offspring. Reference [36] presents PSO_Bounds that uses EDA to manipulate the allowable bounds for PSO particles, in which the location of current promising particles is estimated by EDA while new offspring are generated in an estimated appropriate interval. Reference [37] presents a hybrid EDA–PSO method in which EDA uses a Gaussian model to capture current promising solutions' characteristics and generate new individuals. Reference [38] presents an estimation of distribution PSO that borrows ideas from ant colony optimization. Reference [39] presents the hybrids of EDA and two PSO

variants through probabilistic modeling of best solutions in the swarm. Reference [40] presents an estimation of particle swarm distribution algorithm where EDA is used to estimate the distribution of current selected promising solutions and some portions in a new particle are generated from EDA that can provide global information and others are supplied by PSO that owns local information.

Through the above analysis, we can see clearly that the existing methods use only EDA to estimate the distribution of their current selected particles or pbests. EDA is never used to store the historical memory of particles' promising pbests that may be helpful to improve the performance of PSO. This paper is the first to use EDA to store the historical memory of particles' promising pbests in PSO and then define a new particle generation mechanism.

III. Proposed HMPSO

Original PSO may be trapped into a local optimum because all particles in the swarm may easily be attracted to the gbest region. CLPSO [6] is proposed to avoid its premature convergence to a certain extent. Any particle in it can learn from other particles' current pbests, i.e., each component of a particle's position and velocity can potentially learn from a different particle, thereby giving the particles more chances to use the beneficial information of the swarm, and eventually having a larger potential space to fly. However, both original PSO and CLPSO fail to retain and use all of particles' historical promising pbests except their current pbests. In order to do so, this paper presents HMPSO by innovatively using EDA to estimate and preserve the distribution information of particles' historical promising pbests that are selected by a tournament procedure.

The advantage of incorporating EDA into HMPSO is that the pbests' historical promising information of particles in the swarm can be estimated and preserved. Historical memory is formed, and it is beneficial because it is able to reflect the distribution of particles' historical promising pbests in the solution space. Thus, we can see that the main role of EDA is that it can build a distribution model of this historical memory and extract meaningful information from it to guide the search of each particle. In this way, each particle has more comprehensive learning and search ability with the help of this historical memory information than CLPSO.

HMPSO carries the following information: historical memory, particles' current pbests, and the swarm's gbest. We define a new particle generation mechanism to use them.

A. Historical Memory H

Historical memory H is decided as follows.

1) Generate randomly a swarm of Q particles from a search space and calculate their fitness values, and then obtain all particles' pbests and swarm's gbest.
2) Select N promising pbests from the swarm according to their fitness values by using a selection procedure, e.g., a tournament procedure.
3) Adopt EDA to estimate the distribution of good regions in the search space based on the selected promising pbests. Population-based incremental learning (PBIL) [41] is used to model the distribution of the selected promising pbests and construct historical memory. So there is a distribution vector $\mathbf{P} = (p^1, p^2, \ldots, p^d, \ldots, p^D),\ 1 \le d \le D$, in which p^d is used to characterize the dth component's distribution of the selected promising pbests in the search space, and it is learned and updated from the distribution of historical selected promising pbests' memory and current selected promising pbests. Here, the distribution vector $\mathbf{P}$ is considered as the historical memory for every component. New offspring are generated by sampling the updated distribution model. Let $\widetilde{p}^d$ be a distribution of the dth component by estimating current selected promising pbests. In HMPSO, we assume that the distribution of each component follows a normal distribution, thereby demanding EDA to estimate and preserve two parameters only, i.e., mean and standard deviation, expressed as $\widetilde{p}_m^d$ and $\widetilde{p}_s^d$, respectively. That is, $\widetilde{p}^d = \{\widetilde{p}_m^d, \widetilde{p}_s^d\}$ and $p^d = \{p_m^d, p_s^d\}$. The initial value $\widetilde{p}^d(1)$ is calculated in (7) and (8) when $t = 0$ and is treated as the initial value of p^d. The updating formulas of p_m^d, p_s^d, $\widetilde{p}_m^d$, and $\widetilde{p}_s^d$ at time $t+1$ are as follows:

$$p_m^d(t+1) = (1 - m_\lambda) \times p_m^d(t) + m_\lambda \times \widetilde{p}_m^d(t+1) \quad (5)$$

$$p_s^d(t+1) = (1 - \mathrm{Var}_\lambda) \times p_s^d(t) + \mathrm{Var}_\lambda \times \widetilde{p}_s^d(t+1) \quad (6)$$

$$\widetilde{p}_m^d(t+1) = \frac{\sum_{i=1}^{N} \mathrm{pbest}_i^d}{N} \quad (7)$$

$$\widetilde{p}_s^d(t+1) = \sqrt{\frac{\sum_{i=1}^{N} \left(\mathrm{pbest}_i^d - \widetilde{p}_m^d(t+1)\right)^2}{N}} \quad (8)$$

where p_m^d and p_s^d can be regarded as the distribution of mean and standard deviation for modeling the particles' historical promising pbests. m_λ and $\mathrm{Var}_\lambda \in [0, 1]$ are the learning parameters of mean and standard deviation, respectively. They are used to balance the contributions between historical memory and the information extracted from the current particles' pbests. The bigger m_λ and Var_λ, the greater contribution of current particles' pbests; while the smaller m_λ and Var_λ, the greater contribution of historical memory. Thus, the setting of learning parameters m_λ and Var_λ has a direct impact on exploration and exploitation abilities. For example, if m_λ and Var_λ are 0, there is no exploitation and the offspring solutions are sampled based on cumulative historical memory completely. As m_λ and Var_λ increase, the exploitation ability increases, while the exploration ability to search the portions of historical memory in a problem space diminishes.

B. Information Pool Used in HMPSO

In HMPSO, historical memory H, particles' current pbests, and the swarm's gbest constitute an information pool $I = (H_i^d, \mathrm{Cpbest}_i^d, \mathrm{gbest}_i^d)^T$, to be explained next.

1) H_i^d: It is obtained from particles' historical promising pbests via a tournament selection procedure. When it is selected as the next value of the ith particle's

*d*th component, we perform the learning based on historical memory.

2) Cpbest$_i^d$*:* It is obtained from other particle's pbests. When it is selected as the *i*th particle's *d*th component, we perform comprehensive learning [6]. The way that selects it is as same as CLPSO.
   a) Select two particles from current population randomly excluding particle *i*.
   b) Compare the fitness values of two selected particles' pbests through a tournament selection procedure and choose the better one.
   c) Use the winner's pbest to update the velocity for that component.
3) gbest$_i^d$*:* It is the swarm's global gbest. When it is selected as the next value of the *i*th particle's *d*th component, we perform in-depth learning [1].

Note that pbests and gbest are obtained according to the general procedure in PSO.

C. Velocity Updating and Position Generating Strategy

A new velocity updating equation is designed and used to generate new one

$$V_{i\_j}^d(t+1) = \omega \times V_i^d(t) + c \times r_{i\_j}^d \times \left(Y_{i\_j}^d(t) - X_i^d(t)\right) \quad (9)$$

where $Y_{i\_j}^d$ denotes the information that is used in the *d*th component for the *j*th candidate position in the *i*th particle, $V_{i\_j}^d$ is its velocity, $r_{i\_j}^d$ is a random number in the range [0, 1]. Then a new candidate's position is generated

$$\mathrm{Pos}_{i\_j}^d(t+1) = X_i^d(t) + V_{i\_j}^d(t+1) \quad (10)$$

where $\mathbf{Pos}_{i\_j} = (\mathrm{Pos}_{i\_j}^1, \mathrm{Pos}_{i\_j}^2, \ldots, \mathrm{Pos}_{i\_j}^D)$ is the *j*th candidate position of the *i*th particle. $j \in \{1, 2, 3\}$ is the index of three elements in information pool I, respectively. Here, we assume that the optimization function is to be minimized. The index of the best candidate position is calculated as

$$m_i = \arg\min_{j\in\{1,2,3\}} F\left(\mathbf{Pos}_{i\_j}(t+1)\right) \quad (11)$$

$$Y_{i\_m_i}^d(t+1) = e_{m_i}(t+1) \times \left(H_i^d, \mathrm{Cpbest}_i^d, \mathrm{gbest}_i^d\right)^T \quad (12)$$

$$e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (13)$$

where $m_i \in \{1, 2, 3\}$ defines the best candidate position index from a fitness function $F(\mathbf{Pos}_{i\_j}(t+1))$ and it is used to update all components of the *i*th particle. $Y_{i\_m_i}^d$ represents the information as calculated by the product of e_{m_i} and information pool I. e is a 3×3 unit matrix and $e_{m_i}(t+1)$ is used to indicate which row vector is at iteration $t+1$. $Y_{i\_m_i}^d(t+1)$ determines the best velocity $V_{i\_m_i}^d(t+1)$ in (9) and the best candidate position $\mathrm{Pos}_{i\_m_i}^d(t+1)$ in (10).

The final position is generated from the best candidate position

$$X_i^d(t+1) = \mathrm{Pos}_{i\_m_i}^d(t+1) \quad (14)$$

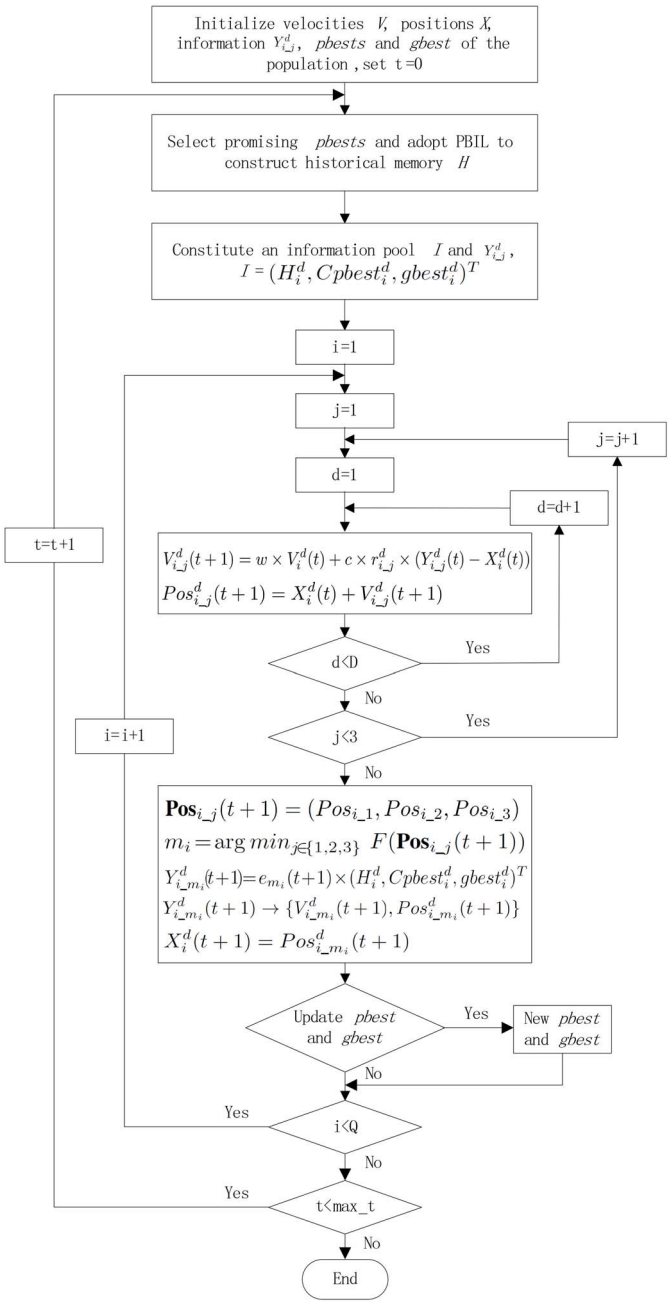


Fig. 1. Flowchart of HMPSO. *t*: time counter from 1 to max_*t*.

D. Complete Procedure of HMPSO

The flowchart of HMPSO is given in Fig. 1. The details of the proposed HMPSO process are presented in Algorithm 1. At each iteration, historical memory, particles' current pbests, and the swarm's gbest in the information pool are used to create three new candidate positions. Then the best one enters the next iteration.

HMPSO differs from other PSO algorithms in the following two aspects.

1) It uses EDA to form historical memory, while other PSO algorithms fail to do so. EDA can estimate and preserve the distribution of some particles' historical promising pbests. It is beneficial because it is able to reflect the promising regions in a search space.

Algorithm 1 HMPSO

begin
  do
    At iteration $t \geq 1$, calculate the fitness value for each particle in the swarm and update their own pbests and the swarm's gbest;
    Select two particles from the current population randomly excluding particle i whose velocity needs update;
    Compare the fitness values of two selected particles' pbests through a tournament selection procedure and select the better one;
    The winner's pbest is preserved to construct the set of promising pbests for a component;
    Through PBIL, estimate the distribution of promising regions in the search space based on (5)–(8) and generate the historical memory by the estimated distribution;
    For every particle, generate three new candidate positions Pos_{i_1}, Pos_{i_2}, and Pos_{i_3} via the information pool I;
    Evaluate the fitness function values of the three new candidate positions and select the best one as the current particle X_i;
  until The end condition is satisfied.
end

2) It defines a new particle generation mechanism. The idea behind it is to generate a new particle through the competition among those generated based on historical memory, particles' current pbests, and the swarm's gbest.

IV. Experimental Evaluation

Extensive simulations are carried out in order to compare HMPSO with CPSO-H6 [3], CLPSO [6], ALCPSO [7], FIPS [4], HPSO–TVAC [5], EDA–PSO [14], and PSEDA [15]. Further experimental evaluations with two HMPSO variants are then carried out to analyze and compare the performance of proposed HMPSO when it uses two different ways to build historical memory. All the mentioned algorithms are coded in MATLAB-R2010a and simulations are executed on a 2.4 GHz Xeon E5-2665 processor with 32 GB main memory running under Windows server 2008 environment.

A. Benchmark Functions and Algorithm Configuration

In order to study their performance deeply, 28 benchmark functions in CEC2013 [42] are used for the experimental tests here, which have been widely used for real-parameter optimization, as shown in Table I. The dimension D is set to 50. A relatively universal evaluation method is used to test the average and standard deviation of the function value error, which is defined as $f(\overrightarrow{x}) - f(\overrightarrow{x^*})$, where $\overrightarrow{x}$ is the best solution found by the algorithm in a test instance and $\overrightarrow{x^*}$ is the global optimum of the benchmark function. All results of the algorithms are obtained from 51 independent runs. In addition, we assume that the termination criterion of all algorithms is a fixed number of function evaluations, which is set to be 500 000. The population size in all algorithms is 40.

TABLE I
28 CEC2013 Benchmark Functions, Search Range: $[-100, 100]^D$

	No.	Functions (n: number of basic functions)	$f(\overrightarrow{x^*})$
Unimodal Functions	F1	Sphere Function	-1400
	F2	Rotated High Conditioned Elliptic Function	-1300
	F3	Rotated Bent Cigar Function	-1200
	F4	Rotated Discus Function	-1100
	F5	Different Powers Function	-1000
Basic Multimodal Functions	F6	Rotated Rosenbrock's Function	-900
	F7	Rotated Schaffers F7 Function	-800
	F8	Rotated Ackley's Function	-700
	F9	Rotated Weierstrass Function	-600
	F10	Rotated Griewank's Function	-500
	F11	Rastrigin's Function	-400
	F12	Rotated Rastrigin's Function	-300
	F13	Non-Continuous Rotated Rastrigin's Function	-200
	F14	Schwefel's Function	-100
	F15	Rotated Schwefel's Function	100
	F16	Rotated Katsuura Function	200
	F17	Lunacek Bi_Rastrigin Function	300
	F18	Rotated Lunacek Bi_Rastrigin Function	400
	F19	Rotated Expanded Griewank's plus Rosenbrock's Function	500
	F20	Rotated Expanded Scaffer's F6 Function	600
Composition Functions	F21	Composition Function 1 (n=5,Rotated)	700
	F22	Composition Function 2 (n=3,Unrotated)	800
	F23	Composition Function 3 (n=3,Rotated)	900
	F24	Composition Function 4 (n=3,Rotated)	1000
	F25	Composition Function 5 (n=3,Rotated)	1100
	F26	Composition Function 6 (n=5,Rotated)	1200
	F27	Composition Function 7 (n=5,Rotated)	1300
	F28	Composition Function 8 (n=5,Rotated)	1400

TABLE II
PSO Algorithms Used in the Comparison

Algorithms	Year	Parameters Settings	References
CPSO-H6	2004	ω:0.9-0.4, c_1=c_2=1.49	[3]
CLPSO	2006	ω:0.9-0.4, c=1.49445, m=7	[6]
ALCPSO	2013	ω:0.4, c_1=c_2=2.0, Θ_0=60, T=2	[7]
FIPS	2004	χ=0.729, Σc_i=4.1	[4]
HPSO-TVAC	2004	ω:0.9-0.4, c_1=2.5-0.5, c_2=0.5-2.5	[5]
EDA-PSO	2010	ω:0.7, c_1=0.1, c_2=0.2, N=50	[14]
PSEDA	2011	w_1=0.09313, w_2=0.38187, w_3=0.38187, w_4=0.05, w_5=0.09313	[15]

The parameter configurations for the seven existing PSO algorithms that compare with HMPSO are given in Table II, according to their corresponding references.

In HMPSO, the value of ω is decreased linearly from 0.9 to 0.4 for all benchmark functions [3], [5], [6], $N = 40$ and $c = 1.49445$. Besides, it has its own unique parameters. m_λ and Var_λ are the learning parameters of mean and standard deviation, respectively, in EDA. Their ranges are $[0, 1]$. "0" means that all new offspring will be sampled based on historical memory, while "1" means that all new offspring will be sampled based on the distribution of current particles' pbests. In this paper, we set the values of m_λ and Var_λ to be same. The first reason for doing so is to try to simplify the parameter settings since there exists too many of their combinations. The second reason is that the performance of HMPSO is relatively better when m_λ and Var_λ are set to be equal through a great number of tests. Next, in our preliminary experiments, we focus only on finding the most suitable parameters for them. The results in Fig. 2 illustrate the influences of different m_λ and Var_λ on some functions F2, F7, F8, F14, F22, and F27. Their increment is 0.1. Each plot in Fig. 2 contains multiple boxplots. Each boxplot corresponding to a particular value of m_λ and Var_λ uses the minimum, first quartile, median, third quartile, and maximum values of a set of experimental data to describe statistics intuitively. Its function is primarily to identify the exceptional value, to estimate the skewness and tail weight of data distribution, to compare the distribution shape characteristics and the dispersion degree

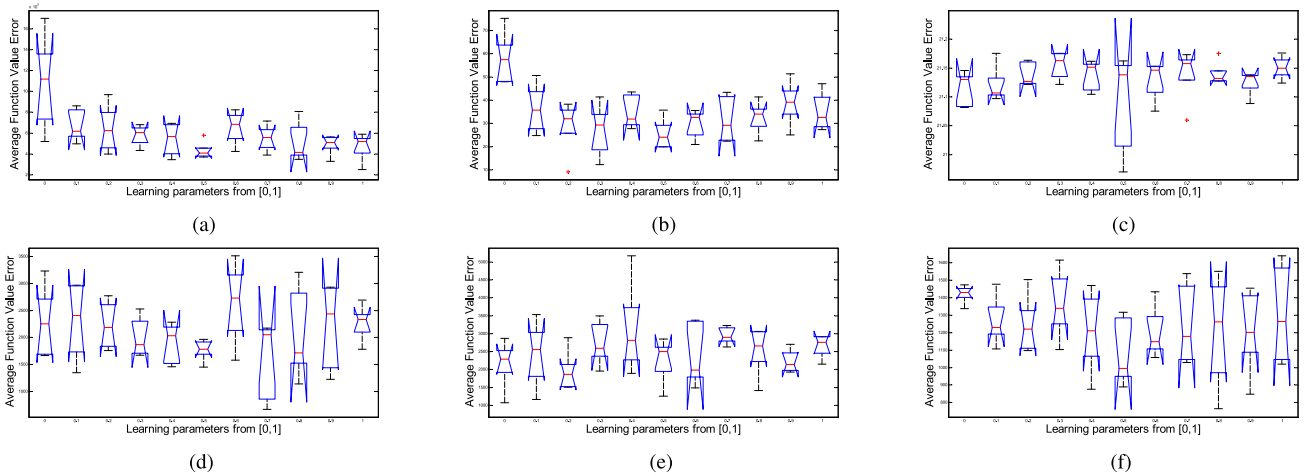


Fig. 2. Influences of different m_λ and Var_λ on HMPSO performance, which range from 0 to 1 on functions (a) F2, (b) F7, (c) F8, (d) F14, (e) F22, and (f) F27.

TABLE III
EXPERIMENTAL RESULTS OF CPSO-H6, CLPSO, ALCPSO, FIPS, HPSO–TVAC, EDA–PSO, PSEDA, AND HMPSO OVER 51 INDEPENDENT RUNS ON 28 FUNCTIONS. "MEAN E" AND "STD D" INDICATE THE MEAN AND STANDARD DEVIATION OF THE FUNCTION VALUE ERRORS. "<," ">," AND "≐" DENOTE THAT THE PERFORMANCE OF THE CORRESPONDING ALGORITHM IS WORSE THAN, BETTER THAN, AND SIMILAR TO THAT OF HMPSO

	CPSO-H6 *Mean* $E\pm$*Std* D	CLPSO *Mean* $E\pm$*Std* D	ALCPSO *Mean* $E\pm$*Std* D	FIPS *Mean* $E\pm$*Std* D	HPSO-TVAC *Mean* $E\pm$*Std* D	EDA-PSO *Mean* $E\pm$*Std* D	PSEDA *Mean* $E\pm$*Std* D	HMPSO *Mean* $E\pm$*Std* D
$F1$	2.54E-09±5.84E-08 <	2.29E-13±0.00E+00 <	7.16E-12±2.98E-11 <	3.41E-13±1.24E-13 <	8.49E-13±1.59E-13 <	1.88E+04±2.62E+03 <	5.51E+04±5.61E+02 <	**2.19E-13±3.16E-14**
$F2$	4.93E+06±3.43E+06 <	3.15E+07±7.11E+06 <	3.56E+07±2.24E+07 <	1.69E+07±1.31E+07 <	3.92E+06±3.51E+06 <	3.71E+08±5.17E+07 <	2.25E+09±1.35E+08 <	**6.67E+05±1.96E+05**
$F3$	2.01E+10±1.67E+10 <	1.88E+09±7.41E+08 <	1.91E+09±2.51E+09 <	8.43E+07±6.32E+07 <	9.92E+08±1.02E+09 <	1.10E+11±3.36E+10 <	7.31E+12±2.09E+12 <	**5.38E+07±7.83E+06**
$F4$	2.15E+05±3.61E+04 <	2.58E+04±4.48E+03 <	5.74E+03±1.78E+03 <	3.51E+03±1.13E+03 <	3.28E+03±1.06E+03 <	6.23E+04±7.80E+03 <	6.91E+04±9.07E+02 <	**4.47E+02±1.26E+02**
$F5$	1.68E-05±1.37E-05 <	2.49E-13±4.38E-14 <	1.95E-12±5.51E-13 <	3.05E-13±7.43E-14 <	9.15E-12±2.35E-11 <	5.01E+03±7.34E+02 <	1.43E+04±1.05E+03 <	**1.72E-13±5.50E-14**
$F6$	5.07E+01±2.09E+01 <	4.62E+01±7.51E-01 <	6.78E+01±3.99E+01 <	4.94E+01±1.69E+01 <	6.35E+01±2.68E+01 <	1.19E+03±2.13E+02 <	5.91E+03±1.89E+02 <	**4.28E+01±1.25E+00**
$F7$	4.63E+02±6.15E+02 <	9.91E+01±1.11E+01 <	1.22E+02±2.61E+01 <	4.98E+01±1.25E+01 <	1.39E+02±2.86E+01 <	2.35E+02±3.14E+01 <	2.31E+03±3.03E+02 <	**2.84E+01±7.57E+00**
$F8$	2.11E+01±4.59E-02 ≐	2.12E+01±3.72E-02 ≐	2.11E+01±4.41E-02 ≐	2.11E+01±4.17E-02 ≐	2.12E+01±3.53E-02 ≐	2.11E+01±3.69E-02 ≐	2.12E+01±3.68E-02 ≐	**2.11E+01±2.78E-02**
$F9$	6.99E+01±4.57E+00 <	5.17E+01±2.71E+00 <	5.46E+01±5.42E+00 <	5.99E+01±5.01E+00 <	5.78E+01±3.71E+00 <	6.98E+01±1.79E+00 <	7.09E+01±5.48E+00 <	**3.93E+01±6.75E+00**
$F10$	1.61E+01±2.65E+01 <	6.39E+00±1.64E+00 <	9.59E-01±7.93E-01 <	1.97E-01±9.42E-02 <	4.57E-01±2.48E-01 <	3.34E+03±4.07E+02 <	8.99E+03±1.65E+02 <	**9.78E-02±4.77E-02**
$F11$	**2.29E-09±2.97E-09** >	1.81E-07±8.21E-07 >	3.11E+01±6.27E+00 >	1.02E+02±1.85E+01 <	1.49E+01±7.43E+00 >	7.05E+02±4.30E+01 <	9.80E+02±2.79E+01 <	7.47E+01±1.25E+01
$F12$	9.15E+02±1.89E+02 <	2.73E+02±3.23E+01 <	2.69E+02±6.31E+01 <	1.25E+02±4.08E+01 <	5.81E+02±9.76E+01 <	6.71E+02±4.94E+01 <	1.11E+03±3.66E+01 <	**9.80E+01±3.37E+01**
$F13$	9.41E+02±1.47E+02 <	3.65E+02±2.85E+01 <	4.20E+02±7.22E+01 <	2.74E+02±5.02E+01 <	7.46E+02±8.69E+01 <	7.07E+02±4.30E+01 <	1.09E+03±3.88E+01 <	**2.28E+02±7.54E+01**
$F14$	**1.37E+01±2.97E+00** >	4.88E+01±7.31E+00 >	1.06E+03±3.32E+02 >	5.11E+03±1.42E+03 <	1.25E+03±2.99E+02 >	1.39E+04±3.86E+02 <	1.34E+04±6.67E+02 <	1.74E+03±5.11E+02
$F15$	9.44E+03±1.11E+03 <	8.75E+03±4.88E+02 <	8.63E+03±2.06E+03 <	1.31E+04±7.72E+02 <	8.98E+03±1.09E+03 <	1.43E+04±3.69E+02 <	1.39E+04±7.48E+02 <	**6.85E+03±8.91E+02**
$F16$	2.49E+00±6.66E-01 >	**2.45E+00±3.40E-01** >	3.03E+00±3.72E-01 >	3.21E+00±3.32E-01 <	2.53E+00±4.76E-01 >	3.36E+00±3.04E-01 <	3.32E+00±2.61E-01 <	3.10E+00±3.74E-01
$F17$	**5.09E+01±1.17E-03** >	8.58E+01±4.45E+00 >	1.18E+02±1.84E+01 >	1.61E+02±2.72E+01 <	8.26E+01±1.73E+01 >	1.22E+03±6.58E+01 <	1.11E+03±2.91E+01 <	1.57E+02±9.97E+01
$F18$	7.28E+02±1.57E+02 <	4.38E+02±2.31E+01 >	3.98E+02±8.93E+01 >	**3.45E+02±3.52E+01** >	7.38E+02±1.15E+02 <	1.22E+03±6.69E+01 <	1.10E+03±2.85E+01 <	4.68E+02±2.10E+01
$F19$	9.42E+00±4.23E-01 <	5.93E+00±7.79E-01 <	1.43E+01±5.38E+00 <	1.06E+01±3.01E+00 <	9.05E+00±2.15E+00 <	9.26E+03±3.31E+03 <	3.33E+05±1.05E+04 <	**5.76E+00±9.48E-01**
$F20$	2.49E+01±3.31E-01 <	2.31E+01±4.42E-01 <	2.45E+01±8.97E-01 <	2.35E+01±1.44E+00 <	2.41E+01±7.55E-01 <	2.42E+01±7.04E-01 <	2.49E+01±1.32E-01 <	**1.90E+01±1.02E+00**
$F21$	7.91E+02±4.15E+02 >	**3.82E+02±1.73E+02** >	8.76E+02±3.57E+02 >	7.41E+02±4.31E+02 >	9.46E+02±2.08E+02 <	3.21E+03±4.65E+02 <	3.87E+03±1.08E+01 <	9.27E+02±2.46E+02
$F22$	**8.26E+01±9.46E+01** >	1.28E+02±6.72E+01 >	2.64E+03±5.95E+02 <	5.05E+03±8.86E+02 <	1.43E+03±3.95E+02 >	1.45E+04±3.65E+02 <	1.63E+04±5.73E+02 <	2.26E+03±6.59E+02
$F23$	1.22E+04±1.42E+03 <	1.07E+04±6.01E+02 <	9.51E+03±1.74E+03 <	1.31E+04±7.98E+02 <	1.07E+04±1.33E+03 <	1.47E+04±5.65E+02 <	1.61E+04±5.01E+02 <	**7.18E+03±9.26E+02**
$F24$	3.94E+02±1.37E+01 <	3.41E+02±6.89E+00 <	3.53E+02±1.65E+01 <	2.96E+02±1.80E+01 <	3.94E+02±1.88E+01 <	3.79E+02±4.62E+00 <	1.17E+03±1.96E+02 <	**2.64E+02±1.65E+01**
$F25$	4.05E+02±1.76E+01 <	3.79E+02±8.02E+00 <	3.91E+02±1.10E+01 <	3.51E+02±1.76E+01 <	4.54E+02±1.87E+01 <	3.78E+02±4.53E+00 <	7.01E+02±4.16E+01 <	**3.45E+02±1.53E+01**
$F26$	4.67E+02±6.79E+01 <	**2.05E+02±7.67E-01** >	4.19E+02±6.46E+01 <	3.89E+02±3.29E+01 <	3.91E+02±1.24E+02 <	4.52E+02±7.11E+01 <	5.81E+02±3.63E+01 <	3.57E+02±5.75E+01
$F27$	2.24E+03±1.03E+02 <	1.63E+03±3.61E+02 <	1.76E+03±1.46E+02 <	1.25E+03±1.05E+02 <	2.11E+03±1.46E+02 <	2.08E+03±5.27E+01 <	3.45E+03±2.95E+02 <	**1.09E+03±1.84E+02**
$F28$	2.48E+03±2.27E+03 <	**4.02E+02±6.15E-04** >	1.59E+03±1.68E+03 <	1.27E+03±1.19E+03 <	3.39E+03±2.54E+03 <	3.09E+03±9.96E+02 <	9.81E+03±4.69E+02 <	5.09E+02±2.99E+02
	21	18	21	25	22	27	27	Worse than HMPSO
	6	9	6	2	5	0	0	Better than HMPSO
	1	1	1	1	1	1	1	Similar to HMPSO

of different grouped experimental data, etc. It seems that the setting of $m_\lambda = \mathrm{Var}_\lambda = 0.5$ is more appropriate and robust than other parameters. Hence, 0.5 is adopted for all further experiments in this paper.

B. Comparison With Seven PSO Algorithms

The experimental results are given in Table III, which show the mean and standard deviation of 28 benchmark functions. Each is obtained from 51 independent runs. The last three rows of Table III summarize the experimental results. The best results obtained by eight algorithms are shown in bold.

1) Functions F1–F5 are unimodal functions. Clearly, HMPSO is overall the best algorithm among these methods. It outperforms other seven algorithms on all five benchmark functions. The outstanding performance of HMPSO should be due to the use of the swarm's global gbest and particles' historical promising pbests that provide a promising search area, which leads to a very fast and precise convergence.
2) F6–F20 are basic multimodal functions. HMPSO is significantly better than the others on nine rotated functions, i.e., F6, F7, F9, F10, F12, F13, F15, F19, and F20. All algorithms have same performance on rotated Ackley's function F8, which is a nonseparable and asymmetrical function where the plains are vast and its global optimum is difficult to locate. Specially, HMPSO outperforms EDA–PSO and PSEDA on fourteen

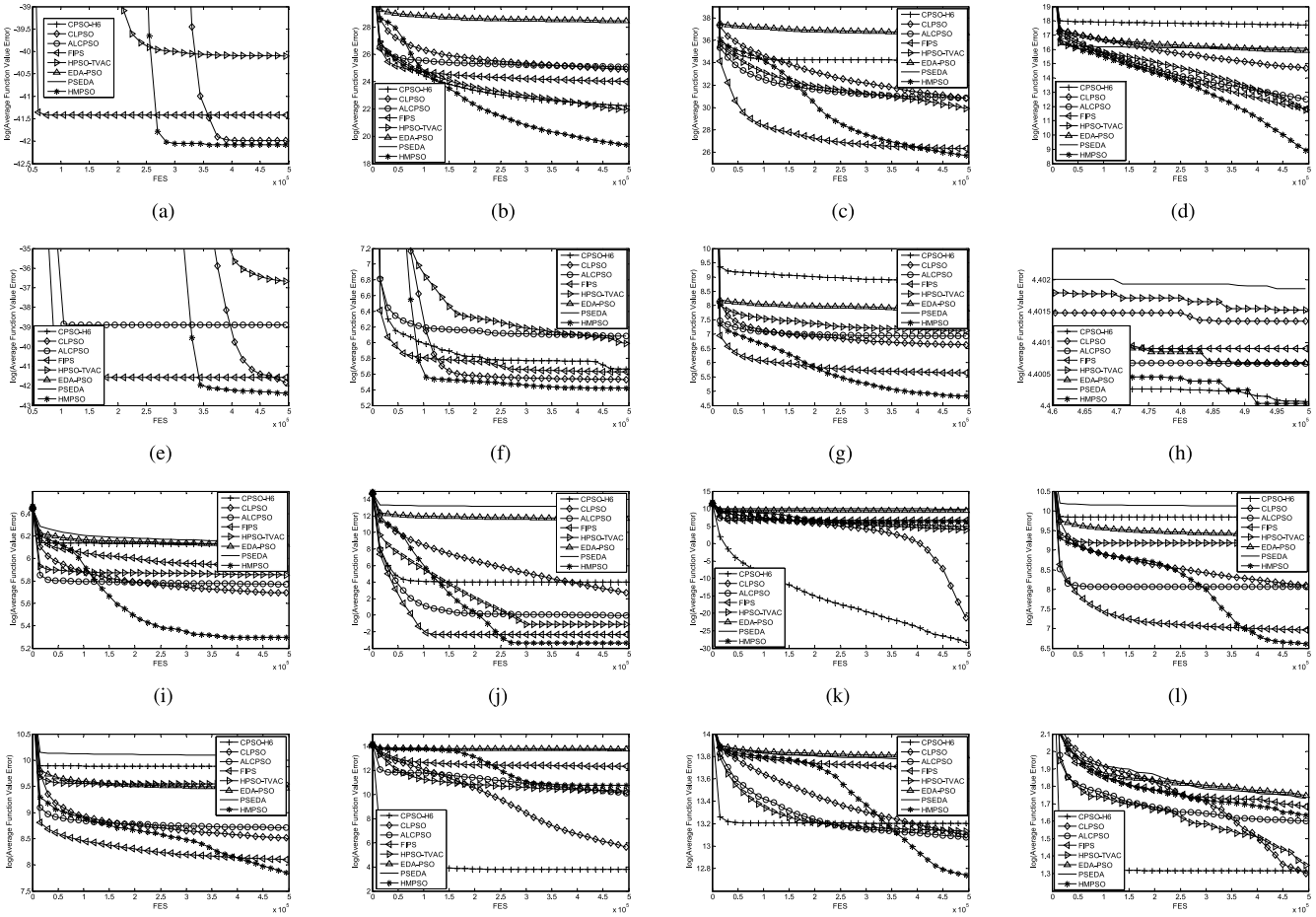


Fig. 3. Evolution of the base 2 logarithm of the mean of function value errors derived from all algorithms over 51 independent runs on functions (a)–(p) F1–F16.

functions except F8. It achieves the best results on most rotated multimodal functions. This implies that it is more effective in solving functions with more linkage due to its historical memory of particles' promising pbests that are selected by a tournament procedure, which pays attention to the relationship among different components to some extent by the competition of the fitness values that are calculated by considering all components of different particles' pbests. CPSO-H6, CLPSO, ALCPSO, and HPSO–TVAC outperform HMPSO on F11, F14, and F17. CPSO-H6 is the best. Rastrigin's function F11 is a multimodal, asymmetrical function and owns a huge number of local optima. Schwefel's function F14 is a multimodal function in which local optima count is huge. Lunacek bi-Rastrigin function F17 is a nonseparable multimodal. CPSO-H6 performs well on the three unrotated functions since it can utilize its cooperative mechanism by decomposing a larger search space into several smaller ones, which ensures that the search space is sampled more thoroughly when the number of local optima is huge. On Rotated Katsuura function F16, CPSO-H6, CLPSO, ALCPSO, and HPSO–TVAC outperform HMPSO. CLPSO is the winner since it can successfully avoid falling into the deep local optimum which is far from the global optimum. FIPS, CLPSO, and ALCPSO outperform HMPSO on Rotated Lunacek bi-Rastrigin function F18 that is a nonseparable multimodal function, and FIPS is the winner by using the information of the entire neighborhood to guide the particles. To sum up, HMPSO is ranked first in all algorithms since it wins nine rotated functions.

3) Finally, F21–F28 are composition functions. Composition functions are constructed based on some basic benchmark functions to obtain more challenging problems with a randomly located global optimum and several randomly located deep local optima. They are much harder than the others. Overall, the performance of HMPSO is better than its seven competitors. It outperforms them on four rotated functions F23–F25 and F27. HMPSO outperforms CPSO-H6 on six functions except F21 and F22. CLPSO outperforms HMPSO on F21, F22, F26, and F28 since the former is good at complex multimodal problems irrespective of whether they are unrotated or rotated. ALCPSO and FIPS are better than HMPSO on F21 only. HPSO–TVAC is better than HMPSO on F22 only. Besides, HMPSO outperforms EDA–PSO and PSEDA on all the eight functions.

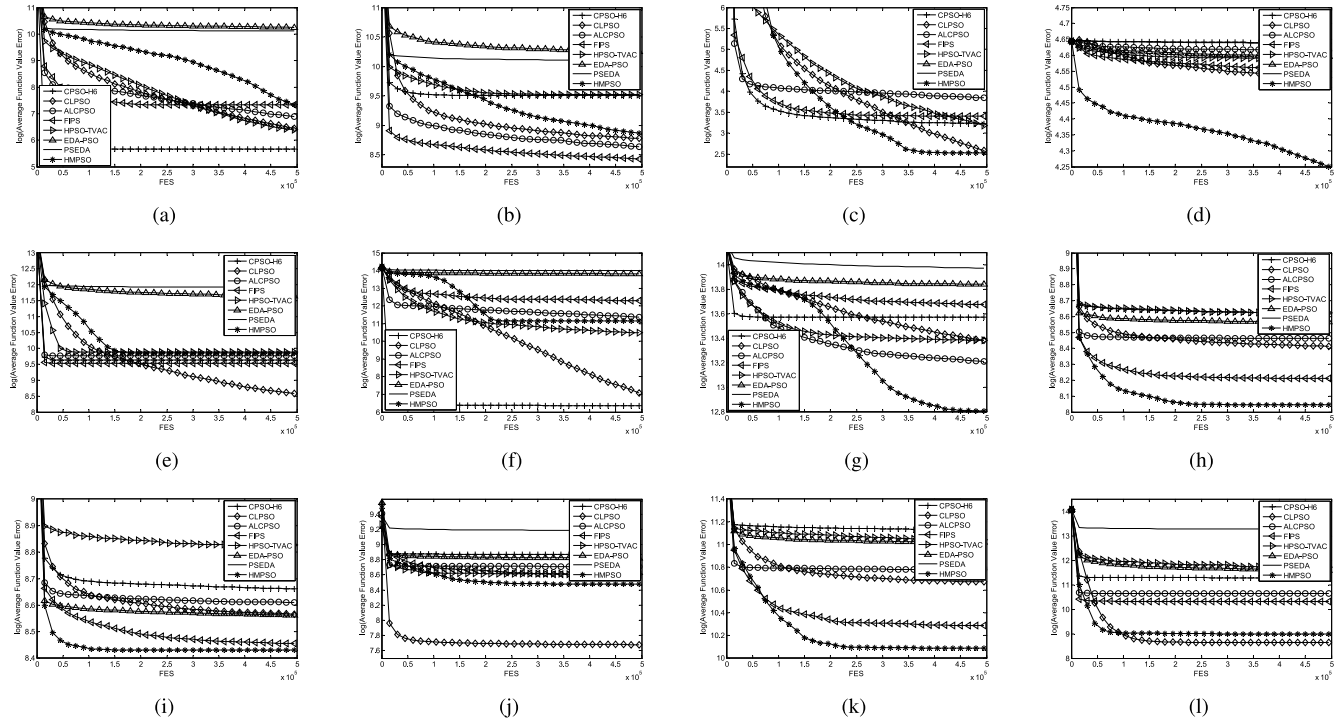


Fig. 4. Evolution of the base 2 logarithm of the mean of function value errors derived from all algorithms over 51 independent runs on functions (a)–(l) F17–F28.

The evolution of the base 2 logarithm of the mean of function value errors derived from the eight algorithms over 51 independent runs on 28 benchmark functions are shown in Figs. 3 and 4. Comparing their results and convergence, it is easy to observe the performance advantage of HMPSO over its competitors. The last three rows in Table III indicate that the numbers of functions for which HMPSO are better than CPSO-H6, CLPSO, ALCPSO, FIPS, HPSO–TVAC, EDA–PSO, and PSEDA are 21, 18, 21, 25, 22, 27, and 27, respectively. The numbers of functions for which they are similar to 1. Among these eight PSO algorithms, CPSO-H6 presents the best performance on some unrotated multimodal functions. However, its performance is seriously affected after rotation. CLPSO presents the best performance on some multimodal and composition functions irrespective of whether they are unrotated or rotated. However, it does not perform well when the functions are unimodal or simple multimodal. ALCPSO and FIPS present excellent performance on some rotated multimodal functions. However, their performance on unimodal functions is similar to CLPSO, and they do not perform well on composition functions. HPSO–TVAC presents excellent performance on some unimodal and unrotated multimodal functions. However, its performance is not good on rotated multimodal functions and composition ones. EDA–PSO and PSEDA present excellent performance on some rotated multimodal functions. However, they are inferior to other PSO algorithms. HMPSO achieves the best results on all unimodal functions, and a key reason is that it uses the information of swarm's global gbest and particles' historical promising pbests that provide a promising search area from its information pool. Moreover, HMPSO presents excellent performance on most rotated multimodal and composition functions. This implies that the HMPSO is effective in solving functions with more linkage. This property is due to the historical memory in HMPSO that preserves the correlation information among the components to a certain extent. In addition, facing composition functions, HMPSO is more competitive than CLPSO.

C. Comparison With HMPSO Variants

In this section, two HMPSO variants with novel mechanisms to build historical memory are proposed. In HMPSO, the historical memory is represented as a single global distribution model that is built by those selected promising particles' pbests. It can thus be viewed as the memory of overall population. In the first variant, we utilize the historical pbests of a particle to build a local distribution model for that particle, and this local distribution model is represented as the historical memory of a particle. Each particle in population generates an offspring through its local historical memory. This variant is called HMPSO-L.

Next, in HMPSO, we assume that each component follows a normal distribution. A single global distribution model is built without considering the relationship among different components deeply. In the second variant, we utilize a multivariate Gaussian distribution to represent different components' relationship and assume the whole population to follow a normal distribution. Then all components of a particle can be generated simultaneously. This variant is called HMPSO-M. Its historical memory can also be viewed as the memory of overall population.

The experimental results over 51 independent runs on 28 benchmark functions among HMPSO-L, HMPSO-M,

TABLE IV
EXPERIMENTAL RESULTS OF HMPSO-L, HMPSO-M, AND HMPSO OVER 51 INDEPENDENT RUNS ON 28 BENCHMARK FUNCTIONS

	HMPSO-L $Mean\ E \pm Std\ D$	HMPSO-M $Mean\ E \pm Std\ D$	HMPSO $Mean\ E \pm Std\ D$
$F1$	7.69E-13±5.27E-13 <	6.96E-13±4.94E-13 <	**2.19E-13±3.16E-14**
$F2$	1.15E+06±7.31E+05 <	3.47E+06±2.22E+06 <	**6.67E+05±1.96E+05**
$F3$	1.14E+08±1.70E+07 <	8.71E+07±6.07E+07 <	**5.38E+07±7.83E+06**
$F4$	5.44E+02±3.82E+02 <	2.22E+03±1.46E+03 <	**4.47E+02±1.26E+02**
$F5$	1.38E-12±7.52E-13 <	1.71E-12±1.46E-12 <	**1.72E-13±5.50E-14**
$F6$	4.63E+01±1.12E+01 <	4.45E+01±1.36E+00 <	**4.28E+01±1.25E+00**
$F7$	6.57E+01±1.10E+01 <	5.34E+01±8.49E+00 <	**2.84E+01±7.57E+00**
$F8$	2.11E+01±3.23E-02 ≐	2.11E+01±3.39E-02 ≐	**2.11E+01±2.78E-02**
$F9$	4.77E+01±5.62E+00 <	4.87E+01±5.00E+00 <	**3.93E+01±6.75E+00**
$F10$	**9.14E-02±4.11E-02** >	9.52E-02±5.17E-02 >	9.78E-02±4.77E-02
$F11$	8.27E+01±2.50E+01 <	**7.25E+01±1.38E+01** >	7.47E+01±1.25E+01
$F12$	1.70E+02±4.07E+01 <	1.28E+02±3.96E+01 <	**9.80E+01±3.37E+01**
$F13$	3.06E+02±5.02E+01 <	2.43E+02±4.23E+01 <	**2.28E+02±7.54E+01**
$F14$	**1.62E+03±5.01E+02** >	1.69E+03±5.42E+02 >	1.74E+03±5.11E+02
$F15$	7.77E+03±1.09E+03 <	8.66E+03±2.46E+03 <	**6.85E+03±8.91E+02**
$F16$	3.21E+00±4.71E-01 <	3.37E+00±2.79E-01 <	**3.10E+00±3.74E-01**
$F17$	1.90E+02±2.11E+01 <	1.88E+02±2.72E+01 <	**1.57E+02±9.97E+01**
$F18$	**2.90E+02±8.28E+01** >	3.04E+02±9.40E+01 >	4.68E+02±2.10E+01
$F19$	1.22E+01±3.45E+00 <	1.19E+01±3.00E+00 <	**5.76E+00±9.48E-01**
$F20$	2.06E+01±1.07E+00 <	2.16E+01±7.58E-01 <	**1.90E+01±1.02E+00**
$F21$	**8.83E+02±3.04E+02** >	8.87E+02±3.25E+02 >	9.27E+02±2.46E+02
$F22$	**1.63E+03±6.50E+02** >	1.89E+03±7.03E+02 >	2.26E+03±6.59E+02
$F23$	7.92E+03±1.11E+03 <	7.73E+03±1.25E+03 <	**7.18E+03±9.26E+02**
$F24$	3.12E+02±1.37E+01 <	3.03E+02±1.48E+01 <	**2.64E+02±1.65E+01**
$F25$	3.97E+02±1.40E+01 <	3.89E+02±1.27E+01 <	**3.45E+02±1.53E+01**
$F26$	3.76E+02±8.86E+01 <	3.63E+02±8.98E+01 <	**3.57E+02±5.75E+01**
$F27$	1.53E+03±1.26E+02 <	1.41E+03±1.48E+02 <	**1.09E+03±1.84E+02**
$F28$	6.63E+02±9.10E+01 <	8.18E+02±1.15E+02 <	**5.09E+02±2.99E+02**
	22	21	Worse than HMPSO
	5	6	Better than HMPSO
	1	1	Similar to HMPSO

and HMPSO are shown in Table IV, where "Mean *E*" and "Std *D*" indicate the mean and standard deviation of the function value errors, respectively. "<," ">," and "≐" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of HMPSO, respectively. The last three rows of Table IV summarize the experimental results. The best results among the three algorithms are shown in bold.

1) Functions F1–F5 are unimodal functions. Clearly, HMPSO is overall the best. It outperforms the other two on all five benchmark functions. These results indicate that the historical memory built by assuming the distribution of each component follows a normal distribution in HMPSO outperforms the other two mechanisms on unimodal functions.
2) F6–F20 are basic multimodal functions. HMPSO is significantly better than HMPSO-L and HMPSO-M on ten rotated functions, i.e., F6, F7, F9, F12, F13, F15–F17, F19, and F20. All algorithms have same performance on F8. HMPSO achieves the best results on most rotated multimodal functions. HMPSO-L outperforms HMPSO-M and HMPSO on F10, F14, and F18. This implies that the history memory built by a local distribution model for that particle has good ability to solve these unrotated and rotated functions. HMPSO-M outperforms HMPSO on four functions F10, F11, F14, and F18. Especially, on function F11, HMPSO-M is the best. This implies that the historical memory built by a multivariate Gaussian distribution by assuming the whole population to follow a normal distribution is more competitive than HMPSO-L and HMPSO on these unrotated and rotated functions.
3) Finally, F21–F28 are composition functions. Overall, HMPSO ranks the first. It outperforms the other two on six functions F23–F28. HMPSO-L and HMPSO-M outperform HMPSO on functions F21 and F22. HMPSO-L is the best on these two functions.

The last three rows in Table IV indicate that the numbers of functions for which HMPSO is better than HMPSO-L and HMPSO-M are 22 and 21, respectively. The numbers of functions for which they are similar to 1. It is clear from the results that with the historical memory that built by assuming the distribution of each component to follow a normal distribution, HMPSO presents the best performance on all unimodal functions, and majority of rotated multimodal and composition functions. HMPSO-L presents excellent performance on some unrotated and rotated multimodal functions. HMPSO-M is more competitive than HMPSO-L and HMPSO on some unrotated and rotated functions. These results indicate that the performance of HMPSO has the potential to be improved with the help of integrating the historical memory built by a local distribution model for each particle or a multivariate Gaussian distribution.

D. Analysis and Statistics of HMPSO

In order to supply a thorough comparison among all algorithms, a *t*-test [43] is carried out. Table V presents the *t*- and *P*-value on every benchmark function of this two-tailed test with a significance level of 0.05 between HMPSO and other algorithms. It should be mentioned that when the *t*-value is negative, it means that HMPSO outperforms the corresponding algorithm in terms of both mean and standard deviation; and vice versa. A *P*-value represents the estimated probability of getting the observed or more extreme results, assuming that the null hypothesis is true. It is a measure of how much evidence we have against the null hypothesis. If the *P*-value is less than 0.05 or 0.01, one often rejects the null hypothesis and declares the result to be statistically significant. If the *P*-value is near zero, it suggests that at least one sample median is significantly different from the others. Row "General Merit" represents the number of benchmark functions that HMPSO performs significantly better than other algorithms based on the *t*- and *P*-value.

From Table V, it can be seen that 21 *t* values are negative on functions F1–F7, F9, F10, F12, F13, F15, F18–F20, and F23–F28 in CPSO-H6, 19 *t* values are negative on functions F1–F10, F12, F13, F15, F19, F20, F23–F25, and F27 in CLPSO, 21 *t* values are negative on functions F1–F7, F9, F10, F12, F13, F15, F19, F20, and F22–F28 in ALCPSO, 25 *t* values are negative on functions F1–F7, F9–F17, F19, F20, and F22–F28 in FIPS, 23 *t* values are negative on functions F1–F10, F12, F13, F15, F18–F21, and F23–F28 in HPSO–TVAC, 27 *t* values are negative on functions F1–F7 and F9–F28 in EDA–PSO, 28 *t* values are negative on

TABLE V
COMPARISONS BETWEEN HMPSO AND OTHER ALGORITHMS ON t-TESTS

Func	t-tests	CPSO-H6	CLPSO	ALCPSO	FIPS	HPSO-TVAC	EDA-PSO	PSEDA	HMPSO-L	HMPSO-M
F1	t-value	**-3.1017**	-1.0000	-1.6626	**-6.4919**	**-27.6483**	**-51.3937**	**-702.7253**	**-4.7294**	**-6.7547**
	P-value	**0.0025**	0.3197	0.0995	**3.3017E-009**	**1.2317E-048**	**1.0252E-073**	**1.6517E-186**	**7.4020E-006**	**9.5957E-010**
F2	t-value	**-8.8485**	**-30.8281**	**-11.1383**	**-8.7574**	**-6.5829**	**-50.9317**	**-119.6019**	**-4.6306**	**-6.0898**
	P-value	**3.2899E-014**	**7.1607E-053**	**3.2170E-019**	**5.1986E-014**	**2.1565E-009**	**2.4475E-073**	**9.4894E-110**	**1.0989E-005**	**2.1099E-008**
F3	t-value	**-8.5898**	**-17.4290**	**-5.2741**	-0.6661	**-6.5281**	**-23.0123**	**-25.0542**	**-2.7439**	-1.7054
	P-value	**1.2041E-013**	**4.5130E-032**	**7.7368E-007**	0.5069	**2.7881E-009**	**9.6719E-042**	**6.8274E-045**	**0.0072**	0.0912
F4	t-value	**-42.0702**	**-40.1864**	**-21.0827**	**-18.8530**	**-18.4535**	**-56.5346**	**-523.0464**	-1.4086	**-8.3368**
	P-value	**2.0973E-065**	**1.5891E-063**	**1.3959E-038**	**1.0770E-034**	**5.7111E-034**	**1.0162E-077**	**1.0939E-173**	0.1621	**4.2629E-013**
F5	t-value	**-8.8073**	**-6.4911**	**-22.6849**	**-9.2392**	**-2.7298**	**-48.8255**	**-97.3656**	**-11.5729**	**-7.1844**
	P-value	**4.0458E-014**	**3.3141E-009**	**3.2267E-041**	**4.6049E-015**	**0.0075**	**1.4213E-071**	**6.8340E-101**	**3.6669E-020**	**1.2298E-010**
F6	t-value	**-2.3309**	**-10.3875**	**-4.2640**	**-2.3013**	**-5.2291**	**-38.7196**	**-221.4781**	-1.6828	-0.7788
	P-value	**0.0218**	**1.4014E-017**	**4.5587E-005**	**0.0234**	**9.3695E-007**	**5.2401E-062**	**2.1086E-136**	0.0955	0.4380
F7	t-value	**-5.0322**	**-37.0856**	**-24.0266**	**-9.8945**	**-26.0895**	**-45.0121**	**-53.7262**	**-19.1540**	**-14.6904**
	P-value	**2.1437E-006**	**2.9593E-060**	**2.4920E-043**	**1.6863E-016**	**2.0271E-046**	**3.4042E-068**	**1.4124E-075**	**3.1041E-035**	**1.0243E-026**
F8	t-value	1.9070	-0.0937	0.9851	0.6143	-0.4042	1.0285	-0.9871	-0.4547	-0.8357
	P-value	0.0594	0.9255	0.3270	0.5404	0.6869	0.3062	0.3260	0.6503	0.4053
F9	t-value	**-25.5512**	**-10.8506**	**-11.5599**	**-16.3882**	**-15.9068**	**-29.5573**	**-24.8807**	**-6.1439**	**-6.6901**
	P-value	**1.2454E-045**	**1.3623E-018**	**3.9128E-020**	**4.3844E-030**	**3.8128E-029**	**3.1928E-051**	**1.2439E-044**	**1.6484E-008**	**1.3020E-009**
F10	t-value	**-4.3055**	**-27.5485**	**-7.6436**	**-5.9039**	**-9.8304**	**-58.7294**	**-390.4435**	1.8159	1.3223
	P-value	**3.8946E-005**	**1.6965E-048**	**1.3159E-011**	**4.8936E-008**	**2.3310E-016**	**2.5163E-079**	**5.3819E-161**	0.0724	0.1891
F11	t-value	39.9282	39.9282	21.4458	**-7.6715**	28.2074	**-99.7236**	**-208.1152**	-1.7913	1.3476
	P-value	2.9162E-063	2.9162E-063	3.4372E-039	**1.1478E-011**	2.0839E-049	**6.3963E-102**	**1.0493E-133**	0.0763	0.1808
F12	t-value	**-30.1982**	**-22.7199**	**-15.7006**	**-2.6758**	**-32.0542**	**-65.0621**	**-124.3282**	**-8.8033**	**-2.9436**
	P-value	**4.6283E-052**	**2.8354E-041**	**9.7112E-029**	**0.0087**	**2.0671E-054**	**1.1642E-083**	**2.0195E-111**	**4.1279E-014**	**0.0040**
F13	t-value	**-30.5204**	**-10.9908**	**-12.2420**	**-2.6929**	**-31.2354**	**-38.0293**	**-70.3136**	**-5.4749**	-0.0055
	P-value	**1.7751E-052**	**6.7403E-019**	**1.3293E-021**	**0.0083**	**2.1777E-053**	**2.8278E-061**	**5.8390E-087**	**3.2602E-007**	0.9956
F14	t-value	25.0008	24.5576	9.2479	**-15.4538**	7.2805	**-132.7274**	**-97.0661**	2.1907	1.6968
	P-value	8.2084E-045	3.8315E-044	4.4074E-015	**2.9958E-028**	7.7262E-011	**3.0414E-114**	**9.2702E-101**	0.0308	0.0928
F15	t-value	**-12.5072**	**-12.4159**	**-5.2740**	**-36.0609**	**-5.2277**	**-53.7738**	**-42.4524**	**-4.0088**	**-4.5278**
	P-value	**3.6056E-022**	**5.6477E-022**	**7.7407E-007**	**4.0196E-059**	**9.4239E-007**	**1.2965E-075**	**8.9011E-066**	**1.1770E-004**	**1.6485E-005**
F16	t-value	7.7327	9.9271	2.5126	-0.1315	12.5331	**-2.1441**	-1.6870	-0.1366	**-2.3546**
	P-value	8.4990E-012	1.4310E-016	0.0136	0.8956	3.1765E-022	**0.0345**	0.0947	0.8916	**0.0205**
F17	t-value	7.5678	5.2958	3.2392	-0.5209	6.0775	**-58.6300**	**-58.6265**	-1.6130	-1.3346
	P-value	1.9081E-011	7.0522E-007	0.0016	0.6036	2.2318E-008	**2.9669E-079**	**2.9842E-079**	0.1099	0.1850
F18	t-value	**-11.3644**	9.3625	6.3552	23.1053	**-15.8263**	**-74.6936**	**-123.0853**	15.6045	13.2096
	P-value	**1.0382E-019**	2.4737E-015	6.2357E-009	6.8844E-042	**5.4878E-029**	**1.5507E-089**	**5.4798E-111**	1.5047E-028	1.1752E-023
F19	t-value	**-5.8474**	-0.3211	**-10.8067**	**-10.4371**	**-9.6091**	**-19.9617**	**-226.0554**	**-12.9778**	**-13.8050**
	P-value	**3.1266E-056**	0.7488	**1.6986E-018**	**1.0919E-017**	**7.1240E-016**	**1.1653E-036**	**2.7373E-137**	**3.6192E-023**	**6.7153E-025**
F20	t-value	**-28.6217**	**-16.8961**	**-21.4613**	**-8.9055**	**-20.1565**	**-21.5993**	**-30.4881**	**-3.6433**	**-5.7887**
	P-value	**5.6868E-050**	**4.6189E-031**	**3.2389E-039**	**2.4706E-014**	**5.3441E-037**	**1.9101E-039**	**1.9531E-052**	**4.2891E-004**	**8.1973E-008**
F21	t-value	2.1951	12.8658	1.0347	2.8270	-0.1767	**-30.5350**	**-81.3720**	0.9691	0.9583
	P-value	0.0305	6.2418E-023	0.3033	0.0057	0.8601	**1.6998E-052**	**3.3968E-093**	0.3348	0.3402
F22	t-value	24.1021	23.7557	**-2.0456**	**-17.1919**	8.6535	**-112.8303**	**-112.4471**	5.5932	3.8399
	P-value	1.9057E-043	6.5492E-043	**0.0434**	**1.2646E-031**	8.7547E-014	**3.0907E-107**	**4.3314E-107**	1.9457E-007	2.1604E-004
F23	t-value	**-20.2145**	**-21.2048**	**-7.9816**	**-32.1256**	**-14.5629**	**-47.8733**	**-58.9189**	**-3.1932**	**-2.0000**
	P-value	**4.2403E-037**	**8.6988E-039**	**2.4892E-012**	**1.6874E-054**	**1.8600E-026**	**9.4019E-071**	**1.8396E-079**	**0.0019**	**0.0482**
F24	t-value	**-38.1674**	**-24.6808**	**-23.0811**	**-5.7348**	**-32.8638**	**-40.8526**	**-32.4370**	**-12.4664**	**-8.9344**
	P-value	**2.0138E-061**	**2.4918E-044**	**7.5219E-042**	**1.0417E-007**	**2.1136E-055**	**3.3698E-064**	**6.9912E-055**	**4.4066E-022**	**2.1364E-014**
F25	t-value	**-14.2268**	**-8.6510**	**-12.2327**	**-2.2713**	**-27.7312**	**-8.9162**	**-55.1002**	**-13.8591**	**-11.2041**
	P-value	**9.0526E-026**	**8.8623E-014**	**1.3918E-021**	**0.0253**	**9.4489E-049**	**2.3415E-014**	**1.2266E-076**	**5.1880E-025**	**2.3140e-019**
F26	t-value	**-7.7643**	20.1068	**-4.0847**	**-2.0958**	-1.1392	**-6.4337**	**-22.0891**	-0.5398	-0.6336
	P-value	**7.2740E-012**	6.5182E-037	**8.9103E-005**	**0.0386**	0.2574	**4.3302E-009**	**2.9806E-040**	0.5905	0.5278
F27	t-value	**-33.5633**	**-7.2751**	**-16.2398**	-0.5435	**-26.8338**	**-31.4261**	**-45.5340**	**-10.3301**	**-6.1269**
	P-value	**3.0597E-056**	**7.9297E-011**	**8.5119E-030**	0.5880	**1.7235E-047**	**1.2529E-053**	**1.1337E-068**	**1.8722E-017**	**1.7818E-008**
F28	t-value	**-5.9491**	1.4286	**-4.2166**	**-3.4336**	**-7.8172**	**-15.6462**	**-87.1051**	-0.9379	-1.6228
	P-value	**3.9934E-008**	0.1562	**5.4518E-005**	**8.6801E-004**	**5.6058E-012**	**1.2442E-028**	**4.1204E-096**	0.3506	0.1078
	General Merit	21	16	20	21	20	27	26	15	15

functions F1–F28 in PSEDA, 23 t values are negative on functions F1–F9, F11–F13, F15–F17, F19, F20, and F23–F28 in HMPSO-L, and 22 t values are negative on functions F1–F9, F12, F13, F15–F17, F19, F20, and F23–F28 in HMPSO-M. In brief, HMPSO outperforms its competitors on these benchmark functions.

Among all the obtained P values, 25 P values are much smaller than the 0.01 on functions F1–F5, F7, F9–F20, and F22–F28 in CPSO-H6, 24 P values are much smaller than the 0.01 on functions F2–F7, F9–F18, and F20–F27 in CLPSO, 23 P values are much smaller than the 0.01 on functions F2–F7, F9–F15, F17–F20, and F23–F28 in ALCPSO, 19 P values are much smaller than the 0.01 on functions F1, F2, F4, F5, F7, F9–F15, F18–F20, F22–F24, and F28 in FIPS, 25 P values are much smaller than the 0.01 on functions F1–F7, F9–F20, F22–F25, F27, and F28 in HPSO–TVAC, 26 P values are much smaller than the 0.01 on functions F1–F7, F9–F15, and F17–F28 in EDA–PSO, 26 P values are much smaller than the 0.01 on functions F1–F7, F9–F15, and F17–F28 in PSEDA, 17 P values are much smaller than the 0.01 on functions F1–F3, F5, F7, F9, F12, F13, F15, F18–F20, F22–F25, and F27 in HMPSO-L, and 15 P values are much smaller than the 0.01 on functions F1, F2, F4, F5, F7, F9, F12, F15, F18–F20, F22, F24, F25, and F27 in HMPSO-M. In brief, the experimental results obtained via HMPSO and other algorithms are statistically significant.

It seems that a two-tailed t-test in Table V is appropriate. However, there is no mention about the fact that data are normally distributed or not. In order to ensure the correctness of the experimental results, we simultaneously adopt another nonparametric test method, i.e., Wilcoxon's rank sum test. Such test on every benchmark function at a 0.05 significance level between HMPSO and other algorithms P values as shown in Table VI. Twenty-seven P values are much smaller than 0.05 on functions F1–F20 and F22–F28 in CPSO-H6, so are 25 P values on functions F2–F7, F9–F18, and F20–F28 in CLPSO, 27 P values on F1–F7 and F9–F28 in ALCPSO, 24 P values on F1, F2, F4–F7, F9–F15, F17–F26, and F28 in FIPS, 27 P values on F1–F7 and F9–F28 in HPSO–TVAC, 27 P values on F1–F7 and F9–F28 in EDA–PSO, 26 P values

TABLE VI
COMPARISONS BETWEEN HMPSO AND OTHER ALGORITHMS ON WILCOXON'S RANK SUM TEST AT A 0.05 SIGNIFICANCE LEVEL

Func	W-test	CPSO-H6	CLPSO	ALCPSO	FIPS	HPSO-TVAC	EDA-PSO	PSEDA	HMPSO-L	HMPSO-M
F1	P-value	**1.9882E-020**	0.3269	**1.7828E-020**	**2.1713E-008**	**4.8994E-021**	**1.9882E-020**	**1.9882E-020**	**3.2008E-020**	**1.1451E-019**
F2	P-value	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**6.2271E-016**	**2.4302E-012**	**3.3037E-018**	**3.3037E-018**	**7.2534E-004**	**1.7228E-007**
F3	P-value	**7.0854E-018**	**3.3037E-018**	**1.3434E-017**	0.3843	**1.5738E-015**	**3.3037E-018**	**3.3037E-018**	**6.9081E-004**	0.1919
F4	P-value	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	0.2035	**2.4327E-016**
F5	P-value	**8.0376E-019**	**3.9493E-008**	**7.6445E-019**	**2.4406E-012**	**7.9445E-019**	**8.0376E-019**	**8.0376E-019**	**7.6160E-019**	**7.8031E-019**
F6	P-value	**7.7921E-006**	**1.6326E-014**	**1.0746E-015**	**1.1791E-012**	**5.1389E-013**	**3.2987E-018**	**3.2987E-018**	**4.5038E-012**	**1.2990E-012**
F7	P-value	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**2.8406E-013**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**5.9455E-018**	**1.8407E-016**
F8	P-value	**0.0182**	0.5559	0.1391	0.2362	0.6880	0.0842	0.6880	0.6251	0.9360
F9	P-value	**3.7170E-018**	**1.8143E-014**	**1.2582E-014**	**4.7528E-017**	**3.1869E-017**	**3.3037E-018**	**3.3037E-018**	**7.9813E-008**	**1.4387E-008**
F10	P-value	**6.1616E-007**	**3.3030E-018**	**7.9212E-012**	**6.4893E-007**	**2.8474E-015**	**3.3030E-018**	**3.3030E-018**	0.1205	0.0879
F11	P-value	**3.3030E-018**	**3.3030E-018**	**3.3022E-018**	**1.8722E-010**	**3.3030E-018**	**3.3030E-018**	**3.3030E-018**	0.4616	0.1919
F12	P-value	**3.3037E-018**	**8.3812E-017**	**1.1111E-016**	**3.7444E-005**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**2.3533E-014**	**1.7653E-006**
F13	P-value	**3.3037E-018**	**1.2115E-013**	**6.0120E-015**	**0.0202**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**7.3218E-006**	0.4066
F14	P-value	**3.3037E-018**	**3.3037E-018**	**2.7027E-013**	**4.4344E-018**	**2.3278E-010**	**3.3037E-018**	**3.3037E-018**	**0.0178**	**0.0399**
F15	P-value	**1.2668E-015**	**3.1704E-015**	**7.7954E-006**	**3.3037E-018**	**5.5105E-006**	**3.3037E-018**	**3.3037E-018**	**2.9374E-004**	**2.5797E-004**
F16	P-value	**8.3982E-010**	**3.2972E-013**	**0.0146**	0.3318	**4.8606E-015**	**0.0483**	0.1373	0.5249	**0.0374**
F17	P-value	**3.3037E-018**	**3.3037E-018**	**0.0100**	**1.0637E-005**	**2.8743E-016**	**3.3037E-018**	**3.3037E-018**	**5.4957E-008**	**1.2880E-007**
F18	P-value	**2.8479E-015**	**1.1518E-013**	**4.8308E-007**	**3.9424E-018**	**8.9469E-018**	**3.3037E-018**	**3.3037E-018**	**6.6834E-017**	**2.1322E-017**
F19	P-value	**8.5611E-006**	0.7786	**8.9469E-018**	**8.2591E-015**	**1.9052E-013**	**3.3037E-018**	**3.3037E-018**	**3.3449E-015**	**8.9469E-018**
F20	P-value	**2.3591E-018**	**3.3037E-018**	**7.6301E-018**	**1.5084E-014**	**3.9127E-018**	**3.2440E-018**	**2.8493E-018**	**4.5323E-004**	**1.9199E-007**
F21	P-value	0.0776	**1.3841E-012**	**0.0034**	**7.9291E-004**	**9.5940E-004**	**2.7130E-018**	**2.7130E-018**	**0.0141**	**0.0106**
F22	P-value	**3.3037E-018**	**3.3030E-018**	**0.0263**	**2.6834E-017**	**4.1013E-012**	**3.3037E-018**	**3.3037E-018**	**3.9135E-008**	**2.0364E-004**
F23	P-value	**6.3037E-018**	**4.1812E-018**	**1.0477E-011**	**3.3037E-018**	**1.9002E-017**	**3.3037E-018**	**3.3037E-018**	**0.0030**	0.0891
F24	P-value	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.1146E-008**	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**2.6990E-015**	**3.5577E-012**
F25	P-value	**1.3434E-017**	**7.2139E-012**	**2.0582E-016**	**0.0272**	**3.3037E-018**	**2.9414E-012**	**3.3037E-018**	**4.4897E-017**	**6.0120E-015**
F26	P-value	**7.4308E-015**	**2.6735E-012**	**1.9052E-013**	**5.4439E-005**	**4.4544E-005**	**7.9976E-013**	**3.3037E-018**	**6.2551E-006**	**0.0110**
F27	P-value	**3.3037E-018**	**4.4372E-013**	**2.0129E-017**	0.8461	**3.3037E-018**	**3.3037E-018**	**3.3037E-018**	**3.2107E-014**	**1.2418E-007**
F28	P-value	**1.7667E-017**	**2.5357E-016**	**3.5558E-017**	**1.5187E-017**	**7.6853E-018**	**1.1389E-016**	**6.0455E-019**	**3.0366E-016**	**1.7361E-016**
	General Merit	27	25	27	24	27	27	26	23	22

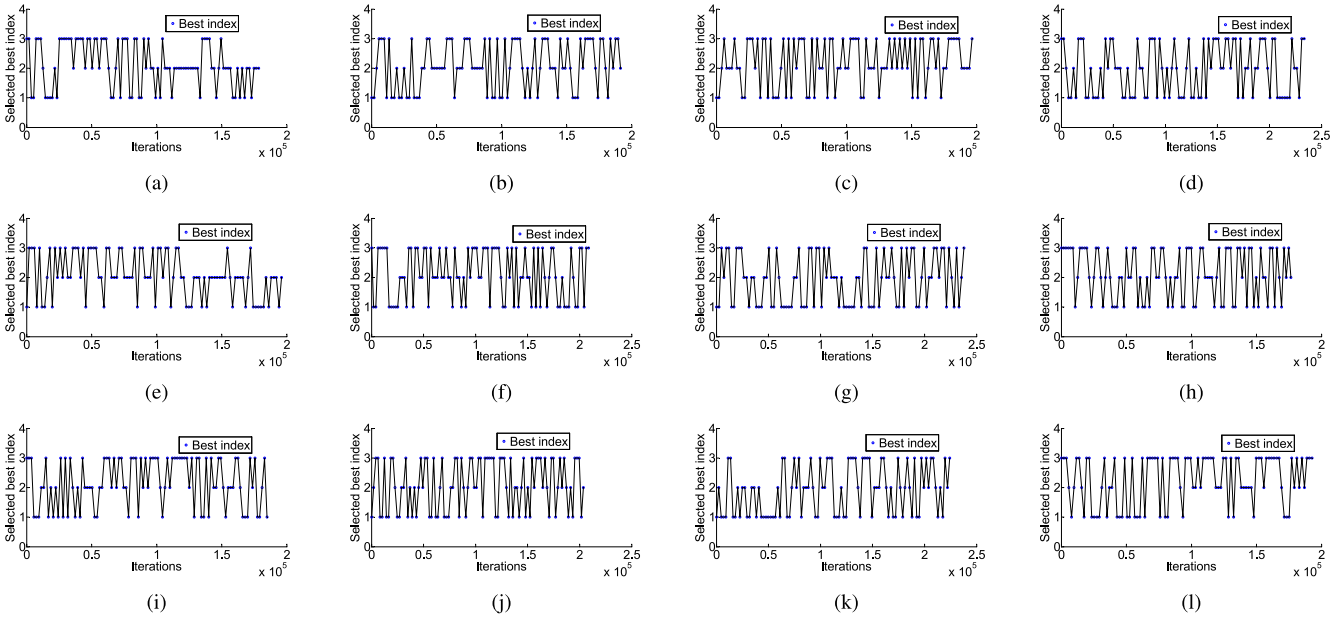


Fig. 5. Evolution of a selected best index in HMPSO on functions (a) F1, (b) F3, (c) F7, (d) F8, (e) F11, (f) F13, (g) F16, (h) F18, (i)–(k) F21–F23, and (l) F26.

on F1–F7, F9–F15, and F17–F28 in PSEDA, 23 P values on F1–F3, F5–F7, F9, F12–F15, and F17–F28 in HMPSO-L, and 22 P values on F1, F2, F4–F7, F9, F12, F14–F22, and F24–F28 in HMPSO-M. They suggest that the experimental results obtained via HMPSO and other algorithms are statistically significant.

In order to show the evolution of a selected best index in HMPSO clearly, we present the evolution curves on some functions, as described in Fig. 5. "1" denotes that the current best index is derived from the historical memory. "2" denotes that the current best index is derived from other particles' pbests as same as CLPSO. "3" denotes that the current best index is derived from the swarm's global gbest.

It is clear from Fig. 5 that with historical memory, HMPSO can pursue good solutions continuously through the interaction of three kinds of information, i.e., historical memory, particles' current pbests, and the swarm's gbest. The historical memory retains particles' historical promising pbests information, which can be reused so as to improve the search process. The historical memory can help the particles return to those optimum-likely regions that they have searched while keeping high population diversity.

V. Conclusion

In this paper, a new PSO algorithm is proposed, called HMPSO, which uses EDA to estimate and preserve the distribution information of particles' historical promising pbests. New particles are generated by the competition among those generated based on historical memory, particles' current pbests, and the swarm's gbest from an information pool. The three kinds of information have the distinct advantages and therefore they can complement one another. The structure of HMPSO is simple and it is easy to implement. Moreover, under our framework, the users can easily build their own information pool for solving different problems.

The significance of HMPSO mainly lies in two aspects. First, memory is an essential feature of all living systems, From the evolutionism point of view, it is worthwhile examining if historical memory is helpful to the process of evolution algorithms. In this paper, we make the first attempt on this topic to propose an HMPSO algorithm and show that historical memory built by particles' historical promising pbests is helpful to PSO. Second, the proposed HMPSO manages to prevent premature convergence and keeps the fast-converging feature via the competition among three kinds of information.

As shown in the experiments on 28 CEC2013 benchmark functions, HMPSO achieves the best results on all unimodal functions, and a key reason is that it uses the information of swarm's global gbest and particles' historical promising pbests that determine a promising search area. In addition, although HMPSO uses the information of the swarm's gbest, it presents excellent performance on most rotated multimodal functions and composition functions. This property indicates that HMPSO owns the abilities to effectively prevent premature convergence by particles' current pbests and historical memory. Next, two HMPSO variants with novel mechanisms to build historical memory are presented. One uses the historical pbests of a particle to build a local distribution model for that particle, while the other uses a multivariate Gaussian distribution to represent different components' relationship and takes it as the whole historical memory. The results indicate that the performance of HMPSO may be promoted through the help of these two mechanisms.

In the future, we first plan to design a coordination mechanism to integrate historical memory built by a local distribution model for each particle or a multivariate Gaussian distribution effectively. Secondly, we plan to assess the performance of HMPSO when applied to other optimization functions and industrial optimization problems, e.g., multi-objective evolutionary optimization [25] and optimal power flow problem [12]. Its use to solve some discrete optimization problems should be pursued [44]–[47].

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, WA, Australia, 1995, pp. 1942–1948.

[2] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing*, vol. 19, no. 2, pp. 431–448, Feb. 2015.

[3] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[4] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.

[5] A. Ratnaweera, S. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[7] W.-N. Chen *et al.*, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[8] W. Lin, X. Gu, Z. Lian, Y. Xu, and B. Jiao, "A self-government particle swarm optimization algorithm and its application in Texaco gasification," *J. Softw.*, vol. 8, no. 2, pp. 472–479, 2013.

[9] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.

[10] Z. H. Liu, J. Zhang, S. W. Zhou, X. H. Li, and K. Liu, "Coevolutionary particle swarm optimization using AIS and its application in multiparameter estimation of PMSM," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1921–1935, Dec. 2013.

[11] H. Duan, Q. Luo, Y. Shi, and G. Ma, "Hybrid particle swarm optimization and genetic algorithm for multi-UAV formation reconfiguration," *IEEE Comput. Intell. Mag.*, vol. 8, no. 3, pp. 16–27, Aug. 2013.

[12] Q. Kang, M. Zhou, J. An, and Q. Wu, "Swarm intelligence approaches to optimal power flow problem with distributed generator failures in power networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 2, pp. 343–353, Apr. 2013.

[13] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Parallel Problem Solving from Nature—PPSN* (LNCS 1141). Berlin, Germany: Springer, 1996, pp. 178–187.

[14] E. Bengoetxea and P. Larrañaga, "EDA-PSO: A hybrid paradigm combining estimation of distribution algorithms and particle swarm optimization," in *Swarm Intelligence*. Berlin, Germany: Springer, 2010, pp. 416–423.

[15] V. Santucci and A. Milani, "Particle swarm optimization in the EDAs framework," in *Soft Computing in Industrial Applications*. Berlin, Germany: Springer, 2011, pp. 87–96.

[16] Y. Shi, H. Liu, L. Gao, and G. Zhang, "Cellular particle swarm optimization," *Inf. Sci.*, vol. 181, no. 20, pp. 4460–4493, 2011.

[17] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progr. Artif. Intell.*, vol. 1, no. 1, pp. 103–117, 2012.

[18] J.-J. He, J.-H. Xiao, X. Shi, and T. Song, "A membrane-inspired algorithm with a memory mechanism for knapsack problems," *J. Zhejiang Univ. Sci. C*, vol. 14, no. 8, pp. 612–622, 2013.

[19] U. F. Siddiqi, Y. Shiraishi, M. Dahb, and S. M. Sait, "A memory efficient stochastic evolution based algorithm for the multi-objective shortest path problem," *Appl. Soft Comput.*, vol. 14, pp. 653–662, Jan. 2014.

[20] B. Jeddi and V. Vahidinasab, "A modified harmony search method for environmental/economic load dispatch of real-world power systems," *Energy Convers. Manage.*, vol. 78, pp. 661–675, Feb. 2014.

[21] C. Stone and L. Bull, "Evolution of cellular automata with memory: The density classification task," *Biosystems*, vol. 97, no. 2, pp. 108–116, 2009.

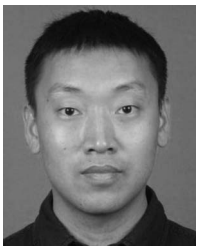
[22] U. F. Siddiqi, Y. Shiraishi, M. Dahb, and S. M. Sait, "Finding multi-objective shortest paths using memory-efficient stochastic evolution based algorithm," in *Proc. IEEE 3rd Int. Conf. Netw. Comput. (ICNC)*, Okinawa, Japan, 2012, pp. 182–187.

[23] Y. Zhong and L. Zhang, "Sub-pixel mapping based on artificial immune systems for remote sensing imagery," *Pattern Recognit.*, vol. 46, no. 11, pp. 2902–2926, 2013.

[24] R. Kundu *et al.*, "Improved CMA-ES with memory based directed individual generation for real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 748–755.

[25] Y. Wang and B. Li, "Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 630–637.

[26] T. Wang, Z. Chen, K. Li, X. Deng, and D. Li, "Memory does not necessarily promote cooperation in dilemma games," *Phys. A, Statist. Mech. Appl.*, vol. 395, pp. 218–227, Feb. 2014.
[27] M. K. Manshad, A. K. Manshad, and M. R. Meybodi, "Memory/search RCLA-EC: A CLA-EC for moving parabola problem," in *Proc. IEEE 6th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, Seogwipo, Korea, 2011, pp. 732–737.
[28] M. Brunato and R. Battiti, "R-EVO: A reactive evolutionary algorithm for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 770–782, Dec. 2011.
[29] X. Peng, X. Gao, and S. Yang, "Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments," *Soft Comput.*, vol. 15, no. 2, pp. 311–326, 2011.
[30] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4348–4360, 2011.
[31] Y. Zhou, J. Wang, and J. Yin, "A discrete estimation of distribution particle swarm optimization for combinatorial optimization problems," in *Proc. IEEE 3rd Int. Conf. Nat. Comput.*, vol. 4. Haikou, China, 2007, pp. 80–84.
[32] J. Wang, Z. Kuang, X. Xu, and Y. Zhou, "Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9398–9408, 2009.
[33] J. Wang, Y. Zhang, Y. Zhou, and J. Yin, "Discrete quantum-behaved particle swarm optimization based on estimation of distribution for combinatorial optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 897–904.
[34] J. Wang, "Genetic particle swarm optimization based on estimation of distribution," in *Bio-Inspired Computational Intelligence and Applications*. Berlin, Germany: Springer, 2007, pp. 287–296.
[35] R. M. Alguliev, R. M. Aliguliyev, and C. A. Mehdiyev, "An optimization model and DPSO-EDA for document summarization," *Int. J. Inf. Technol. Comput. Sci.*, vol. 3, no. 5, pp. 59–68, 2011.
[36] M. El-Abd and M. S. Kamel, "PSO_Bounds: A new hybridization technique of PSO and EDAs," in *Foundations of Computational Intelligence*. Berlin, Germany: Springer, 2009, pp. 509–526.
[37] M. El-Abd, "Preventing premature convergence in a PSO and EDA hybrid," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 3060–3066.
[38] M. Iqbal and M. A. M. de Oca, "An estimation of distribution particle swarm optimization algorithm," in *Ant Colony Optimization and Swarm Intelligence*. Berlin, Germany: Springer, 2006, pp. 72–83.
[39] R. V. Kulkarni and G. K. Venayagamoorthy, "An estimation of distribution improved particle swarm optimization algorithm," in *Proc. IEEE 3rd Int. Conf. Intell. Sensors Sensor Netw. Inf.*, Melbourne, QLD, Australia, 2007, pp. 539–544.
[40] C. W. Ahn, J. An, and J.-C. Yoo, "Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs," *Inf. Sci.*, vol. 192, pp. 109–119, Jun. 2012.
[41] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-163, 1994.
[42] J. J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Dıaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, 2013.
[43] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[44] F. Chu, Z. Zhu, and S. Mammar, "Recent advances and issues in facility location problems," in *Contemporary Issues in Systems Science and Engineering*, M. C. Zhou, H.-X. Li, and M. Weijnen, Eds. Hoboken, NJ, USA: IEEE Press/Wiley, 2015, pp. 817–834.
[45] K. Xing, L. Han, M. Zhou, and F. Wang, "Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 603–615, Jun. 2012.
[46] J. Sun, Q. Zhang, and X. Yao, "Meta-heuristic combining prior online and offline information for the quadratic assignment problem," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 429–443, Mar. 2014.
[47] W. Dong and M. C. Zhou, "Gaussian classifier-based evolutionary strategy for multimodal optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1200–1216, Jun. 2014.



Jie Li received the B.S. degree in computer science from the Henan University of Science and Technology, Luoyang, China, in 2003, and the M.S. degree in computer science from Tongji University, Shanghai, China, in 2009. He is currently pursuing the Ph.D. degree in computer science at Tongji University, Shanghai, China.

His current research interests include machine learning, intelligent algorithm, learning automata, particle swarm optimization, and estimation of distribution algorithm.

JunQi Zhang received the Ph.D. degree in computing science from Fudan University, Shanghai, China, in 2007.

He became a Post-Doctoral Research Fellow and a Lecturer with the Key Laboratory of Machine Perception, Ministry of Education in Computer Science at Peking University, Beijing, China, in 2007. He is currently an Associate Professor with the Department of Computer Science and Technology, Tongji University, Shanghai. His current research interests include machine learning, intelligent and learning automata, computational intelligence, particle swarm optimization, statistical learning, high-dimensional index, and multimedia data management.

Prof. Zhang was a recipient of the Outstanding Post-Doctoral Award from Peking University and the Financial Support from the China Postdoctoral Science Foundation.

ChangJun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995.

He is currently a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. His current research interests include concurrency theory, Petri nets, formal verification of software, clusters, grid technology, program testing, intelligent transportation systems, and service-oriented computing. He has authored over 100 publications.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and is currently a Distinguished Professor of Electrical and Computer Engineering. His current research interests include Petri nets, Internet of things, semiconductor manufacturing, transportation, and energy systems. He has over 600 publications including 12 books, 290+ journal papers (majority in the IEEE TRANSACTIONS), and 28 book chapters.

Prof. Zhou is the Founding Editor of the IEEE Press Book Series on Systems Science and Engineering. He is a fellow of the International Federation of Automatic Control and the American Association for the Advancement of Science.