Information Sciences 192 (2012) 109–119

Contents lists available at ScienceDirect

Information Sciences

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

Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs

Chang Wook Ahn [a], Jinung An [b], Jae-Chern Yoo [a,*]

[a] School of Information and Communication Engineering, Sungkyunkwan University, Republic of Korea
[b] Daegu Gyeongbuk Institute of Science and Technology (DGIST), Republic of Korea

ARTICLE INFO

Article history:
Available online 23 July 2010

Keywords:
Estimation of distribution algorithms
Particle swarm optimization
Extended compact genetic algorithm
Local search
Global search
Probabilistic model building

ABSTRACT

This paper presents a novel framework of the *estimation of particle swarm distribution algorithms* (EPSDAs). The aim is to effectively combine *particle swarm optimization* (PSO) with the *estimation of distribution algorithms* (EDAs) without losing their unique features. This aim is achieved by incorporating the following mechanisms: (1) selection is applied to the local best solutions in order to obtain more promising individuals for model building, (2) a probabilistic model of the problem is built from the selected solutions, and (3) new individuals are generated by a stochastic combination of the EDA's model sampling method and the PSO's particle moving mechanism. To exhibit the utility of the EPSDA framework, an *extended compact particle swarm optimization* (EcPSO) is developed by combining the strengths of the *extended compact genetic algorithm* (EcGA) with *binary PSO* (BPSO), along the lines of the suggested framework. Due to its effective nature of harmonizing the global search of EcGA with the local search of BPSO, EcPSO is able to discover the optimal solution in a fast and reliable manner. Experimental results on artificial to real-world problems have adduced grounds for the effectiveness of the proposed approach.

1. Introduction

Nature often signals paradigm shifts in the cutting-edge computing technologies [1,33]. Evolutionary computing and swarm intelligence are such representative instances. They have flaunted an enviable success record in dealing with a variety of real-world applications.

Particle swarm optimization (PSO), which is inspired from the flocking behavior of birds, is a stochastic search and optimization algorithm [9,18]. In PSO, each individual, named a *particle*, traverses search space by learning from the historical experience of a population, called a *swarm*. PSO operates, gracefully adjusting the trajectory of each particle towards its own best (i.e., local best) position and the global best position discovered by its neighbors, as well as the whole swarm. It is effective in discovering multiple (local best) solutions. Although sharing/exchanging information between particles takes place to some extent, it is performed in a strictly limited fashion. In other words, only the global best position at each iteration/generation is exploited as the reference of the coevolution between particles (i.e., the knowledge exchange between particles). Thus, PSO is thought of as a weak-cooperative search method, i.e., the local-search intensive.

For the last two decades, *genetic and evolutionary algorithms* (GEAs) have been successfully applied to various engineering and scientific problems [1,10,11]. In recent days, a new paradigm of GEAs has received great attention. This new paradigm is the *estimation of distribution algorithms* (EDAs), also known as *probabilistic model building genetic algorithms* [20,27]. Combining the linkage learning with the graphical probabilistic modeling under the features of natural selection, EDAs are good at

* Corresponding author.
E-mail addresses: cwan@skku.edu (C.W. Ahn), yoojc@skku.edu (J.-C. Yoo).

the automatic discovery and exploitation of problem regularities. Unlike PSO, a population evolves with intensive communication among all of the promising individuals, ensuring an effective mixing and reproduction of superior partial solutions, i.e., *building blocks* (BBs), thereby readily extracting promising regions of search space and, thus, reliably discovering the global optimum [1,27]. Thus, EDAs are regarded as a strong-cooperative search method, i.e., the global-search intensive.

In recent years, some efforts to effectively combine the strengths of PSO and EDAs have been tried [16,34–36]. They started from an investigation such that each particle may benefit from the global statistic gathered from all of the local best positions (i.e., solutions) of a swarm. Accordingly, they have better performances than extant PSO schemes on widely-known benchmark problems. However, the research has been performed with two limited factors. First, only a *univariate marginal distribution* (UMD) model has been the focus. Usually, the search capability of algorithms can be significantly improved by identifying and exploiting the regularities of problems, but the UMD model-based approach fails to do so [11,13,14,20,23,26,27]. Typically, such a task can be achieved by incorporating more complex *multivariate distribution* (MD) models [1,11,20,27]. Second, using all of the local best solutions, in the context of probabilistic model building, may not be apt for obtaining meaningful/useful information, since good models can be learned from a set of solutions that are properly fitted in terms of the exploration of search space and the exploitation of solutions found thus far. The local best solutions discovered by the PSO mechanism (i.e., a kind of local search), on the other hand, mainly focus on the exploration of the search space. It is desirable to apply the natural selection concept in the level of local best solutions (as in EDAs) in order to further promote the exploitation of solutions. The above matters are the underlying motivation for this study.

In a similar context, a few efforts to strike a balance between the recent nature-inspired algorithms and the advanced EDAs have been made [5,31]. For instance, DE/EDA [31] has been developed by incorporating the global information extracted by a multivariate EDA (e.g., a Gaussian model with a diagonal covariance matrix) with the local information obtained by a differential evolution, and BAIS [5] has been proposed by replacing the traditional cloning and mutation procedures of the artificial immune system with the probabilistic model building and sampling methods of the Bayesian optimization algorithm. Further issues, however, are beyond the scope of this paper.

This paper presents a novel framework of the *estimation of particle swarm distribution algorithms* (EPSDAs) with a view to bring together the benefits of PSO and EDAs. For demonstrating the utility of the suggested framework, an *extended compact particle swarm optimization* (EcPSO) is developed by following the framework of the EPSDAs. The rest of the paper is organized as follows. Section 2 provides some background knowledge of PSO and EDAs. Section 3 describes the framework of EPSDAs and also presents EcPSO in detail. Performance tests are conducted in Section 4, and the paper concludes with a summary in Section 5.

2. Existing algorithms

This section briefly provides some background information on PSO and EDAs.

2.1. Particle swarm optimization

In principle, PSO emulates the dynamics of bird flocking to solve optimization problems [9,18]. It is characterized by a moving process, such that each particle flies through the multidimensional search space while its velocity and position are updated by its own (or neighbor's) best information, as well as the best knowledge of the entire swarm.

Initially, PSO is equipped with a swarm of N possible particles (i.e., solutions) randomly generated in an n-dimensional search space. A particle i at its tth iteration consists of a position vector, $x_i(t)$, and a velocity vector, $v_i(t)$. At each iteration, the particle i is updated by means of two important vectors. The first vector is $pbest_i$, which amounts to the best solution (i.e., position) found by the particle i, so far. The second vector is $gbest(t)$, which is the best of $pbest_k(t)$, $\forall k$. Unless the termination criteria are met, each particle is updated in each iteration by the following rule:

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot r_1 \cdot (pbest_{ij}(t) - x_{ij}(t)) + c_2 \cdot r_2 \cdot (gbest_j(t) - x_{ij}(t)), \tag{1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{2}$$

where i ($\in\{1,\ldots,N\}$) is the particle's index in the swarm and j ($\in\{1,\ldots,n\}$) denotes the component's index of the corresponding particle. Also, w is the positive inertia weight, c_1 and c_2 are the cognitive and social learning factors, respectively, and r_1 and r_2 are two random numbers uniformly distributed from zero to one [9,18]. Note that w controls the influence of the past velocity on the present one, and c_1 and c_2 determine the effect of the local best and the global best solutions on the present velocity of the particle, respectively.

Physically, each particle traverses the search space by striking a balance between its own and its social experiences [9,18,25]. Mathematically, a particle evolves with its previous velocity, its own best information, and the global best knowledge of the whole swarm. It has been empirically observed that PSO is a good alternative when solving diverse global optimization problems [25]. Moreover, some theoretical and empirical studies on the optimal inertia weight and the optimal learning factors have been performed [4,6].

With a similar concept, employing a sigmoid function, with a view of solving combinatorial optimization problems, has developed a discrete version of PSO, also known as *binary PSO* (BPSO) [19]. In principle, the velocity has been regarded as the probability of taking '1' or '0' on a bit (i.e., position). The velocity is updated by the same rule of Eq. (1); but the position renewal is conducted by the following rule [19]:

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r \leqslant sig(v_{ij}(t+1)), \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

where r is a random number in $[0,1]$ and $sig(\cdot)$ is the sigmoid function that transforms the velocity into the probability, according to the following expression:

$$sig(v_{ij}(t)) = \frac{1}{1+exp\{-v_{ij}(t)\}}. \tag{4}$$

Recently, a *modified BPSO* (mBPSO) has been developed from the concept of the genotype-phenotype mapping [21]. It replaces the position renewal of BPSO (Eq. (3)) with the following rules:

$$x_{ij}^{(g)}(t+1) = x_{ij}^{(g)}(t) + v_{ij}(t+1), \tag{5}$$

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r \leqslant sig\left(x_{ij}^{(g)}(t+1)\right), \\ 0 & \text{otherwise}, \end{cases} \tag{6}$$

where $x_{ij}^{(g)}(t)$ and $x_{ij}(t)$ represent the genotypic and phenotypic positions of the jth dimension of the ith particle, respectively. Overall, mBPSO has offered an improved performance over the traditional BPSO.

Naturally, PSO is apt for locally discovering better solutions. However, every particle tends to search around its own previous best positions without achieving any improvement [16]. It denotes that the explored regions are quite often revisited by the swarm (without success), thereby bringing forth unnecessary search. This arises from two critical problems of PSO. First, it does not learn/treat the whole swarm as a single entity [16]. Second, it does not take into account any regularity of problems (e.g., dependence of variables).

2.2. Estimation of distribution algorithms

EDAs have attracted much interest in recent years [1,20,27]. Incorporating the linkage learning techniques into the graphical probabilistic modeling, EDAs construct promising probabilistic models from superior solutions found thus far, which uncover some important regularities of problems. Exploiting the probabilistic models to traverse the search space, EDAs efficiently evolve the whole population towards the promising regions of the global optimum. In general, EDAs iterate the following three phases, until the termination criteria are satisfied:

1. Select good individuals from a population.
2. Estimate the probability distribution from the selected individuals.
3. Generate new individuals (i.e., offspring) from the estimated distribution.

The last phase replaces the traditional recombination and mutation operators of GEAs. EDAs ensure an effective mixing and reproduction of BBs due to their ability to accurately discover and utilize the structure of a given problem. It results in solving hard problems with a linear or sub-quadratic scalability in terms of fitness function evaluations [1,26,27]. However, there is a trade-off between the accuracy of the estimated distribution and the efficiency of the computation. For instance, a complicated model is preferable if the fitness function is computationally expensive.

Depending on how intricate and involved the probabilistic models are, they are generally classified into two categories: *no dependencies* and *dependencies*. The former assumes statistical independence of all the (decision) variables of the problem. All the algorithms in this category compute the probability distribution as a product of univariate distributions [20,27]. It approximates the order-one behavior of GEAs with a uniform crossover. However, the assumption appears to be unrealistic in view of the fact that many optimization problems involve certain types of interaction between variables. Thus, it gives rise to the disruption of BBs, thereby failing to converge to the optimum. The *population-based incremental learning* (PBIL) [3], the *compact genetic algorithm* (cGA) [12], and the *univariate marginal distribution algorithm* (UMDA) [22] are representative examples.

The second category assumes that the decision variables statistically interact with each other to some extent. In this category, all the algorithms endeavor to use general probabilistic models in such a manner of accurately learning the information on dependencies (i.e., BBs) and exploiting the knowledge for performing a better search [20,27]. At the expense of some computational cost in the probabilistic model learning, the overall time complexity for decomposable problems can be significantly improved by reducing the number of fitness function evaluations until the optimum is reached. Accordingly, the approach is able to solve many difficult problems quickly, accurately, and reliably. The *extended compact genetic algorithm* (EcGA) [13,14], the *factorized distribution algorithm* (FDA) [23], and the *Bayesian optimization algorithm* (BOA) [26] are widely-known in this regard. Recently, the adaptive estimated maximum-entropy distribution algorithm [32], the assignment function-based BOA [24], and the eigen decomposition EDA [8] have been introduced.

In general, EcGA has been recognized as a competent EDA in the way that an appropriate multivariate distribution model is employed in terms of the modeling accuracy and the computational efficiency [13,14]. It provides a direct linkage map of decision variables (i.e., BB information) by means of a *marginal product model* (MPM). Incorporating the *minimal description length* (MDL) concept over model complexity and (compressed) population complexity, EcGA learns the MPM with a greedy search algorithm. Further details of EcGA are described below.

EcGA starts with a set of randomly generated individuals. After evaluating their fitness values, tournament selection is performed. A probabilistic model is then learned from the selected individuals, and an MPM is employed as the feasible probability distribution model. Note that the MPM partitions (decision) variables into several mutually independent groups and then computes marginal probabilities for each (linkage) group. In the model building phase, a greedy search is performed with the MDL metric. The metric defined in Eq. (7) consists of two terms, *model complexity* and *population complexity*

$$\begin{aligned} \min \quad & \log_2(N)\sum_{i=1}^{m}(2^{k_i}-1) - N\sum_{i=1}^{m}\sum_{j=1}^{2^{k_i}} p_{ij}\log_2(p_{ij}) \\ \text{s.t.} \quad & 2^{k_i} \leqslant N, \quad \forall i \in \{1,\ldots,m\}, \end{aligned} \tag{7}$$

where N is the population size, m is the number of partitions, and k_i is the number of variables in the ith partition.[1] Further, p_{ij} denotes the probability of the jth variable sequence in the ith partition.

In Eq. (7), the first term represents the number of bits needed to store all the marginal probabilities, and the second term amounts to the entropy of the marginal distribution over all of the partitions [13,14]. The (greedy) search method discovers a promising model structure by incrementally merging two partitions (i.e., subsets) that yield the highest improvement in the MDL metric. The merging process continues until no more improvement is achieved. Lastly, new individuals are generated by sampling the final probabilistic model. The above procedures are iterated unless the termination criteria are satisfied.

As aforementioned, EDAs excel at discovering the promising region (of search space) in which the global optimum exists [1,14,27]. Only focusing on utilizing the global information of the population, however, EDAs are not efficient in rapidly leading individuals toward the closest local optimum.

3. Proposed approach

This section presents the framework of EPSDAs and then demonstrates its utility by designing EcPSO. Also, its computational time complexity is mentioned.

3.1. Estimation of particle swarm distribution algorithms

In principle, PSO and EDAs are very competent in local search and global search, respectively. With this in view, EPSDAs have been designed to obtain the best of both approaches, but only a small amount of work has been done in this regard [16,34–36]. In [34–36], combining PSO with PBIL developed a simple version of an EPSDA. Its performance has been investigated for widely-known combinatorial optimization problems, such as knapsack and graph partitioning. In [16], another EPSDA has been proposed by means of the mixture of one-dimensional Gaussian functions, which approximates the distribution of the good regions in each dimension. This can be viewed as a combination of PSO and a continuous PBIL with the Gaussian kernel. Its effectiveness has been verified with several numerical optimization benchmarks, such as Rosenbrock and Griewangk functions.

Conceptually, the above-mentioned approaches have tried to exploit the global statistic collected from all of the local best solutions (of the whole swarm) in order to efficiently perform the search. Moreover, the statistic has been independently measured in each dimension. Thus, they are very effective in solving problems with independent variables since UMD models have been genuinely incorporated. However, they may suffer when it comes to solving real-world problems due to some important interactions of variables with which BBs are associated. To find the optimum of such problems, it is necessary to discover and preserve the BBs. This can be naturally achieved by incorporating MD models. As described earlier, a set of solutions well configured in terms of exploration and exploitation are needed for good probabilistic model building. Taking all the local best solutions may not be a proper choice for model construction, because the local best solutions formed by the search mechanism of PSO mainly promote the exploration of the search space. In other words, it is lacking in the exploitation of solutions since the (single) global best solution is the only source of exploitation. The problem can be simply resolved by applying the selection to the local best solutions (as done in EDAs).

Let $\mathcal{P}(t) = \{X(t), \widetilde{X}(t), \Theta(t), V(t)\}$ be a particle swarm at the tth generation, which consists of particles $X(t)$, their local best positions $\widetilde{X}(t)$, the global best position $\Theta(t)$ defined as the best one among $\widetilde{X}(t)$, and the particles' velocities $V(t)$.[2] With the terminologies, the framework of EPSDAs can be described as follows:

[1] Thus, $\sum_{i=1}^{m} k_i$ equals the individual length (i.e., problem size n).

[2] The term 'local best' is also known as 'particle best.' Moreover, 'position' is identically used as 'solution.'

```
STEP 1. INITIALIZATION.
    Randomly generate the initial particles $X(0)$.
STEP 2. EVALUATION.
    Evaluate the fitness values for all particles $X(t)$.
STEP 3. UPDATE.
    Update the local best solutions $\widetilde{X}(t)$, the global best solution $\Theta(t)$, and the particles' velocities $V(t)$.
STEP 4. SELECTION.
    Select a set of promising solutions, $\widetilde{X}^{\mathcal{S}}(t)$, from the local best solutions $\widetilde{X}(t)$.
STEP 5. LEARNING.
    Learn a probabilistic model $\mathcal{M}(t)$ from the selected local best solutions, $\widetilde{X}^{\mathcal{S}}(t)$, using a metric (and constraints).
STEP 6. CREATION.
    Generate new particles $X(t+1)$ from both the estimated distribution $\mathcal{M}(t)$ and the current particle swarm $\mathcal{P}(t)$.
STEP 7. TERMINATION.
    If the termination criteria are not met, go to STEP 2.
```

Unlike PSO and EDAs, STEPS 5 and 6 uniquely characterize EPSDAs as they harmonize the local search of PSO with the global search of EDAs. Thus, EPSDAs are quite good at simultaneously performing the local search and the global search in a cooperative manner. The global search is performed by the EDA process of generating new individuals from the probabilistic model built on the selected set of solutions. The local search is done with the PSO process of looking for a better solution in the way of each (current) local best position. Both features can be clearly carried through by STEPS 5 and 6 because new particles are generated by assembling several partial solutions obtained from both the probabilistic model (of EDAs) and the particle flying model (of PSO). Moreover, the mechanism is processed with the information on variable dependencies. Thus, any algorithm developed under the above suggested framework can solve hard problems quickly, accurately, and reliably.

3.2. Extended compact particle swarm optimization

Along the lines of the framework of EPSDAs, EcPSO is developed for supporting its utility. The design goal of EcPSO is to effectively combine the BPSO with the traditional EcGA without compromising their unique features.

Due to its ability to learn and promote BBs, EcGA is able to explore better regions of the search space in a fast and reliable manner (see Section 2.2). Thus, EcGA is very apt at the global search. Meanwhile, PSO can independently traverse the search space because each particle mainly evolves/flies with its own best experience and the best knowledge of the swarm (see Section 2.1). Once some particles access the basin of the attractors, the PSO tends to quickly converge to its optimum. Thus, PSO excels at a fast (multiple) local search. In essence, combining the global search and the local search methods may produce a powerful search mechanism, striking a balance between them. The idea is to apply a proper selection method to the set of local best solutions (as explained in Section 3.1) and then learn an MPM from the selected solutions. After that, new particles are created by merging the MPM sampling method of EcGA and the particle evolution model of PSO. Detailed procedures are described below.

A particle swarm $\mathcal{P}(t)$ consists of particles $X(t)$, their local best solutions $\widetilde{X}(t)$, the global best solution $\Theta(t)$, and the particles' velocities $V(t)$, that is, $\mathcal{P}(t) = \{X(t), \widetilde{X}(t), \Theta(t), V(t)\}$. Let $X_i(t) = \{X_{i,k}(t) \in \{0,1\} \mid k = 1,\ldots,n\}$ be the ith particle in the n-dimensional space at the tth generation, and denote the value of its kth variable as $X_{i,k}(t)$. Similarly, all notations with regard to $\widetilde{X}(t)$ and $V(t)$ can be naturally interpreted. For the global best solution, $\Theta_k(t)$ denotes the value of the kth variable of $\Theta(t)$. An MPM learned at the tth generation is denoted as $\mathcal{M}(t)$. Moreover, m is the number of partitions (i.e., BBs), and Z_k represents a set of (decision) variables belonging to the kth partition. Thus, a problem can be decomposed into several disjoint sub-problems, that is, $X_i(t) = \bigcup_{k=1}^{m} X_{i,Z_k}(t)$ and $\phi = \bigcap_{k=1}^{m} X_{i,Z_k}(t)$. Meanwhile, the sub-MPM corresponding to the sub-problem Z_k, is denoted by $\mathcal{M}_{Z_k}(t)$. An indicator function, $I_\alpha(r)$, is defined as

$$I_\alpha(r) = \begin{cases} 1 & \text{if } r \leqslant \alpha, \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

where r is a random number in [0,1]. Furthermore, $sampling(\mathcal{M}_{Z_k}(t))$ creates a partial particle by sampling the sub-MPM $\mathcal{M}_{Z_k}(t)$ (as done in [13,14]), and $flying(\mathcal{P}_{i,Z_k}(t))$[3] generates a part of the ith particle by emulating the particle flying model (see Eqs. (1)–(4)) in the level of sub-space Z_k. With these terminologies, EcPSO operates as follows:

```
STEP 1. INITIALIZATION.
    Randomly generate the initial $N$ particles $X(0)$.
STEP 2. EVALUATION.
    Evaluate the fitness values for all particles $X(t)$.
STEP 3. UPDATE.
    Update the local best solutions $\widetilde{X}(t)$, the global best solution $\Theta(t)$, and the particles' velocities $V(t)$.
```

[3] It is obvious that $\mathcal{P}_{i,Z_k}(t) = \{X_{i,Z_k}(t), \widetilde{X}_{i,Z_k}(t), \Theta_{Z_k}(t), V_{i,Z_k}(t)\}$.

STEP 4. SELECTION.
Apply a tournament selection to $\widetilde{X}(t)$ and obtain a set of selected solutions, $\widetilde{X}^{\mathcal{S}}(t)$.
STEP 5. LEARNING.
Learn an MPM $\mathcal{M}(t)$ from the set $\widetilde{X}^{\mathcal{S}}(t)$ using the MDL metric and the greedy search.
STEP 6. CREATION.
Generate new N particles $X(t+1)$ by the following rule:

$$X_i(t+1) = \bigcup_{k=1}^{m} \{I_\alpha(r) \cdot \mathit{flying}(\mathcal{P}_{i,Z_k}(t)) + (1 - I_\alpha(r)) \cdot \mathit{sampling}(\mathcal{M}_{Z_k}(t))\}. \tag{9}$$

STEP 7. TERMINATION.
If the termination criteria are not met, go to STEP 2.

As is usual with EDAs, EcPSO starts with the initial randomly generated N particles, $X(0)$ (see STEP 1). After evaluating the fitness values for all current particles, $X(t)$, in STEP 2, the update process of STEP 3 is performed, as follows: (1) update the local best solutions by $\widetilde{X}_i(t) = \arg\max_{x \in \{\widetilde{X}_i(t-1), X_i(t)\}} f(x)$, where $f(\cdot)$ is the fitness function,[4] (2) update the global best solution by $\Theta(t) = \arg\max_{x \in \{\widetilde{X}_i(t) |\ \forall i \in \{1,\ldots,N\}\}} f(x)$, and (3) update the velocities by Eq. (1), which can be rewritten as

$$V_{i,j}(t) = w \cdot V_{i,j}(t-1) + c_1 \cdot r_1 \cdot (\widetilde{X}_{i,j}(t) - X_{i,j}(t)) + c_2 \cdot r_2 \cdot (\Theta_j(t) - X_{i,j}(t)). \tag{10}$$

The tournament selection is then applied to the updated local best solutions, $\widetilde{X}(t)$, by which a set of selected individuals, $\widetilde{X}^{\mathcal{S}}(t)$, is obtained (see STEP 4). Recall that the tournament selection (without replacement) works by randomly choosing non-overlapping s individuals[5] and selecting the best one as a parent for the next generation. With the selected solutions $\widetilde{X}^{\mathcal{S}}(t)$, an MPM $\mathcal{M}(t)$ is learned by utilizing the MDL metric given in Eq. (7) and the greedy search incrementally merging two partitions (see STEP 5). In STEP 6, a set of new particles are generated by properly combining the MPM sampling of EcGA (as in [13,14]) with the particle flying process of BPSO (i.e., Eqs. (1)–(4)). Finally, the above steps are iterated until the termination criteria are satisfied (see STEP 7).

Note that EcPSO can be characterized by the method of creating new particles given in STEP 6. It assembles/mixes partial solutions created by the sampling of MPM and the flying model of current particles in the level of sub-problems (i.e., partitions or sub-space). Here, the mixing rate of partitions can be adjusted by changing the value of parameter α of $I_\alpha(r)$. That is, EcPSO becomes the global-search (local-search) intensive as α goes to 0 (1). In this way, EcPSO can effectively bring together the strengths of both worlds: the global search of EcGA and the local search of PSO.

3.3. Computational time complexity

In general, the computational time complexity is an important issue in designing optimization algorithms. This section briefly makes mention of the issue in this regard.

In many real-world problems, the time spent evaluating a candidate solution usually outweighs the time required by the rest of the algorithmic components [29]. This implies that the number of (fitness function) evaluations until a reliable convergence to the optimum can be the measure of the (computational) time complexity of algorithms. Meanwhile, the computational overhead usually increases as a low-order polynomial of the problem size [29]. Thus, the growth of the number of evaluations in terms of the problem size until the global convergence is the more appropriate measure of time complexity, which is also known as *scalability*.

We assume decomposable problems with bounded difficulty for tractability. Such problems are formed by concatenating subfunctions, in which the overall fitness is the sum of contributions from all subfunctions that are disjoint with a uniform order [2,26,29]. Note that the number of evaluations can be computed as a product of the population size needed for discovering the problem regularities (i.e., population complexity) and the number of generations required for population convergence (i.e., convergence complexity) [2,29]. Some efforts to analyze the time complexity (i.e., scalability) of EDAs have been successfully made [2,29,30]. Based on those results, we shortly investigate the time complexity of EcGA, EcPSO, and mBPSO.

To start with, EcGA is considered. As aforementioned, the time complexity consists of population complexity and convergence complexity. For the population complexity, there are three main factors: initial supply, decision-making, and model building, but the last one that denotes the population size required for the correct MPM building is dominant [29]. Thus, the population complexity is $O(n)$ [29,30]. From the analogy between the dynamics of EcGA for decomposable problems and that of UMDA, the convergence complexity for the selection methods whose intensity is constant, e.g., tournament and truncation selection, is bounded by $O(\sqrt{n})$ [2,29]. Thus, the overall time complexity of EcGA becomes $O(n\sqrt{n})$, i.e., $O(n^{1.5})$.

To bring together the strengths of the local and global searches, EcPSO ingeniously combines BPSO with EcGA. The use of BPSO is very effective in traversing the search space. But it does not nearly affect the population complexity since discovering the problem regularities mainly depends on not the search progress but the MPM building process [29]. Thus, the population

[4] Without loss of generality, the task is to maximize the fitness function.
[5] It is assumed that the size of a tournament is s.

complexity can be approximated to that of EcGA, that is, $O(n)$. Moreover, the number of generations until convergence can be reduced by employing BPSO. However, it is not able to reduce the computational overhead accruing from the growth of the problem size since the effect of the local search is not strong enough to come up with the growth of the search space complexity [28]. Approximately, EcPSO has the same convergence complexity as EcGA, that is, $O(\sqrt{n})$. As a result, the overall time complexity of EcPSO amounts to $O(n\sqrt{n})$, i.e., $O(n^{1.5})$. It is interesting to note that EcPSO can reduce the running/search time itself without degrading the overall time complexity (i.e., scalability) of EcGA.

As for mBPSO, there is no process for discovering the problem regularities. Obviously, the population complexity is exponential [29], which leads to the exponential complexity in the convergence time. Thus, the overall time complexity of mBPSO surely becomes exponential.[6]

The above approach is somewhat conservative, but the empirical evidence in Section 4.2 supports the results. It is worthwhile to investigate the time complexity (of the algorithms) of other types of problems, but further study is beyond the scope of this paper.

4. Experiments and results

This section empirically demonstrates the effectiveness of EcPSO in conquering hard problems.

4.1. Test problems

In this experiment, two kinds of test problems are taken into account: (fully) deceptive problems [7,14,26] and Ising Spin Glass (ISG) problems [15,28]. They are regarded as the representative hard-type artificial and real-world problems, respectively.

A deceptive problem consists of additive trap functions [1,7,26]. First, we define a trap function, f_{trap}, which is a constituent of a deceptive problem, as follows:

$$f_{trap}(u(y_1,\ldots,y_k)) = \begin{cases} 1.0 & \text{if } u = k, \\ (1.0-\delta)\left(1-\frac{u}{k-1}\right) & \text{if } u < k, \end{cases} \tag{11}$$

where $u(\cdot)$ is the unitation (i.e., the number of 1s) of a k-bit long substring, and δ is the fitness difference between the best BB (i.e., all 1s) and its deceptive attractor (i.e., all 0s). The function becomes harder as k increases and δ decreases. Based on the trap function, a family of deceptive problems can be constructed by additively concatenating a number of the trap functions. Accordingly, the overall fitness amounts to the sum of all trap function values, formulated by

$$F_{k\text{-}bit}(y_1,\ldots,y_{m*k}) = \sum_{i=1}^{m} f_{trap}(y_{(i-1)*k+1},\ldots,y_{i*k}). \tag{12}$$

The task is to maximize the problem. It has one global optimum in the string of all 1s and $(2^{n/k}-1)$ local optima. In order to solve this type of problem, all bits of each sub-problem (i.e., group) must be treated together because their lower order statistics are misleading. It denotes that no algorithm finds it easy to discover the global optimum without incorporating the knowledge of BBs (i.e., linkage information). Thus, deceptive problems have been perceived as common test benchmarks in the community of evolutionary optimization.

ISG is a widely-known problem in the field of statistical physics [15,28]. The aim is to find the ground states of ISG systems. A finite-dimensional ISG is typically modeled by a regular two- or three-dimensional grid, where each node i denotes a spin σ_i and each edge (i,j) represents a coupling between two spins σ_i and σ_j. In this experiment, two-dimensional ISG systems are investigated with periodic boundary conditions. Being arranged on a two-dimensional grid, the spins interact with only their nearest neighbors. The state of an ISG system is defined by a Hamiltonian $\mathcal{H}$ that specifies the system energy by

$$\mathcal{H}(\sigma) = -\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} J_{ij}\sigma_i\sigma_j, \tag{13}$$

where a set of spins $\sigma = \{\sigma_0,\sigma_1,\ldots,\sigma_{n-1}\}$ taking values from $\{-1,+1\}$ represents the physical state of the spins, and J_{ij}, a real number, specifies a coupling coefficient from the ith spin (σ_i) to the jth spin (σ_j). The task is to discover the state of the spins at which the system energy is minimized (i.e., the ground state is achieved). Such ISG systems are frequently employed as test problems in the study of GEAs because the symmetry between spins and a large number of plateaus are exhibited.

4.2. Experimental setup and results

EcPSO is compared with the *modified BPSO* (mBPSO) [21] and the conventional EcGA [13,14]. The number of (fitness function) evaluations until reaching the global optimum is taken as the performance measure. Clearly, the minimum number of such evaluations (in terms of problem size) is of interest. The corresponding population (i.e., swarm) size N is empirically

[6] An exact value of the exponent is not important in the study of computational time complexity.

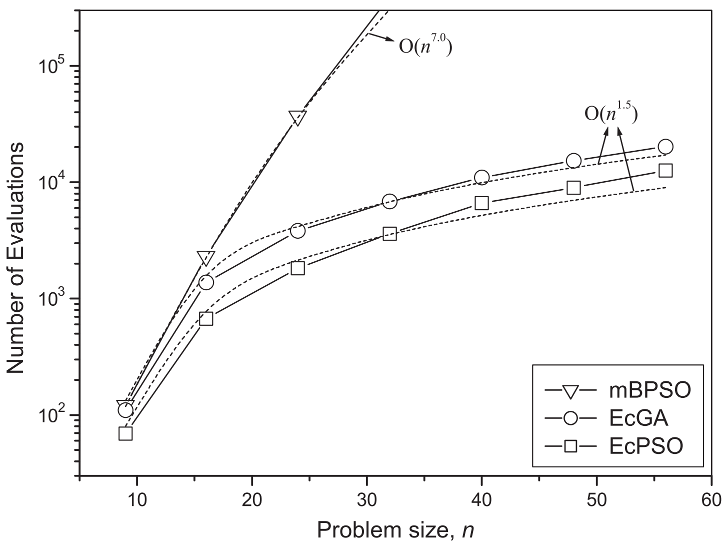

Fig. 1. Performance comparison of mBPSO, EcGA, and EcPSO in terms of the average number of evaluations until reaching the optimum, as applied to the 4-bit deceptive problem.

Table 1
Statistical comparison of the algorithms for the 4-bit deceptive problem. The numbers in non-parenthesis and parenthesis denotes *mean* and *deviation*, respectively.

n	8	16	24	32	40	48	56
mBPSO	121.24	2315.7	36983.1	393166.7	–	–	–
	(91.97)	(2321.6)	(52828.7)	(619639.9)			
EcGA	109.82	1377.1	3803.4	6831.4	10952.8	15352.1	20311.5
	(56.24)	(438.5)	(476.9)	(861.7)	(1452.2)	(1366.3)	(2311.4)
EcPSO	69.3	671.3	1823.1	3641.4	6592	8959.5	12637.5
	(41.78)	(236.5)	(395.6)	(643.5)	(805.6)	(1185.8)	(1633.9)
	Statistical t-test: (EcGA – EcPSO)						
t-value	5.57[a]	7.58[a]	16.45[a]	15.65[a]	13.45[a]	18.94[a]	16.46[a]

[a] Statistical significance by a paired, two-tailed test at $\alpha = 0.01$. Clearly, the statistical test for mBPSO does not need to be performed.

found by a bisection method [1,13,14,26]. Moreover, due to its effectiveness in achieving low selection noise, the tournament selection without replacement is employed in both EcGA and EcPSO. In order to provide the moderate selection intensity, the size of the tournament is set to 4 for deceptive problems and 16 for ISG systems. The parameter α of $I_\alpha(r)$ of EcPSO is fixed at 0.1 in order to incorporate the local search at a proper level. As for mBPSO, the parameter setting suggested by the original work [21] is employed: $w = 0.6893$ and $c_1 = c_2 = 1.4269$. Each experiment is terminated either when the optimum is found or when every individual returns the same quality. All results are averaged over 30 runs.

To begin with, two deceptive problems are tested here: 4-bit and 5-bit (i.e., $k = 4$ and $k = 5$). The fitness difference, δ is set to 1/4 and 1/5, respectively. Fig. 1 and Table 1 compare the number of evaluations until convergence to the optimum when mBPSO, EcGA, and EcPSO are applied to the 4-bit deceptive problem. Also, the performance comparison for the 5-bit deceptive problem is exhibited in Fig. 2 and Table 2. The results demonstrate that EcPSO achieves a significant improvement in comparison with mBPSO and EcGA. Further, it is seen that the theoretical models of the time complexity of the algorithms are consistent with the experimental results.

As for mBPSO, the number of evaluations required for discovering the optimum exponentially grows as the problem size increases, that is, the exponential time complexity, $O(n^{7.0})$ and $O(n^{8.0})$. Moreover, EcGA and EcPSO can solve the deceptive problems with a quadratic time complexity in terms of the problem size, i.e., $O(n^{1.5})$.[7] In the running/search time, more specifically, EcPSO converges to the optimum with, on average, 55% (for 4-bit deception) and 64% (for 5-bit deception) of the number of evaluations of EcGA (see the tables). The good news for EcPSO is that its performance improvement grows as the problem's difficulty increases. Tables 1 and 2 support the claim of the dominant performance of EcPSO over mBPSO and EcGA. As a result, it follows that EcPSO traverses the search space of difficult problems more efficiently than mBPSO and EcGA.

[7] In [14], a more accurate complexity has been empirically found as $O(n^{1.5} log(n))$.

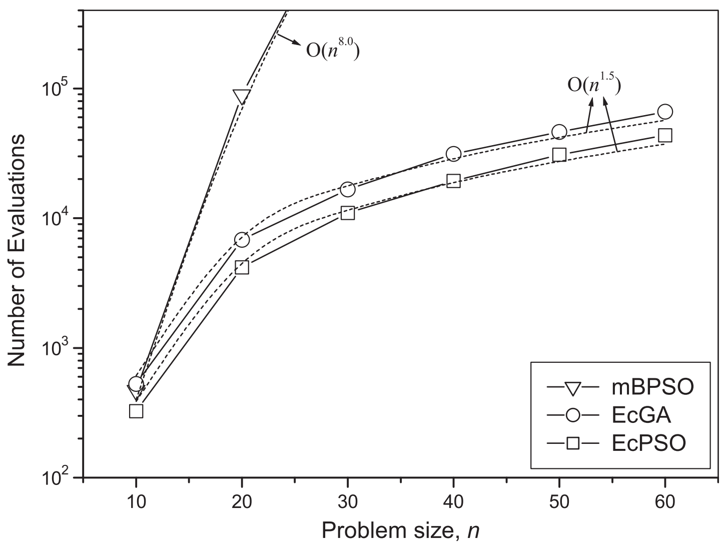

Fig. 2. Performance comparison of mBPSO, EcGA, and EcPSO in terms of the average number of evaluations until reaching the optimum, as applied to the 5-bit deceptive problem.

Table 2
Statistical comparison of the algorithms for the 5-bit deceptive problem. The numbers in non-parenthesis and parenthesis denote *mean* and *deviation*, respectively.

n	10	20	30	40	50	60
mBPSO	469.8 (332.8)	89371.6 (161191.6)	3.1571E6 (3.9180E6)	–	–	–
EcGA	527.47 (241.05)	6816.4 (2033.9)	16658.3 (2209.4)	31233.3 (3067.4)	46183 (4446)	66200 (6303.6)
EcPSO	325.7 (204.6)	4181.3 (549.3)	10957.5 (3738.3)	19400 (2362.2)	30791.7 (4292.7)	43525 (5064.4)
	Statistical *t*-test: (EcGA – EcPSO)					
t-value	2.94[a]	6.75[a]	6.68[a]	15.21[a]	13.71[a]	15.61[a]

[a] Statistical significance by a paired, two-tailed test at $\alpha = 0.01$. Clearly, the statistical test for mBPSO does not need to be performed.

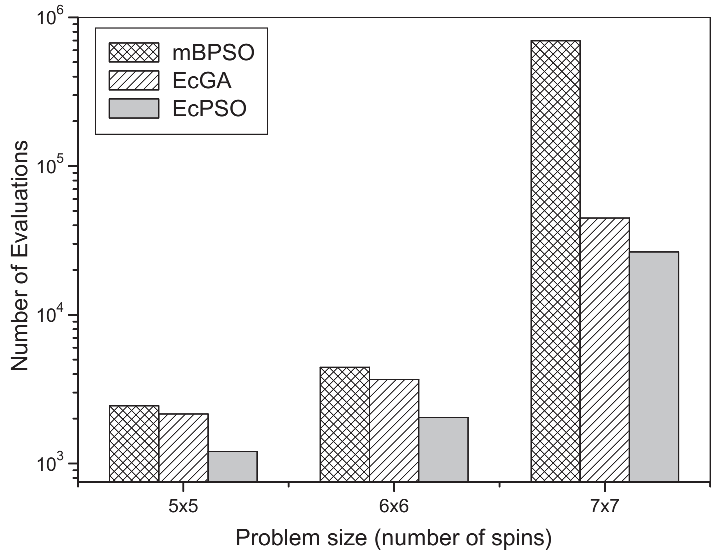

Fig. 3. Performance comparison of mBPSO, EcGA, and EcPSO in terms of the average number of evaluations until reaching the optimum, as applied to the ISG systems.

Table 3
Statistical comparison of the algorithms for the ISG systems. The numbers in non-parenthesis and parenthesis denote *mean* and *deviation*, respectively.

n	25	36	49
mBPSO	2443.9	4447.1	695583.3
	(2262.2)	(3892.9)	(861064.1)
EcGA	2150.3	3666.2	44725
	(992.4)	(1671.8)	(20289.9)
EcPSO	1201.2	2033.5	26480.4
	(662.6)	(1155.4)	(27056.3)
	Statistical t-test: (EcGA − EcPSO)		
t-value	4.69[a]	5.22[a]	4.83[a]

[a] Statistical significance by a paired, two-tailed test at $\alpha = 0.01$. Clearly, the statistical test for mBPSO does not need to be performed.

At this juncture, ISG systems with various spins are tested for demonstrating the feasibility of EcPSO in solving real-world problems. With a view of getting a quantitative comprehension of the disorder in ISG systems, we consider the ±J spin glass whose value (i.e., coupling coefficient) is chosen from $\{-1,+1\}$ with an equal probability. Fig. 3 and Table 3 exhibit the performance comparison of mBPSO, EcGA, and EcPSO, as applied to ISG systems with $\{25, 36, 49\}$ spins arranged on 5×5, 6×6, and 7×7 toroids, respectively. The ground states (i.e., optimal solutions) of ISG systems are obtained from an online server [17]. Trends similar to those found in Figs. 1 and 2 can be observed here, as well; therefore, it is proved that EcPSO outperforms mBPSO and EcGA in finding the ground states of ISG systems. Also, the efficiency of EcPSO over mBPSO and EcGA is clearly supported by the statistical test in Table 3.

Last but not least, it is observed that the scalability performance (in terms of the problem size) of EcPSO is comparable to that of EcGA. In other words, the time complexity of the two algorithms is asymptotically identical. This implies that the invitation of PSO mechanisms not only enhances the convergence performance, but also does not affect the unique features of EcGA (i.e., the problem decomposition and the maximum inter-mixing of BBs).

5. Conclusion

This paper has presented an efficient framework of the estimation of particle swarm distribution algorithms (EPSDAs). The aim was to bring together the strengths of the particle swarm optimization (PSO) and the estimation of distribution algorithms (EDAs). This was achieved by discovering problem regularities from the selected set of local best solutions and then generating new individuals by means of the probabilistic model sampling technique of EDAs and the particle evolving process of PSO. Moreover, an extended compact particle swarm optimization (EcPSO) has been developed by referring to the framework of EPSDAs. The purpose was to demonstrate the utility of the suggested framework. The algorithm was tested on (artificial) deceptive problems, as well as (real-world) Ising Spin Glass systems. The experimental results exhibited that EcPSO achieves significantly better performance than those obtained by mBPSO and EcGA. Specifically, EcPSO discovered the optimal solution by the least number of evaluations, without compromising the scale-up behavior in terms of the problem size (i.e., time complexity). Of course, more work to develop alternatives for dealing with non-disjoint or continuous problems is needed. However, the strategy of using EPSDAs to harmonize the local search of PSO with the global search of EDAs on the basis of problem regularities has definite advantages over other algorithms. Thus, the approach proposed herein can be considered an effective tool for solving a variety of difficult, especially deceptive, and/or symmetric problems.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education, Science and Technology (MEST) (No. 2010-0015520).

References

[1] C.W. Ahn, Advances in Evolutionary Algorithms: Theory, Design and Practice, Springer, 2006.
[2] C.W. Ahn, R.S. Ramakrishna, On the scalability of real-coded Bayesian optimization algorithm, IEEE Transactions on Evolutionary Computation 12 (3) (2008) 307–322.
[3] S. Baluja, Population-Base Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.
[4] F. van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (8) (2006) 937–971.
[5] P.A.D. de Castro, F.J.V. Zuben, BAIS: a Bayesian artificial immune system for the effective handling of building blocks, Information Sciences 179 (10) (2009) 1426–1440.
[6] M. Clerc, Particle Swarm Optimization, Wiley-ISTE, 2006.
[7] K. Deb, D.E. Goldberg, Analyzing deception in trap functions, in: Foundations of Genetic Algorithms, 1993, pp. 93–108.
[8] W. Dong, X. Yao, Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms, Information Sciences 178 (15) (2008) 3000–3023.

[9] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, second ed., Addison-Wesley, Reading, MA, 2007.
[11] D.E. Goldberg, K. Sastry, Genetic Algorithms: The Design of Innovation, second ed., Springer, 2008.
[12] G. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 287–297.
[13] G. Harik, Linkage Learning via Probabilistic Modeling in the ECGA. IlliGAL Technical Report No. 99010, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
[14] G.R. Harik, F.G. Lobo, K. Sastry, Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA), Studies in Computational Intelligence 33 (2006) 39–61.
[15] C.V. Hoywegen, Detecting spin-flip symmetry in optimization problems, in: Theoretical Aspects of Evolutionary Computing (Natural Computing Series), Springer-Verlag, 2001, pp. 423–437.
[16] M. Iqbal, M. Montes de Oca, An estimation of distribution particle swarm optimization algorithm, in: Proceedings of ANTS'06, LNCS, vol. 4150, 2006, pp. 72–83.
[17] M. Jünger, <http://www.informatik.uni-koeln.de/ls_juenger/projects/sgs.html>.
[18] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[19] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of the World Multi-conference on Systemics, Cybernetics and Informatics, 1997, pp. 4104–4109.
[20] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.
[21] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Progress in Natural Science 18 (2008) 1161–1166.
[22] H. Mühlenbein, G. Paaß, From recombination for genes to the estimation of distributions. I. Binary parameters, in: Parallel Problem Solving from Nature – PPSN IV, Lecture Notes in Computer Science, vol. 1141, 1996, pp. 178–187.
[23] H. Mühlenbein, T. Mahnig, FDA – a scalable evolutionary algorithm for the optimization of additively decomposed function, Evolutionary Computation 7 (4) (1999) 353–376.
[24] M. Munetomo, N. Murao, K. Akama, Introducing assignment functions to Bayesian optimization algorithms, Information Sciences 178 (1) (2008) 152–163.
[25] K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1 (2–3) (2002) 235–306.
[26] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA – the Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kauffman, 1999, pp. 525–532.
[27] M. Pelikan, D.E. Goldberg, F. Lobo, A survey of optimization by building and using probabilistic models, Computational Optimization and Applications 21 (1) (2002) 5–20.
[28] M. Pelikan, D.E. Goldberg, Hierarchical BOA solves Ising spin glasses and MAXSAT, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03), LNCS, vol. 2724, 2003, pp. 1271–1282.
[29] M. Pelikan, Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms, StudFuzz 170, Springer-Verlag, 2005.
[30] K. Sastry, D.E. Goldberg, On Extended Compact Genetic Algorithm. IlliGAL Technical Report No. 2000026, University of Illinois at Urbana-Champaign, Urbana, IL, 2000.
[31] J. Sun, Q. Shang, E. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, Information Sciences 169 (3–4) (2005) 249–262.
[32] L. Tan, D. Taniar, Adaptive estimated maximum-entropy distribution model, Information Sciences 177 (15) (2007) 3110–3128.
[33] K. Tang, X. Yao, Editorial: Special Issue on "Nature Inspired Problem-Solving", Information Sciences 178 (15) (2008) 2983–2984.
[34] J. Wang, Genetic particle swarm optimization based on estimation of distribution, in: Proceedings of LSMS'07, LNCS, vol. 4688, 2007, pp. 287–296.
[35] J. Wang, A novel discrete particle swarm optimization based on estimation of distribution, in: Proceedings of ICIC'07, LNAI, vol. 4682, 2007, pp. 791–802.
[36] Y. Zhou, J. Wang, J. Yin, A discrete estimation of distribution particle swarm optimization for combinatorial optimization problems, in: Proceedings of International Conference on Natural Computation, 2007, pp. 80–84.