Open Mathematics

Research Article

Jingyun Wang and Sanyang Liu*

Learning Bayesian networks based on bi-velocity discrete particle swarm optimization with mutation operator

https://doi.org/10.1515/math-2018-0086
Received May 21, 2017; accepted June 13, 2018.

Abstract: The problem of structures learning in Bayesian networks is to discover a directed acyclic graph that in some sense is the best representation of the given database. Score-based learning algorithm is one of the important structure learning methods used to construct the Bayesian networks. These algorithms are implemented by using some heuristic search strategies to maximize the score of each candidate Bayesian network. In this paper, a bi-velocity discrete particle swarm optimization with mutation operator algorithm is proposed to learn Bayesian networks. The mutation strategy in proposed algorithm can efficiently prevent premature convergence and enhance the exploration capability of the population. We test the proposed algorithm on databases sampled from three well-known benchmark networks, and compare with other algorithms. The experimental results demonstrate the superiority of the proposed algorithm in learning Bayesian networks.

Keywords: Bayesian networks, Structure learning, Particle swarm optimization, Mutation operator

MSC: 68R10, 68T20, 68W25

1 Introduction

Bayesian networks (BNs) are probabilistic graphical models used for representing the probabilistic relationships among the random variables in the domain and doing probabilistic inference with these variables. They have been successfully used for modeling and reasoning, with applications such as pattern recognition [1, 2], medical diagnosis [3, 4], risk analysis [5, 6], computational biology [7–9], and many others.

The issue of learning BN structures from data is receiving increasing attention. Algorithms for BN structures learning can be grouped into two categories. Constraint-based algorithms [10–13] construct a graph from data by employing conditional independence tests; statistical or information theoretic measures are used to test conditional independencce between the variables. Local structure learning algorithms, designed by using the knowledge of the Markov blankets of the variables, can reduce the number of dependence tests to some extent. However, these algorithms depend on the accuracy of the statistical tests, they may perform badly with the insufficient or noisy data. Score-based learning algorithms [14–16] try to construct a network by maximizing the score function of each candidate network using some greedy search or heuristic search algorithms. However, the space of all possible structures increases rapidly with the increasing number of variables [17], deterministic search method may fail to find optimal solution and are often trapped in local optimum.

Jingyun Wang: School of Mathematics and Statistics, Xidian University, Xi'an, Shannxi 710126, China
***Corresponding Author: Sanyang Liu:** School of Mathematics and Statistics, Xidian University, Xi'an, Shannxi 710126, China, E-mail: liusanyang@126.com

On the other hand, approximation or nondeterministic algorithms are often promising to solve the problem of BN structures learning[18, 19]. To overcome the drawbacks of the score-based algorithms, swarm intelligence algorithms have been used to learn BN structures [20]. Recently, several swarm intelligence algorithms have been successfully applied to BN structures learning, such as ant colony optimization algorithm (ACO) [21–23], artificial bee colony algorithm (ABC) [24], bacterial foraging optimization (BFO) [25] and particle swarm optimization (PSO) [26–29].

Although, these swarm intelligence algorithms have good performance in the problem of BN structures learning, there may exist some unavoidable drawbacks. For instance, in global optimization problems, the challenge for PSO is that it may be trapped in the local optimum due to its poor exploration. To enhance the exploration ability of PSO, many strategies have been proposed, so that it can be widely used in many research and application areas with its advantages not only in easy implementation and few parameters to adjust, but also in the ability of quick discovery of optimal solutions. BN structures learning is one of this cases. The classical PSO is designed for continuous problems. In order to extent its applications, and apply it in discrete space to learn BNs, several discrete PSOs for discrete optimization have been presented.

To keep the efficiency of the classical PSO in continuous space, the fast convergence speed and global search advantages, the bi-velocity discrete PSO was proposed and applied in the steiner tree problem in graphs and the multicast routing problem in communication networks [30, 31]. Since a BN structure is represented as a connectivity matrix, in which each element is 0 or 1, and the particle corresponding to the BN structure can be represented by a binary string. Thus, we adopt the velocity and position updating rules similar to that proposed in [30, 31]. However, in PSO each particle moves toward its past best position and the global best position found so far, the exploitation ability is enhanced, but the number of nonzero elements of the velocity tends to zero with the increasing iterations. In this case, if the current global best position is not the global optimum, the particles in the swarm may be trapped in the local optima. To prevent the algorithm from being trapped in a local optimum and enhance the exploration capability, a mutation strategy is introduced to conduct mutation on each new particle. In this paper, an efficient bi-velocity discrete particle swarm optimization with mutation operator algorithm is designed to solve the problem of BN structures learning (BVD-MPSO-BN).

2 Bayesian networks and *k*2 metric

Bayesian networks are knowledge representation tools capable of representing independence and dependence relationships among variables. A Bayesian network, on one hand, is a directed acyclic graph (DAG) $G = (X, E)$, where $X = \{X_1, X_2, \cdots, X_n\}$ the set of nodes, represents the random variables in a special domain. E is the set of edges, each edge represents the directed influence of one node on another. On the other hand, a Bayesian network uniquely encodes a joint probability distribution over the random variables. It decomposes according to the structure as

$$P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i)), \tag{1}$$

where $Pa(X_i)$ is the set of the parents of node X_i in G.

When performing the score-and-searching approach for learning BNs from data, a score metric must be specified. So far many score criteria for evaluating the learned networks have been proposed. One of the most well-known score criterion in learning BN structures has been given by Cooper and Herskovits (1992). The score function for a given structure G and training database $\mathcal{D}$ is

$$P(G, \mathcal{D}) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{2}$$

where n is the number of variables, each variable X_i has r_i possible values, q_i is the number of parent configurations of variable X_i, N_{ijk} is the number of cases in $\mathcal{D}$, where X_i takes on value k with parent configuration j and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

By using the logarithm of the above function and assuming a uniform prior for $P(G)$, the decomposable $k2$ metric can be expressed as

$$f(G, \mathcal{D}) = \log(P(G, \mathcal{D})) = \sum_{i=1}^{n} f(X_i, Pa(X_i)), \tag{3}$$

where $f(X_i, Pa(X_i))$ is the $k2$ score of node X_i and defined as

$$f(X_i, Pa(X_i)) = \sum_{j=1}^{q_i} (\log(\frac{(r_1 - 1)!}{(N_{ij+r_i-1})!}) + \sum_{k=1}^{r_i} \log(N_{ijk}!)). \tag{4}$$

3 Particle swarm optimization

Particle swarm optimization is a population based stochastic optimization technique, each particle in PSO is a potential solution, the position of a particle is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, $i = 1, 2, \cdots, N$, in which D is the dimension of the search space, N is the number of particles. Each particle has a velocity $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. When a particle updates its position, it records its past best position $pbest_i = (pbest_{i1}, pbest_{i2}, \cdots, pbest_{iD})$ and the global best position $gbest = (gbest_1, gbest_2, \cdots, gbest_D)$ found by any particle in the population. In the standard PSO, the new velocity is calculated according to its previous velocity, the distances of the current position from both its past best position and the global best position. After a particle updates its velocity via Eq.(5), it flies toward a new position according to Eq.(6). Each particle compares its current fitness value with its own past best fitness value – if it is better then a particle updates the past best position and the past best fitness value with the current position and its fitness value. The particle also compares its fitness value with global best fitness value – if it is better then a particle updates the global best position and the global best fitness value with the current position and its fitness value.

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r1_{ij}(t)[pbest_{ij}(t) - x_{ij}(t)] + c_2 r2_{ij}(t)[gbest_j(t) - x_{ij}(t)], \tag{5}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{6}$$

where ω is the inertia weight, c_1 and c_2 are positive acceleration coefficients, $r1$ and $r2$ are two independently uniformly distributed random values in the range $[0, 1]$, t is the number of iterations.

4 Learning BNs using the bi-velocity discrete PSO with mutation operator

Considering the fact that the original PSO algorithm operates only in a continuous search space, some strategies were proposed to solve the problems in a discrete search space and then applied to learn BN structures. To keep the original PSO framework, a bi-velocity discrete particle swarm optimization was proposed, and used for the steiner tree problem in graphs [30] and the multicast routing problem in communication networks [31]. In this section, we intent to use the bi-velocity discrete particle swarm optimization with mutation to solve the problem of BN structures learning.

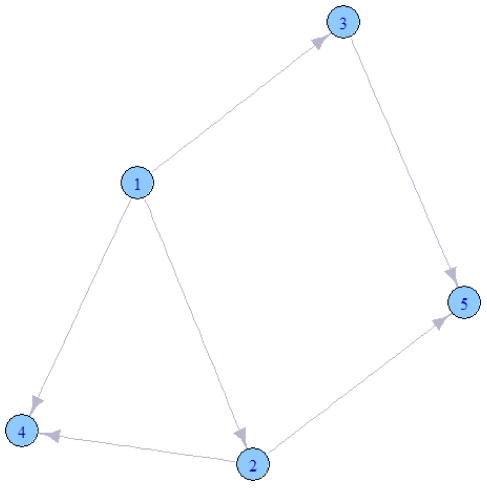
4.1 Problem representation

The problem of BN structures learning is discrete. A BN structure can be represented by an $n \times n$ connectivity matrix A, whose each element a_{ij} is defined as in Eq. (7), where n is the number of nodes. We intend to use the bi-velocity discrete particle swarm optimization to solve the BN structures learning problem. The position

of a particle is encoded as a binary string : $a_{11}, a_{12}, \cdots, a_{1n}, a_{21}, \cdots, a_{2n}, \cdots, a_{n1}, \cdots, a_{nn}$, similar to [32]. A BN structure G is shown in Fig. 1, the corresponding connectivity matrix $A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$, and the binary string representing $X = (0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

$$a_{ij} = \begin{cases} 1, & \text{if } i \text{ is a parent of } j \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

Fig. 1. An example of Bayesian network



When bi-velocity discrete PSO works in BN structures learning, the position of each particle i is represented as

$$X_i = (x_{i1}, x_{i2}, \cdots, x_{iD}), \tag{8}$$

while the velocity is encoded as

$$V_i = \begin{pmatrix} v_{i1}^0 & v_{i2}^0 & \cdots & v_{ij}^0 & \cdots & v_{iD}^0 \\ v_{i1}^1 & v_{i2}^1 & \cdots & v_{ij}^1 & \cdots & v_{iD}^1 \end{pmatrix}, \tag{9}$$

where $D = n^2$, $x_{ij} = 0$ or 1, $0 \le v_{ij}^0 \le 1$, $0 \le v_{ij}^1 \le 1$. v_{ij}^0 is the probability of x_{ij} being 0, and v_{ij}^1 is the probability of x_{ij} being 1.

4.2 Initial solution construction

To generate initial solutions, we use a method analogous to the one used in [25]. Each initial solution is derived by first starting with an empty graph dose not having any edges, and then adding the absent edges one by one to the current graph if and only if the new graph is a directed acyclic graph and the score of the new graph is higher than that of the previous graph. This procedure repeats until the number of added edges reaches the predefined value. By using the method mentioned above, a certain number of initial solutions are generated.

4.3 Bi-velocity discrete PSO with mutation operator

4.3.1 Updating rules

To keep the concept of original PSO in a continuous search space, different updating rules have been proposed in bi-velocity discrete PSO, which are described in detail as follows:

(1): $Velocity = Position1 - Position2$: Suppose that $X_1 = (x_{11}, x_{12}, \cdots, x_{1D})$ is in $Position1$ and $X_2 = (x_{21}, x_{22}, \cdots, x_{2D})$ is in $Position2$, $Position1$ is better than $Position2$, then X_2 must be learn from X_1. The jth dimension of V_i is calculated according to the difference between x_{1j} and x_{2j}, if x_{1j} is b, but x_{2j} is not (b is 0 or 1), which means that the jth dimension of X_2 is different from that of X_1, X_2 learns from X_1, so $v_{ij}^b = 1$ and $v_{ij}^{1-b} = 0$. If x_{2j} is equal to x_{1j}, which indicates that it is not necessary for X_2 to learn from X_1 on jth dimension, thus, $v_{ij}^1 = v_{ij}^0 = 0$.

For example, if $X_1 = (1,1,0,0,1,0,0,0)$ and $X_2 = (1,0,1,1,0,0,0,1)$, then, $V_i = X_1 - X_2 = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$.

(2): $Velocity = Coefficient \times Velocity$: This equation represents that the $Velocity$ is multiplied by ω or $c \times r$. Because each element of the $Velocity$ is the possibility for the position being 0 or 1, so the element that is larger than 1 is set to 1.

For example, $c \times r = (1.2, 0.8, 0.3, 1.5, 0.5, 0.2, 1.3, 0.7)$, $V = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$, then, $(c \times r) \times V = \begin{pmatrix} 0 & 0 & 0.3 & 1.5 & 0 & 0 & 0 & 0.7 \\ 0 & 0.8 & 0 & 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}$, the final velocity $V = \begin{pmatrix} 0 & 0 & 0.3 & 1 & 0 & 0 & 0 & 0.7 \\ 0 & 0.8 & 0 & 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}$.

(3): $Velocity = Velocti y1 + Velocity2$: Suppose that V_1 is $Velocti y1$ and V_2 is $Velocity2$, $V_i = V_1 + V_2$, the jth dimension v_{ij}^b in velocity V_i is the greater one between v_{1j}^b and v_{2j}^b, in which $b = 0$ or $b = 1$.

For example, $V_1 = \begin{pmatrix} 0 & 0 & 0.3 & 1 & 0 & 0 & 0 & 0.7 \\ 0 & 0.8 & 0 & 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}$, $V_2 = \begin{pmatrix} 0.1 & 0 & 0.5 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0.3 & 0 & 0.2 & 0.8 & 0 & 0 & 0.1 \end{pmatrix}$,

then, $V_i = V_1 + V_2 = \begin{pmatrix} 0.1 & 0 & 0.5 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0.8 & 0 & 0.2 & 0.8 & 0 & 0 & 0.1 \end{pmatrix}$.

In the continuous searching space, the new position of a particle is calculated by adding the updated velocity to the current position. However, the position and velocity may not be added directly in the discrete searching space. In order to solve the discrete problem, a new updating method has been proposed as Eq.(10) [31]

$$x_{ij} = \begin{cases} rand(0,1), & \text{if } v_{ij}^0 \geq \alpha, v_{ij}^1 \geq \alpha \\ 0, & \text{if } v_{ij}^0 \geq \alpha, v_{ij}^1 < \alpha \\ 1, & \text{if } v_{ij}^0 < \alpha, v_{ij}^1 \geq \alpha \\ x_{ij}, & \text{if } v_{ij}^0 \leq \alpha, v_{ij}^1 \leq \alpha \end{cases}, \tag{10}$$

in which α is a random value in $[0, 1]$.

4.3.2 Mutation operation

When PSO is implemented, each particle moves toward its past best position and the global best position found so far, the exploitation ability is enhanced. However, according to the velocity and particle position updating rules, the numbers of nonzero elements in results of $pbest_i - X_i$ and $gbest - X_i$ decrease and even becomes equal to zero with the increasing iterations. In this case, if the current global best position is not the global optimum, the particles in the swarm may be trapped in the local optima. To prevent premature convergence and enhance the exploration capability, a mutation strategy is adopted to conduct mutation on each new particle. The mutation probability depends on the problem dimension, in other words, it is determined by the number of nodes in BN structures learning problem, because a BN structure is represented by an $n \times n$ connectivity matrix whose diagonal elements are all zeros, and the position of a particle is encoded

as a binary string according to the connectivity matrix. Thus, we define the mutation probability $p = 1/(n^2 - n)$, where n is the number of nodes. The mutation operator of a particle $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ is defined as

$$x_{ij} = \begin{cases} 1 - x_{ij}, & \text{if } rand \le p \text{ and } j \ne 1 + (m-1)(n+1) \\ x_{ij}, & \text{otherwise} \end{cases}, \tag{11}$$

in which, $m = 1, \cdots, n$.

4.4 Procedure of the proposed algorithm for BN structures learning

Based on the description above, the pseudo-code of BVD-MPSO-BN is presented in Algorithm 1. It starts with the initial solutions generated by the method described in section 4.2. Each particle in the swarm is encoded as a binary string corresponding to a directed acyclic graph and evaluated using the $k2$ metric. During iteration, each particle updates its velocity and position according to the updating rules presented in subsection 4.3.1. To increase the probability of escaping from a local optimum, the mutation operator is conducted on each new particle. Because each solution should be a direct acyclic graph, the direct cycles are removed from the new particle if it is a invalid solution. In order to detect and remove the cycles, we first use the depth first search algorithm to detect all back edges, and then invert or delete them. After removing the cycles, the proposed algorithm is executed to improve the past best position of each particle and the global best position of the population. During the main loop of the algorithm, the velocities and the positions of the particles are iteratively updated until a stopping criterion is met.

5 Experimental results

In this section, we use several networks to test the behaviour of BVD-MPSO-BN. These networks are available in the software GeNie [1]. In addition, we compare the proposed algorithm with other algorithms on benchmark networks. The experiments have been executed in a personal computer with Pentium(R) Dual-Core CPU, 2.0 GB memory, and Windows 7, all the algorithms have been implemented in the Matlab language.

5.1 Databases and parameter settings of the algorithms

In our experiments, three benchmark networks are selected, namely Alarm, Asia and Credit networks. Alarm network developed for on-line monitoring of patients in intensive care unites [33] contains 37 nodes and 46 arcs; Asia network is useful in demonstrating basics concepts of Bayesian networks in diagnosis [34], it is a simple graphical model and has 8 nodes and 8 arcs; Credit network for assessing credit worthiness of an individual was developed by Gerardina Hernandez as a class homework at the University of Pittsburgh, it is available in the GeNie software and consists of 12 nodes and 12 arcs. The databases used in our experiments are sampled from these benchmark networks by probabilistic logic sampling. In Table 1, the databases, the original networks, the number of cases in each database, the number of nodes in each network, the number of arcs in each network and the $k2$ scores for the original networks are listed.

We compare BVD-MPSO-BN with other algorithms. BNC-PSO: structure learning Bayesian networks by particle swarm optimization [26], when BNC-PSO is implemented, the population size is 50, inertia weight ω decreases linearly from 0.95 to 0.4, acceleration coefficient c_1 decreases linearly from 0.82 to 0.5 and acceleration coefficient c_2 increases linearly from 0.4 to 0.83. An artificial bee colony algorithm for learning Bayesian networks (ABC-B) [24] had the following parameters: weighted coefficients for the pheromone $\alpha = 1$

[1] http://dslpitt.org/genie/.

Algorithm 1 Learning Bayesian Networks Based on Bi-Velocity Discrete PSO with Mutation Operator (BVD-MPSO-BN)

Require:

Input: Databases

Ensure:

Output: Bayesian Network

Initialization:

a. Set parameters:

$MaxIteration$: maximum number of iterations.

$Popsize$: population size.

t: iteration index.

b. Generate initial population:

Generate initial population, and calculate the fitness values according to Eq.(3).

Set $pbest_i = X_i$ $(i = 1, 2, \cdots, Popsize)$, $f(pbest_i) = f(X_i)$, find $gbest$, namely, the best position of $pbest_i$, $t = 0$.

while $t \leq MaxIteration$ **do**

  for $i = 1 : Popsize$ **do**

    Update the velocity V_i according to velocity updating rules.

    Update the position of particle X_i according to Eq.(10).

    Mutate X_i according to Eq.(11).

    if X_i is a direct cycle graph **then**

      Remove cycles.

    end if

    By Eq.(3), calculate the fitness value of particle X_i.

    if $f(X_i) > f(pbest_i)$ **then**

      $pbest_i = X_i$.

      $f(pbest_i) = f(X_i)$.

      if $f(X_i) > f(gbest)$ **then**

        $gbest = X_i$.

        $f(gbest) = f(X_i)$.

      end if

    end if

  end for

  $t = t + 1$.

end while

return $gbest$

Table 1. Databases used in experiments

Database	Original network	Number of cases	Number of nodes	Number of arcs	Score
Alarm-500	Alarm	500	37	46	-2542.9
Alarm-1000	Alarm	1000	37	46	-5044.1
Alarm-2000	Alarm	2000	37	46	-9739.4
Alarm-3000	Alarm	3000	37	46	-14512.9
Alarm-4000	Alarm	4000	37	46	-19160.6
Alarm-5000	Alarm	5000	37	46	-23780.5
Asia-500	Asia	500	8	8	-555.8
Asia-1000	Asia	1000	8	8	-1101.1
Asia-3000	Asia	3000	8	8	-3326.3
Credit-500	Credit	500	12	12	-2741.2
Credit-1000	Credit	1000	12	12	-5360.6
Credit-3000	Credit	3000	12	12	-15822.3

and the heuristic information $\beta = 2$, pheromone evaporation coefficient $\rho = 0.1$, switching parameter for exploitation versus exploration $q_0 = 0.8$, maximum number of solution stagnation $limit = 0.3$, population size is equal to 40. The parameters for BVD-MPSO-BN were chosen as: the population size is 50, inertia weight $\omega = 0.7283$, acceleration coefficients $c_1 = c_2 = 1.49618$.

5.2 Metrics of the performance

To measure the performance of proposed algorithm, we evaluate the learned results in terms of the $k2$ score and the structural difference (i.e., the differences between the learned structure and the original network). The detailed descriptions of the metrics are defined as below:

- HKS: the highest $k2$ score resulting from all trials carried out.
- LKS: the lowest $k2$ score resulting from all trials.
- AKS: the average $k2$ score (including the mean and the standard deviation) resulting from all trials.
- AEA: the average number of edges accidentally added over all trials, it contains the mean and the standard deviation.
- AED: the average number of edges accidentally deleted over all trials, it contains the mean and the standard deviation.
- AEI: the average number of edges accidentally inverted over all trials, it contains the mean and the standard deviation.
- LSD: the largest structural difference resulting from all trials.
- SSD: the smallest structural difference resulting from all trials.
- ASD: the average structural difference (including the mean and the standard deviation) resulting from all trials.
- AIt: the average number of iterations needed to find an optimal solution over all trials, it contains the mean and the standard deviation.
- AET: the average execution time over all trials.

5.3 Experimental analysis

5.3.1 Learning BNs using bi-velocity discrete PSO with mutation operator

To study the performance of BVD-MPSO-BN algorithm for Bayesian networks learning, we use it to recover the structures from databases sampled from the given benchmark networks. $k2$ score and structural difference as the basic metrics of the performance are adopted to evaluate the learned networks. We test the BVD-MPSO-

Table 2. The $k2$ score, number of iterations and running time of BVD-MPSO-BN on different networks

Database	HKS	LKS	AKS	Alt	AET
Alarm-500	-2527.9	-2554.9	-2536.6±10.1	275.6±71.7	210
Alarm-1000	-5032.2	-5054.9	-5038.4±7.1	157.8±34.5	108.3
Alarm-2000	-9721.8	-9727.2	-9723.1±1.8	201±34.4	187.8
Alarm-3000	-14510.2	-14515.0	-14511.3±1.5	182.1±57.7	192.3
Alarm-4000	-19149.9	-19154.2	-19151.4±1.6	213.2±37.7	248.9
Alarm-5000	-23769.1	-23773.2	-23770.9±1.9	208.4±43.2	243.8
Asia-500	-555.2	-555.2	-555.2±0.0	15.3±10.8	13
Asia-1000	-1100.8	-1101.4	-1100.8±0.2	13.4±10.8	11.8
Asia-3000	-3325.9	-3326.3	-3326.0±0.2	9.5±5.7	11.0
Credit-500	-2707.3	-2716.5	-2709.9±3.9	14.1±6.0	18
Credit-1000	-5333.9	-5338.7	-5335.9±2.2	14±9.6	19.2
Credit-3000	-15806.2	-15808.4	-15806.4±0.7	14.6±5.6	28.1

Table 3. The structural difference of BVD-MPSO-BN on different networks

Database	LSD	SSD	ASD	AEA	AED	AEI
Alarm-500	14	7	12.6±2.8	7.4±1.6(**5**)	2.8±0.9(**2**)	2.4±1.7 (**0**)
Alarm-1000	14	6	8.5±2.5	4.6±1.4 (**2**)	2.5±0.5 (**2**)	1.4±1.2 (**0**)
Alarm-2000	9	4	6±1.4	2.6±0.7 (**1**)	2±0 (**2**)	1.4±1.1 (**0**)
Alarm-3000	9	2	5.9±1.9	3.2±1.1 (**1**)	1±0 (**1**)	1.4±1.0 (**0**)
Alarm-4000	7	2	3.6±1.7	1.7±0.7 (**1**)	1±0 (**1**)	0.9±1.1 (**0**)
Alarm-5000	5	3	4.5±0.7	2.3±0.5 (**2**)	1±0 (**1**)	1±1.2 (**0**)
Asia-500	4	3	3.5±0.5	2.5±0.5(**2**)	1±0(**1**)	0±0 (**0**)
Asia-1000	4	1	1.9±0.9	0.1±0.3 (**0**)	1.1±0.3 (**1**)	0.7±0.5 (**0**)
Asia-3000	2	0	1.1±0.6	0±0 (**0**)	0.9±0.3 (**0**)	0.2±0.4 (**0**)
Credit-500	2	1	1.8±0.4	0±0 (**0**)	1±0 (**1**)	0.8±0.4 (**0**)
Credit-1000	2	1	1.6±0.5	0±0 (**0**)	1±0 (**1**)	0.6±0.5 (**0**)
Credit-3000	2	1	1.1±0.3	0±0 (**0**)	0±0 (**0**)	1.1±0.3 (**0**)

BN algorithm by using the Alarm network with the sample size $n = 500, 1000, 2000, 3000, 4000, 5000$, the Asia network with the sample size $n = 500, 1000, 3000$ and the Credit network with the sample size $n = 500, 1000, 3000$. Table 2 reports the experimental results in terms of $k2$ score, the number of iterations and the execution time. Table 3 shows the experimental results based on the structural difference between the learned network and the original network. Each statistic in Table 2 and Table 3 is the average and standard deviation values over ten independent runs of BVD-MPSO-BN algorithm. We mark the best values in bold.

As shown in Table 2, the difference between HKS and LKS is small on databases Alarm-2000, Alarm-3000, Alarm-4000 and Alarm-5000, except for Alarm-500 and Alarm-1000, which do not have enough samples to correctly learn Alarm networks. The algorithm also returns the small standard deviation, which indicates that the BVD-MPSO-BN algorithm is stable for the network with enough samples. For Asia network, the differences between HKS and LKS are smaller than 0.6 on database Asia-1000 and 0.4 on database Asia-3000, the algorithm returns the same $k2$ score on database Asia-500. In addition, the difference between AKS and the score of the original network is smaller than 0.6, which means that the score of the Asia network obtained by BVD-MPSO-BN algorithm is very close to that of the original network. For Credit network, the proposed algorithm obtains small standard deviation value and the difference between HKS and LKS, which indicates that the BVD-MPSO-BN algorithm also performs well in Credit network.

From the view of the structural difference, as shown in Table 3, the average and standard values in terms of ASD, AEA, AED and AEI are relatively small on databases sampled from Alarm, Asia and Credit networks. For Alarm network, the values of SSD on databases Alarm-3000 and Alarm-4000 are equal to two, which means that only two times of legitimate operations needed to change the learned network to the original one

at the best case. The standard deviation values of AED are equal to zeros on databases Alarm-2000, Alarm-3000, Alarm-4000 and Alarm-5000, which means that the proposed algorithm learns the Alarm network with one or two edges accidentally deleted over ten runs. The average structural difference on database Alarm-500 is larger than 12, which means that the algorithm has poor performance on small databases. For Asia and Credit networks, the average and standard deviation values of AEA approach to zeros, and they are equal to zeros on databases Asia-3000, Credit-500, Credit-1000 and Credit-3000, which means that there are no edges accidentally added when the proposed algorithm learns the Asia network with sample size 3000 and the Credit network with sample size 500, 1000 and 3000. In the best case, the values of AEI are equal to zeros on databases generated from the benchmark networks, that is, there is no accidentally inverted edges in the best case.

The results related to the Alarm, Asia and Credit networks demonstrate that the proposed algorithm is stable for the large networks and able to find structures very close to the original structures for small networks. The performance of the proposed algorithm improves with the increasing sample size.

5.3.2 Learning BNs using different algorithms

Next, we compare BVD-MPSO-BN algorithm with BNC-PSO and ABC-B algorithms. The experimental results are presented in Table 4, Table 5 and Table 6. Each entry is the average and standard deviation values over ten independent runs of the different algorithms. The performance of the algorithms is evaluated based on the accuracy in terms of AKS, AEA, AED, AEI and AIt. The best values for different metrics are marked in bold.

Table 4 shows the experimental results of three different algorithms on databases sampled from Alarm network. From the perspective of $k2$ score, BVD-MPSO-BN achieves the best values of AKS on databases Alarm-3000 and Alarm-5000. Although, BVD-MPSO-BN obtains the higher $k2$ score on databases Alarm-1000 compared with BNC-PSO, the standard deviation is larger than that obtained by BNC-PSO. ABC-B algorithm returns the best $k2$ score on small database Alarm-500. From the view of structural difference, the ASD values of BVD-MPSO-BN on databases Alarm-1000 and Alarm-3000 are the smallest among those of three algorithms. The ASD values returned by ABC-B algorithm on databases Alarm-500 and Alarm-5000 are smaller than that returned by other algorithms. Although ABC-B achieves the smallest average value of ASD on database Alarm-5000, the standard deviation is larger than that of BVD-MPSO-BN. It is obvious that BNC-PSO obtains networks with more incorrect edges compared with original networks on database Alarm-500. The values of AEA returned by ABC-B on different databases sample from Alarm network are the best among those of three algorithms. BVD-MPSO-BN achieves the smallest values of AED on database Alarm-500 and Alarm-1000, the AED values of BNC-PSO and ABC-B on databases Alarm-3000 and Alarm-5000 are the same as that of BVD-MPSO-BN, and they are equal to ones, which means that each of three algorithms learns the BN structures with one edge accidentally deleted on each trial carried out. BVD-MPSO-BN obtains the best values of AEI on databases Alarm-1000, Alarm-500 and Alarm-5000. From the view of the number of iterations, ABC-B often needs less number of iterations compared with BVD-MPSO-BN on databases Alarm-500, Alarm-3000 and Alarm-5000.

From the experimental results on Asia network with sample size $n = 500, 1000, 3000$ in Table 5, we observe that the BVD-MPSO-BN and BNC-PSO algorithms achieve the same well $k2$ score and structural difference on databases Asia-500 and Asia-1000, they learn the BNs from database Asia-3000 with no edges accidentally added over ten executions. In comparison to the use of ABC-B algorithm, BVD-MPSO-BN algorithm can obtain higher $k2$ score, while the ASD values returned by ABC-B algorithm on databases Asia-500 and Asia-1000 are the smallest among three algorithms. There is no accidentally inverted edges generated by BVD-MPSO-BN and BNC-PSO algorithms on database Asia-500. However, the average number of iterations of BVD-MPSO-BN algorithm is the smallest among three algorithms on databases Asia-1000 and Asia-3000.

The experimental results of three algorithms on the Credit network are presented in Table 6. For $k2$ score, BVD-MPSO-BN algorithm does not perform well on databases Credit-500 and Credit-1000, but still obtains relatively good result on databases Credit-3000. From the view of the structure difference, we observe that BVD-MPSO-BN obtains the best ASD results on databases Credit-500 and Credit-1000. BVD-MPSO-BN and

Table 4. The experimental results of three algorithms on Alarm network

Database	Algorithm	AKS	ASD	AEA	AED	AEI	AIt
Alarm-500	BVD-MPSO-BN	-2536.6±10.1	12.6±2.8	7.4±1.6	**2.8±0.9**	2.4±1.7	276.5±71.7
	BNC-PSO	-2548.6±10.0	21±2.6	12±2.3	4±0.9	5±1.6	308.2±64.6
	ABC-B	**-2532.7±4.4**	**10.1±1.6**	**4.7±0.8**	3.3±0.8	**2.1±0.7**	**272±25.6**
Alarm-1000	BVD-MPSO-BN	-5038.4±7.1	**8.5±2.5**	4.6±1.4	**2.5±0.5**	**1.4±1.3**	**157.8±34.5**
	BNC-PSO	**-5040.0±2.7**	10.4±2.2	5.6±1.0	2.7±0.5	2.1±1.7	276±45.7
	ABC-B	-5043.6±8.1	9.1±3.2	**3.4±1.5**	3.2±0.6	2.5±1.7	176.8±37.5
Alarm-3000	BVD-MPSO-BN	**-14511.3±1.5**	**5.6±1.9**	3.2±1.1	**1±0**	**1.4±1.0**	182.1±57.7
	BNC-PSO	-14516.2±5.3	8.2±1.9	4.5±1.1	**1±0**	2.7±1.1	272±55.3
	ABC-B	-14516.5±7.5	6.0±1.9	**2.0±0.9**	**1±0**	3.0±1.3	**162.4±49.6**
Alarm-5000	BVD-MPSO-BN	**-23770.9±1.9**	4.5±0.7	2.3±0.5	**1±0**	**1.2±0.6**	208.4±43.2
	BNC-PSO	-23770.9±6.5	4.5±2.6	2.3±1.2	**1±0**	1.2±1.7	208.4±59.4
	ABC-B	-23771.2±5.3	**3.7±2.5**	**2.0±1.4**	**1±0**	1.3±1.6	**153.9±42.7**

Table 5. The experimental results of three algorithms on Asia network

Database	Algorithm	AKS	ASD	AEA	AED	AEI	AIt
Asia-500	BVD-MPSO-BN	**-555.2±0.0**	3.5±0.5	2.5±0.5	**1±0**	**0±0**	15.3±10.7
	BNC-PSO	**-555.2±0.0**	3.3±0.5	2.3±0.5	**1±0**	**0±0**	18.1±6.6
	ABC-B	-555.3±0.5	**3.1±0.5**	**1.3±0.5**	**1±0**	0.8±0.4	**8.7±2.6**
Asia-1000	BVD-MPSO-BN	**-1100.8±0.2**	1.9±0.8	**0.1±0.3**	**1.1±0.3**	0.7±0.5	**13.4±10.9**
	BNC-PSO	**-1100.8±0.2**	1.9±0.8	**0.1±0.3**	**1.1±0.3**	0.7±0.5	17.3±11.4
	ABC-B	-1100.9±0.3	**1.4±0.8**	0.2±0.4	1.2±0.4	**0±0**	24.1±19.5
Asia-3000	BVD-MPSO-BN	-3325.9±0.1	**1.1±0.5**	**0±0**	**0.9±0.3**	0.2±0.4	**9.5±5.7**
	BNC-PSO	**-3325.9±0**	1.2±0.4	**0±0**	1±0	0.2±0.4	13.5±9.9
	ABC-B	-3326.0±0.5	1.3±0.9	0.2±0.6	1.1±0.3	**0±0**	25.7±13.6

BNC-PSO get the same AEA results and BVD-MPSO-BN obtains the best AED results on three databases. ABC-B obtains the best AEI results on databases Credit-1000 and Credit-5000, and BVD-MPSO-BN obtains the best AEI result on database Credit-500. The average number of iteration of BVD-MPSO-BN is smaller or at least not larger than that of the other two algorithms.

Table 6. The experimental results of three algorithms on Credit network

Database	Algorithm	AKS	ASD	AEA	AED	AEI	AIt
Credit-500	BVD-MPSO-BN	-2709.9±3.9	**1.8±0.4**	**0±0**	**1±0**	**0.8±0.4**	**14.1±6.0**
	BNC-PSO	-2708.2±2.7	1.9±0.3	**0±0**	**1±0**	0.9±0.3	17.4±6.0
	ABC-B	**-2705.32±0**	3±0	**0±0**	**1±0**	2±0	14.6±1.3
Credit-1000	BVD-MPSO-BN	-5335.9±2.2	**1.6±0.5**	**0±0**	**1±0**	0.6±0.5	**14.0±9.6**
	BNC-PSO	**-5335.8±2.1**	2.0±0.7	**0±0**	**1±0**	1±0.7	23.5±16.4
	ABC-B	-5354.9±10.7	2.0±0.6	0.1±0.3	1.9±0.6	**0±0**	25.9±10.1
Credit-3000	BVD-MPSO-BN	**-15806.4±0.7**	1.1±0.3	**0±0**	**0±0**	1.1±0.3	14.6±5.6
	BNC-PSO	-15813.0±5.7	1.1±0.7	**0±0**	0.5±0.5	0.6±0.7	26.0±11.0
	ABC-B	-15815.7±0	**1.0±0**	**0±0**	1±0	**0±0**	**14.6±3.2**

To test the time performance of the proposed algorithm, we evaluate three algorithms on Alarm network with sample size $n = 1000, 3000, 5000$, Asia network with sample size $n = 1000, 3000$ and Credit network with sample size $n = 1000, 3000$. Fig.(2) shows the average running time of three algorithms on different networks. It is obvious that the searching time of the proposed algorithm is the smallest among three algorithms. For BNC-PSO algorithm, the reason is that BVD-MPSO-BN keeps the advantage of fast convergence of classical PSO, while BNC-PSO was proposed by combining PSO with Genetic Algorithm. For ABC-B algorithm, during each iteration, each employed bee finds a new solution in its neighborhood by testing and comparing the $k2$ scores of four operators (addition, deletion, reversion and move). In addition, each onlooker determines a new solution by performing two knowledge-guided operators or four simple operators and comparing their $k2$ scores, so it is time consuming to compute the $k2$ score. Although the number of iterations of ABC-B is often less than that of BVD-MPSO-BN, ABC-B takes much time to reach the near-optimal solutions. Meanwhile, we analyze the changing of time requirement as the changing of sample size. Fig.(3) shows the average results of BVD-MPSO-BN in comparison to ABC-B and BNC-PSO algorithms on databases sampled from Alarm network. Three algorithms generally take much time on learning BNs from large databases. It is obvious that the execution time of the proposed algorithm increases slowly with the increase of the sample size, whereas ABC-B and BNC-PSO algorithms are sensitive to the sample capacity. The overall results demonstrate that BVD-MPSO-BN algorithm is superior to ABC-B and BNC-PSO algorithms in terms of execution time.

Fig. 2. Time performance on three different networks.

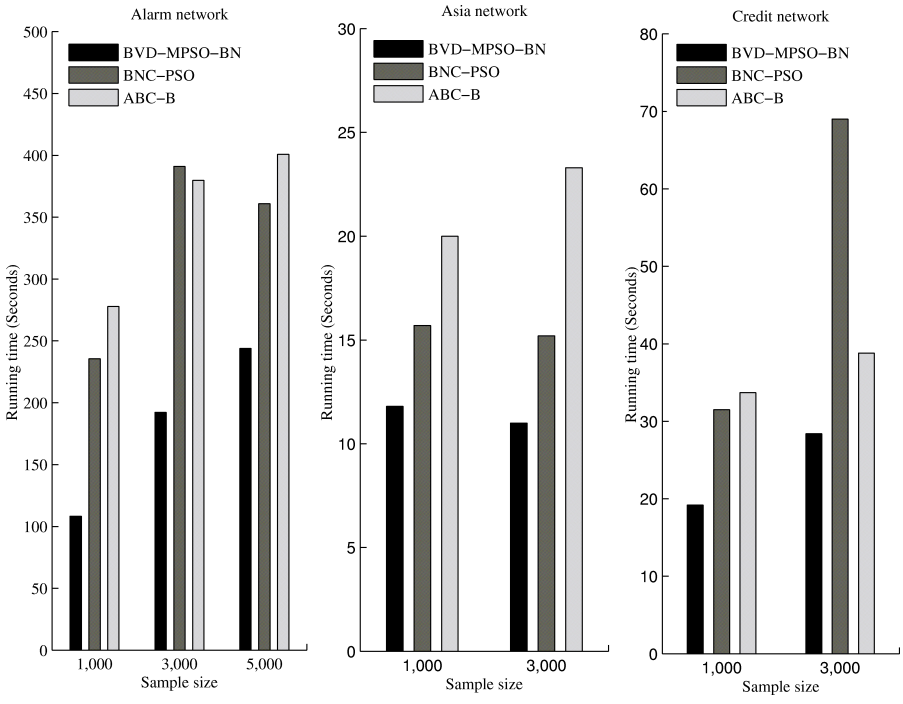


Fig. 4 shows the convergence characteristics of three heuristic algorithms on database Alarm-5000. It is obvious that the final solutions of three algorithms are close to each other. However, the proposed algorithm converges to the optimal solutions faster than both BNC-PSO and ABC-B algorithms. BNC-PSO performs better than ABC-B at the beginning because particles in PSO learn from better and best solutions so that the population quickly converges to the optimal solution. Once the particles are close to the best solution, the convergence speed becomes slower. However, with the help of mutation operator, the particles in proposed

algorithm are easy to jump out of the likely local optima, and hence the fast convergence speed could be remained through the whole evolutionary progress.

Fig. 3. Time performance of three algorithms on Alarm network.

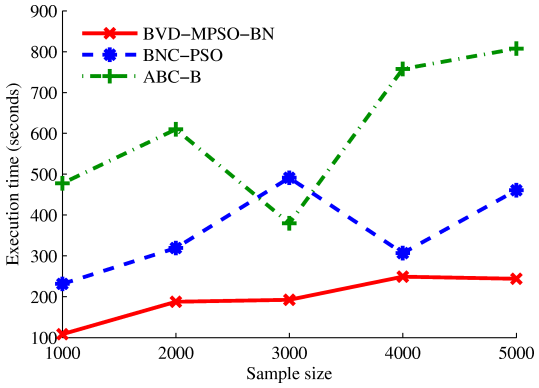
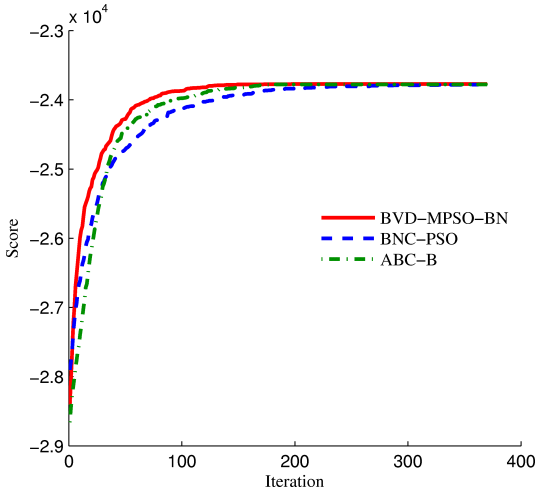


Fig. 4. The score convergence of three algorithms on Alarm-5000.



Based on the observations above, we conclude that BVD-MPSO-BN can guarantee to learn good-quality networks. BVD-MPSO-BN not only keeps the powerful searching capability in finding the optimal solution, but also prevents the particles in swarm from trapping in the local optima.

6 Conclusion

In this paper, we propose a novel score-based algorithm for BNs learning. PSO is a swarm intelligence globalized search algorithm with the advantages of simple computation and rapid convergence capability. However, with the increasing of the number of iterations, the quality of the solution can not be improved, and the algorithm converges to the local optima. In other words, it is easy for PSO to suffer from the premature convergence. To overcome the drawback of the PSO and learn BN structures from data, bi-velocity discrete PSO with mutation algorithm has been proposed. We make a proper balance between exploration and exploitation ability of the proposed algorithm. The experimental results on the databases generated from the benchmark networks demonstrate the effectiveness of our method. Comparing with the BNC-PSO algorithm for BNs learning, the advantage of our algorithm not only lies in its less computation time but also lies in its less error rate between the learned structure and the original network. In the comparison to the use of the ABC-B algorithm, when the number of samples available for structure learning is large, the proposed algorithm performs well and has the better average accurate. The experimental results illustrate the superiority of the proposed algorithm in learning BNs from the data. In this paper, the databases are completely observed, however, there may exist missing data or data with hidden variables in practice. Extending swarm-based algorithms to learn BN structures with incomplete data is our future work. In addition, the performance of the proposed algorithm decreases with the decreasing sample size. Thus, future work will consider the method for structure learning on small databases.

Acknowledgement: The research is supported by the National Natural Science Foundation of China (Grant No.61373174) and (Grant No.11401454).

References

[1] Jayech K., Mahjoub M.A., Ghanmi N., Application of bayesian networks for pattern recognition: Character recognition case, Proceedings of 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012, IEEE, pp. 748–757

[2] Wang Q., Gao X., Chen D., Pattern recognition for ship based on bayesian networks, Proceedings of Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007, vol. 4, IEEE, pp. 684–688

[3] Nikovski D., Constructing bayesian networks for medical diagnosis from incomplete and partially correct statistics, IEEE Transactions on Knowledge and Data Engineering, 2000, 12(4), 509–516

[4] AlObaidi A.T.S., Mahmood N.T., Modified full bayesian networks classifiers for medical diagnosis, Proceedings of International Conference on Advanced Computer Science Applications and Technologies (ACSAT), 2013, IEEE, pp. 5–12

[5] Bonafede C.E., Giudici P., Bayesian networks for enterprise risk assessment, Physica A: Statistical Mechanics and its Applications, 2007, 382(1), 22–28

[6] Liu Q., Pérès F., Tchangani A., Object oriented bayesian network for complex system risk assessment, IFAC-PapersOnLine, 2016, 49(28), 31–36

[7] Li Y., Ngom A., The max-min high-order dynamic bayesian network learning for identifying gene regulatory networks from time-series microarray data, IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2013, IEEE, pp. 83–90

[8] Tamada Y., Imoto S., Araki H., Nagasaki M., Print C., Charnock-Jones D.S., Miyano S., Estimating genome-wide gene networks using nonparametric bayesian network models on massively parallel computers, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2011, 8(3), 683–697

[9] Wang M., Chen Z., Cloutier S., A hybrid bayesian network learning method for constructing gene networks, Computational Biology and Chemistry, 2007, 31(5-6), 361–372

[10] Margaritis D., Learning bayesian network model structure from data(phd thesis), Tech. rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 2003

[11] Tsamardinos I., Aliferis C.F., Statnikov A.R., Statnikov E., Algorithms for large scale markov blanket discovery, Proceedings of FLAIRS Conference, 2003, vol. 2, pp. 376–380

[12] Tsamardinos I., Aliferis C.F., Statnikov A., Time and sample efficient discovery of markov blankets and direct causal relations, Proceedings of Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, ACM, pp. 673–678

[13] Pena J.M., Nilsson R., Björkegren J., Tegnér J., Towards scalable and data efficient learning of markov boundaries, International Journal of Approximate Reasoning, 2007, 45(2), 211–232

[14] Cooper G.F., Herskovits E., A bayesian method for the induction of probabilistic networks from data, Machine learning, 1992, 9(4), 309–347

[15] Alcobé J.R., Incremental hill-climbing search applied to bayesian network structure learning, Proceedings of 15th European Conference on Machine Learning, 2004, IEEE Pisa, Italy, pp. 1–10

[16] Chickering D.M., Optimal structure identification with greedy search, Journal of Machine Learning Research, 2002, 3(11), 507–554

[17] Chickering D.M., Geiger D., Heckerman D., et al., Learning bayesian networks is np-hard, Tech. rep., Citeseer, 1994

[18] Tonda A.P., Lutton E., Reuillon R., Squillero G., Wuillemin P.H., Bayesian network structure learning from limited datasets through graph evolution, Proceedings of European Conference on Genetic Programming, 2012, pp. 254–265

[19] Tonda A., Lutton E., Squillero G., Wuillemin P.H., A memetic approach to bayesian network structure learning, Lecture Notes in Computer Science, 2013, 7835, 102–111

[20] Ji J., Yang C., Liu J., Liu J., Yin B., A comparative study on swarm intelligence for structure learning of bayesian networks, Soft Computing, 2017, 21(22), 6713–6738

[21] De Campos L.M., Fernandez-Luna J.M., Gámez J.A., Puerta J.M., Ant colony optimization for learning bayesian networks, International Journal of Approximate Reasoning, 2002, 31(3), 291–311

[22] Daly R., Shen Q., et al., Learning bayesian network equivalence classes with ant colony optimization, Journal of Artificial Intelligence Research, 2009, 35(1), 391–447

[23] Jun-Zhong J., ZHANG H.X., Ren-Bing H., Chun-Nian L., A bayesian network learning algorithm based on independence test and ant colony optimization, Acta Automatica Sinica, 2009, 35(3), 281–288

[24] Ji J., Wei H., Liu C., An artificial bee colony algorithm for learning bayesian networks, Soft Computing, 2013, 17(6), 983–994

[25] Yang C., Ji J., Liu J., Liu J., Yin B., Structural learning of bayesian networks by bacterial foraging optimization, International Journal of Approximate Reasoning, 2016, 69, 147–167

[26] Gheisari S., Meybodi M.R., Bnc-pso: structure learning of bayesian networks by particle swarm optimization, Information Sciences, 2016, 348, 272–289

[27] Wang T., Yang J., A heuristic method for learning bayesian networks using discrete particle swarm optimization, Knowledge and Information Systems, 2010, 24(2), 269–281

[28] Xing-Chen H., Zheng Q., Lei T., Li-Ping S., Learning bayesian network structures with discrete particle swarm optimization algorithm, IEEE Symposium on Foundations of Computational Intelligence, 2007, IEEE, pp. 47–52
[29] Aouay S., Jamoussi S., Ayed Y.B., Particle swarm optimization based method for bayesian network structure learning, Proceedings of 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), 2013, IEEE, pp. 1–6
[30] Zhong W.L., Huang J., Zhang J., A novel particle swarm optimization for the steiner tree problem in graphs, IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008, IEEE, pp. 2460–2467
[31] Shen M., Zhan Z.H., Chen W.N., Gong Y.J., Zhang J., Li Y., Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks, IEEE Transactions on Industrial Electronics, 2014, 61(12), 7141–7151
[32] Larrañaga P., Poza M., Yurramendi Y., Murga R.H., Kuijpers C.M.H., Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters, IEEE Transactions on Ppattern Analysis and Machine Intelligence, 1996, 18(9), 912–926
[33] Beinlich I.A., Suermondt H.J., Chavez R.M., Cooper G.F., The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, AIME 89, Springer, 1989, pp. 247–256
[34] Lauritzen S.L., Spiegelhalter D.J., Local computations with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society. Series B (Methodological), 1988, 157–224