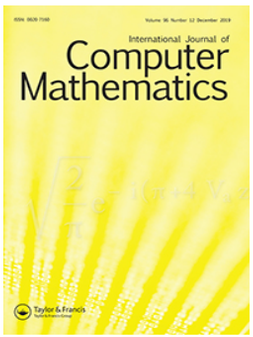

International Journal of Computer Mathematics

ISSN: 0020-7160 (Print) 1029-0265 (Online) Journal homepage: https://www.tandfonline.com/loi/gcom20

A novel discrete particle swarm optimization algorithm for solving bayesian network structures learning problem

Jingyun Wang & Sanyang Liu

To cite this article: Jingyun Wang & Sanyang Liu (2019) A novel discrete particle swarm optimization algorithm for solving bayesian network structures learning problem, International Journal of Computer Mathematics, 96:12, 2423-2440, DOI: 10.1080/00207160.2019.1566535

To link to this article: https://doi.org/10.1080/00207160.2019.1566535

INTERNATIONAL JOURNAL OF COMPUTER MATHEMATICS
2019, VOL. 96, NO. 12, 2423–2440
https://doi.org/10.1080/00207160.2019.1566535

Taylor & Francis
Taylor & Francis Group

ORIGINAL ARTICLE

A novel discrete particle swarm optimization algorithm for solving bayesian network structures learning problem

Jingyun Wang and Sanyang Liu

School of Mathematics and Statistics, Xidian University, Xi'an, People's Republic of China

ABSTRACT

Bayesian network is an effective representation tool to describe the uncertainty of the knowledge in artificial intelligence. One important method to learning Bayesian network from data is to employ a search procedure to explore the space of networks and a scoring metric to evaluate each candidate structure. In this paper, a novel discrete particle swarm optimization algorithm has been designed to solve the problem of Bayesian network structures learning. The proposed algorithm not only maintains the search advantages of the classical particle swarm optimization but also matches the characteristics of Bayesian networks. Meanwhile, mutation and neighbor searching operators have been used to overcome the drawback of premature convergence and balance the exploration and exploitation abilities of the particle swarm optimization. The experimental results on benchmark networks illustrate the feasibility and effectiveness of the proposed algorithm, and the comparative experiments indicate that our algorithm is highly competitive compared to other algorithms.

ARTICLE HISTORY
Received 19 October 2017
Revised 1 November 2018
Accepted 30 December 2018

KEYWORDS
Bayesian networks; structure learning; particle swarm optimization; heuristic algorithm

2010 MATHEMATICS SUBJECT CLASSIFICATION
68T20

1. Introduction

Bayesian networks (BNs), as one of the most widely used classes of probabilistic graphical models, are powerful formalism for representing and reasoning under uncertainty. They combine graph and probability theories to obtain a more comprehensible representation of the joint probability distribution. The structure of a Bayesian network is a directed acyclic graph (DAG), which consists of a set of nodes representing the random variables and edges demonstrating the conditional independence relations among variables. Due to their powerful abilities in representing and reasoning, Bayesian networks have been successfully applied in a variety of research areas, such as medical diagnosis [32], image processing [39], computational biology and bioinformatics [10,25], etc.

In recent years, the problem of BN structures learning from data is receiving much attention with the extensive application of Bayesian networks. Generally, the methods for learning BNs are grouped into two categories: constraint-based and score-based algorithms. Constraint-based algorithms, also known as conditional independence learners, determine causal structure by testing the independencies and dependencies among the variables. For example, learning BN structures via the grow-shrink (GS) [26], interleaved association (IAMB) [34] and max-min parents and children (MMPC) [35] algorithms. Score-based algorithms identify a BN structure with the best score by evaluating the scoring functions of the candidate network structures in the space of the BN structures. Algorithms search for the optimal BN structure by maximizing the score functions using some heuristic algorithms, such as K2 [9], and hill-climbing [1] algorithms. In general, the first approach for learning BNs lies

CONTACT Jingyun Wang wangjingyun0607@163.com School of Mathematics and Statistics, Xidian University, Xi'an 710126, People's Republic of China

on the results of statistical or information theoretic measures, it may perform badly with the insufficient or noisy data. The second approach aims at finding an optimal structure that in some sense is the best representation of the data. However, the number of candidate structures increases super-exponentially as the number of variables grows [28], learning BN structures via score-based method becomes an NP-hard problem [7]. In past decades, meta-heuristic algorithms have been developed and successfully applied in many research and application areas [3,8,16,22,23]. To efficiently solve BN structures learning problem, genetic algorithm (GA) [19], ant colony optimization (ACO) [6,17,29], artificial bee colony (ABC) [15], particle swarm optimization (PSO) [12] and bacterial foraging optimization (BFO) [38] have been used to learn BN structures.

Particle swarm optimization (PSO) is a heuristic global optimization method developed by Kennedy and Eberhart in 1995 [18], it simulates the social behaviors of birds flocks. Each particle in a swarm updates its position according to the current velocity, the personal best position and the global best position. PSO has the advantage in its ability to quickly converge to a reasonably acceptable solution. The original PSO was introduced to optimize continuous problem. Recently, discrete PSO algorithm has been designed to solve no-wait flowshop scheduling problem[27] and fuzzy job shop scheduling problem[21], and also has been used to learn BN structures. In [14,37] a BN has been represented as an adjacency list, some updating rules of velocity and position have been defined to search for the optimal structure in the space of the candidate networks. In [24] a memory binary particle swarm optimization has been proposed for BNs learning. In [36] binary encoding scheme has been introduced to represent a BN, and then, the velocity updating rules based on binary representation and the position updating rules based on stochastic mutation operation have been designed for solving the BN structure learning problem. In [2] the method based on particle swarm optimization and K2 algorithm has been proposed for BNs learning, in which PSO has been used for searching the space of ordering and then each ordering has been calculated by K2 algorithm. In [12] a BN structure has been represented as a connectivity matrix, and each particle has been encoded as a binary string. Gheisari et al. combined PSO with genetic algorithms, and used the stochastic mutation and crossover operators to define the new particle updating rules.

Although the discrete PSO has been used to solve the BN structures learning problem, the mutation and crossover operators changed the original PSO framework. In order to preserve global search and fast convergence speed advantages of the original PSO algorithm and keep the particle swarm formula unchanged, motivated by the bi-velocity discrete PSO presented in [30], we propose a novel discrete particle swarm optimization algorithm for learning Bayesian network structures. The novel discrete PSO algorithm is designed in terms of the characteristics of BNs. It is different from other existing discrete or binary PSO algorithms for learning BN structures that use sigmoid function[36], crossover and mutation operators[12] to update individuals. Each particle in the proposed algorithm is represented as a connectivity matrix that corresponds to a candidate BN structure. The velocity is encoded as a matrix, in which each element is 0,1 or -1, where 0 means that the edge between two nodes is not changed, 1 means adding a directed edge from one node to another and -1 means deleting the directed edge from one node to another. The velocity and position updating rules have been designed according to the learning mechanism of the original PSO in the continuous domain. Moreover, to overcome the premature convergence and balance the exploration and exploitation abilities, the neighbor searching operators are considered in the proposed algorithm.

In this paper, a novel discrete PSO with the neighbor searching operator is proposed for solving BN structures learning problem (NDPSO-BN), the algorithm not only maintains the search advantages of PSO but also matches the characteristics of BNs. The remainder of this paper is organized as follows. In Section 2, we give the concept of BNs and its scoring metric. The classical PSO is introduced in Section 3. In Section 4, the proposed algorithm for learning BNs is described in detail. The experimental results are presented in Section 5. Conclusions are summarized in Section 6.

2. Bayesian networks and scoring metric

2.1. Bayesian networks

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their dependent and independent relationships. The graphical structure $G = (X, E)$ of a BN is a directed acyclic graph (DAG), where nodes $X = (x_1, x_2, \ldots, x_n)$ are the random variables in the domain, edges $E = \{e_{ij}\}$ correspond to the direct influence of one node on another. If there is a directed edge from x_i to x_j, we say that x_i is a parent of x_j, the parent set of x_j is denoted by $Pa(x_j)$. The DAG defines a factorization of the joint probability distribution of $X = (x_1, x_2, \ldots, x_n)$, which can be calculated as a product of factors, each factor represents a conditional probability of the variable given its parents in the network. It can be described according to Equation (1):

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Pa(x_i)). \tag{1}$$

The learning process for constructing a BN from data is divided into two steps: learning the structure of the BN and then estimating the parameters of the local distribution functions conditional on the learned structure. In many domains, the network structures are not known in advance, so usually most of our efforts are concentrated on structures learning. In this paper, we aim at solving the problem of BN structures learning using the score-based methods.

2.2. k2 scoring metric

The score-based algorithms assign a score function to each candidate network structure and try to maximize it with the search algorithms. Several score functions are available for measuring the identified BNs.

The Bayesian score is the posterior probability of a structure G given a random sample $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ with N cases sampled from the joint distribution of X:

$$P(G|\mathcal{D}) = \frac{P(\mathcal{D}|G)P(G)}{P(\mathcal{D})} = \frac{P(G, \mathcal{D})}{P(\mathcal{D})}. \tag{2}$$

Because $P(\mathcal{D})$ does not depend on the structure, the marginal likelihood $P(\mathcal{D}|G)$ or joint probability $P(G, \mathcal{D})$, has been used as the scoring metric. The joint probability is the Bayesian Dirichlet (BD) [13]:

$$P(G, \mathcal{D}) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{3}$$

where $P(G)$ is the structure prior probability, it is a constant for each network. n is the number of variables, q_i is the number of configurations of the parents of the ith variables relative to $\mathcal{D}$, N_{ijk} is the number of instances of the ith variable being in the kth state when its parents are in their jth configuration in $\mathcal{D}$, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. $\alpha_{ijk} > 0$ is the hyper-parameters of the Dirichlet distribution, $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$. The gamma function satisfies $\Gamma(x + 1) = x\Gamma(x)$ and $\Gamma(1) = 1$. Assigning $\alpha_{ijk} = 1$, the BD metric is turned into the simple $k2$ metric, which can be expressed as Equation (4):

$$P(G, \mathcal{D}) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{4}$$

By using the logarithm of the joint probability $P(G, \mathcal{D})$, a decomposable metric can be obtained according to Equation (5), when assuming a uniform prior for $P(G)$, the constant $\log(P(G))$ is

ignored. The scoring metric is expressed as the sum of $f(x_i, Pa(x_i))$ $(i = 1, 2 \cdots, n)$, in which each term $f(x_i, Pa(x_i))$ is the $k2$ score of a variable given its parents, and it is defined as Equation (6):

$$f(G : \mathcal{D}) = \log(P(G, \mathcal{D})) \approx \sum_{i=1}^{n} f(x_i, Pa(x_i)), \tag{5}$$

$$f(x_i, Pa(x_i)) = \sum_{j=1}^{q_i} (\log(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)}) + \sum_{k=1}^{r_i} \log(N_{ijk}!)). \tag{6}$$

3. Particle swarm optimization

PSO is a population-based search algorithm for global optimization. It works by having a population of candidate solutions, called particles. Each particle is moved in the search space according to its own historical best position, known as personal best, as well as the best position found by the entire particles, termed global best. When improved positions are discovered, the personal best and global best update their positions and guide the movements of the particles. This process repeats until a predefined stopping criteria is satisfied.

Formally, optimizing a D-dimensional problem, the position of each particle is $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and the velocity is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, where $i = 1, 2, \ldots, N$ and N is the population size. The personal best of the ith particle and the global best of the swarm are represented by $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$ and $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$, respectively. X_i and V_i are initialized randomly, the jth dimension of the V_i and X_i are updated according to Equation (7) and Equation (8):

$$v_{ij} = \omega \cdot v_{ij} + c_1 \cdot r1_{ij} \cdot (pbest_{ij} - x_{ij}) + c_2 \cdot r2_{ij} \cdot (gbest_j - x_{ij}), \tag{7}$$

$$x_{ij} = x_{ij} + v_{ij}, \tag{8}$$

where ω is the inertia weight, $r1$ and $r2$ are two random values independently generated within the range of [0,1], c_1 and c_2 are the acceleration coefficients which control the influences of the cognitive and social components.

4. The novel discrete particle swarm optimization for structures learning of BN

The original particle swarm optimization is mainly for continuous problem, but the problem of BN structures learning is discrete, and the solution space of the problem is binary-structured. It's obvious that the original PSO cannot be used to learn BN structures, so the basic PSO should be modified to solve the BN structures learning problem. To design a suitable algorithm, the first work is to establish a connection between the algorithm and the given optimization problem.

4.1. Solution representation and construction

When designing the PSO, one important issue lies in its solution representation. Thus, it is required that an appropriate representation of the individuals in swarm should be selected at first. For a fixed network with n random variables, each solution is represented as an $n \times n$ connectivity matrix:

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix}, \tag{9}$$

in which x_{ij} is defined as:

$$x_{ij} = \begin{cases} 1, & \text{if there is a directed edge from node } i \text{ to node } j \\ 0, & \text{if there is no edge between node } i \text{ and node } j \end{cases}. \tag{10}$$

While the velocity is also represented as an $n \times n$ matrix:

$$V = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix}, \tag{11}$$

where v_{ij} is defined as:

$$v_{ij} = \begin{cases} 1, & \text{adding an edge from node } i \text{ to node } j \\ 0, & \text{not changing the edge between node } i \text{ and node } j \\ -1, & \text{removing an edge from node } i \text{ to node } j \end{cases}. \tag{12}$$

When meta-heuristic methods are implemented to solve the problem of BN structures learning, it is necessary to generate a certain number of initial solutions. To construct an initial solution, we start with an empty graph (i.e. the graph with no edges), and then add edges that are not in the current graph one by one to the new solution if and only if the $k2$ score of the new solution is better than the old one, and the new solution is a directed acyclic graph. This process is repeated until the maximum number of added edges is reached.

4.2. Update rules of the particles

As mentioned above, the original PSO is not suitable for solving the BN structures learning problem. Thus, Equations (7) and (8) should be modified, the new velocity and position update rules are presented in this subsection.

4.2.1. Velocity update rule

To make PSO suitable for learning BN structures and keep the velocity updating formula unchanged, the cognitive and social components are calculated still by subtracting the current position from the personal best position and global best position, respectively. The coefficient denotes the probability of each element in the velocity matrix being -1, 0 and 1. The details are given as follows.

(1): $V = X_{better} - X$. Assume that X_{better} and X are two individuals in swarm, X_{better} is better than X. The velocity matrix $V = X_{better} - X$ is updated by considering the difference between X_{better} and X. In the context of learning BN structures, if $v_{ij} = 1$, it means that there is a directed edge from node i to node j in the candidate structure represented by X_{better}, and there does not exist a directed edge from node i to node j in the BN structure represented by X. Because X_{better} is better than X, it is necessary for X to learn from X_{better}, so a directed edge from node i to node j should be added in the BN structure represented by X. If $v_{ij} = 0$, it means that the edge between node i and node j in the BN structure represented by X is same to that in BN structure represented by X_{better}. If $v_{ij} = -1$, it means that there does not exist a directed edge from node i to node j in the BN structure represented by X_{better}, but there is a directed edge from node i to node j in the BN structure represented by X. Because X learns from X_{better}, so the directed edge from node i to node j should be deleted. For

example,

$$X_{better} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, X = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

thus

$$V = X_{better} - X = \begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

That is, in candidate BN structure represented by X, directed edges from node 1 to node 2 and node 2 to node 5 need to be added, the directed edge from node 1 to node 5 should be removed.

(2): $V = (\omega \odot V^0) \oplus (C^1 \odot V^1) \oplus (C^2 \odot V^2)$. Each term of the new velocity is obtained by taking into account the 'product' of the coefficient and the velocity, in which coefficient ω is the inertia weight, and the coefficient matrix C^1 (or C^2) is the product of the acceleration constant c_1 (or c_2) and a random $n \times n$ matrix whose elements are random values on interval $[0, 1]$. V^0 is the current velocity, V^1 and V^2 are obtained by considering $X_{better} - X$ respectively. Each element in coefficient matrix is regard as the possibility for the new velocity being 1, 0 and -1, if any element of the coefficient is larger than 1, this value is set to 1. For example, suppose that

$$C^1 = c_1 \times r_1 = \begin{pmatrix} 0.7 & 1.2 & 0.2 & 0.8 & 1.3 \\ 0.2 & 1.3 & 0.6 & 0.5 & 1.1 \\ 0.1 & 0.6 & 0.6 & 0.1 & 0.8 \\ 1.1 & 1.3 & 0.3 & 0.2 & 0.5 \\ 0.9 & 1.0 & 1.1 & 0.2 & 1.1 \end{pmatrix},$$

we have

$$C^1 = \begin{pmatrix} 0.7 & 1.0 & 0.2 & 0.8 & 1.0 \\ 0.2 & 1.0 & 0.6 & 0.5 & 1.0 \\ 0.1 & 0.6 & 0.6 & 0.1 & 0.8 \\ 1.0 & 1.0 & 0.3 & 0.2 & 0.5 \\ 0.9 & 1.0 & 1.0 & 0.2 & 1.0 \end{pmatrix}.$$

Each element v_{ij} in the final velocity is calculated by comparing ω and the elements in C^1 and C^2, it is identified by Equation (13):

$$v_{ij}(t+1) = \begin{cases} v_{ij}^0, & \text{if } \max(C_{ij}^1, C_{ij}^2) < \omega \\ v_{ij}^{\text{argmax}_{k\in\{1,2\}}(C_{ij}^k)}, & \text{if } \max(C_{ij}^1, C_{ij}^2) > \omega \text{ and } C_{ij}^1 \neq C_{ij}^2 \\ random(v_{ij}^1, v_{ij}^2), & \text{if } \max(C_{ij}^1, C_{ij}^2) > \omega \text{ and } C_{ij}^1 = C_{ij}^2 \\ random(v_{ij}^0, v_{ij}^{\text{argmax}_{k\in\{1,2\}}(C_{ij}^k)}), & \text{if } \max(C_{ij}^1, C_{ij}^2) = \omega \text{ and } C_{ij}^1 \neq C_{ij}^2 \\ random(v_{ij}^0, v_{ij}^1, v_{ij}^2), & \text{if } \max(C_{ij}^1, C_{ij}^2) = \omega \text{ and } C_{ij}^1 = C_{ij}^2 \end{cases}. \quad (13)$$

For example,

$$\omega \odot V^0 = 0.7298 \odot \begin{pmatrix} 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$C^1 \odot V^1 = \begin{pmatrix} 0.7 & 1.0 & 0.2 & 0.8 & 1.0 \\ 0.2 & 1.0 & 0.6 & 0.5 & 1.0 \\ 0.1 & 0.6 & 0.6 & 0.1 & 0.8 \\ 1.0 & 1.0 & 0.3 & 0.2 & 0.5 \\ 0.9 & 1.0 & 1.0 & 0.2 & 1.0 \end{pmatrix} \odot \begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$C^2 \odot V^2 = \begin{pmatrix} 0.8 & 0.07 & 0.4 & 0.2 & 0.7 \\ 0.1 & 1.0 & 0.8 & 1.0 & 0.2 \\ 1.0 & 0.3 & 0.1 & 0.6 & 0.4 \\ 0.9 & 1.0 & 0.1 & 0.6 & 0.4 \\ 0.7 & 0.03 & 0.7 & 0.8 & 0.7 \end{pmatrix} \odot \begin{pmatrix} 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

the final velocity

$$V = \begin{pmatrix} 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

4.2.2. Position update rule

Unlike the procedure of the position updating in the continuous domain, for the discrete problem, we can not get updated position by directly adding the final velocity to the current position. So we construct the novel position update rule as Equation (14), as distinguished from previous position updating rules where the position of particle is adjusted via crossover operator or through the probability that a bit will be one or zero. In the present method, x_{ij} depends on the velocity matrix where each element represents adding, deleting or not changing an edge. To prevent the algorithm trapping into the local optimum, a mutation operator is implemented according to probability *prob*.

$$x_{ij} = \begin{cases} 1, & \text{if } v_{ij} = 1 \\ 0, & \text{if } v_{ij} = -1 \\ 1 - x_{ij}, & \text{if } v_{ij} = 0 \text{ and } r < \text{prob} \\ x_{ij}, & \text{if } v_{ij} = 0 \text{ and } r \geq prob \end{cases}, \tag{14}$$

in which r is random value on the interval [0,1], $prob = 1/(n^2 - n)$, n is the number of nodes. For example,

$$X = X + V = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

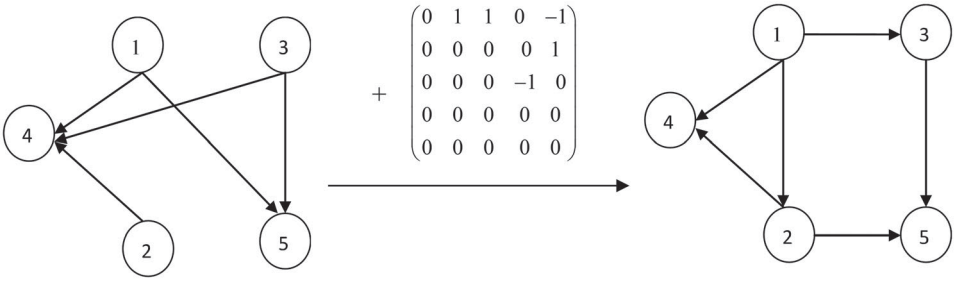

Figure 1. Position update.

then

$$X = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Figure 1 shows an example of generating a new structure.

In the process of building a BN, an invalid solution (i.e. a directed cycle graph) may be generated. To detect and remove the cycles from a BN, we detect all back edges using the depth first search algorithm [33]. After detecting all the back edges, we delete or reverse them randomly and thus remove the cycles. This operator is employed after each particle updates its position.

4.3. Neighbor searching operator

To enhance the exploitation capability of PSO algorithm and obtain a better quality of results, two simple local optimization operators are considered in the proposed algorithm. The neighbor operators consist of two groups: addition operator and deletion operator. Addition operator works by first randomly selecting two different nodes x_i and x_j, in which $x_i \notin Pa(x_j)$, then adding an edge $e_{ij} = x_i \rightarrow x_j$ if it does not generate a directed cycle in current graph G, a new solution $G = G \cup e_{ij}$ is obtained. Deletion operator is implemented by first randomly selecting an edge $e_{ij} = x_i \rightarrow x_j$ in current graph G, and then deleting it from G, namely, a new solution $G = G \setminus \{e_{ij}\}$ is obtained. During each iteration, after each particle updates its position, addition and deletion operators are performed around the current solution. For each simple operator, if the $k2$ score is better than that of the current solution, then the particle updates its position with the new solution, otherwise, it continues to stay with the current state.

4.4. Procedure of the proposed algorithm for BN structures learning

We summarize the procedure of the novel method for BN structures learning as below, and the NDPSO-BN algorithm is presented in Algorithm 1.

Step 1: Randomly generate directed acyclic graphs as the initial population.
Step 2: Calculate the $k2$ score of each candidate solution.
Step 3: Update the personal best position of each particle.
Step 4: Update the velocity and position of each particle and then generate the new particles.
Step 5: Validate the new particle if it is a directed cyclic graph.
Step 6: Recalculate the $k2$ score of each new particle.
Step 7: Employ the neighbor searching and update the particles using the greedy strategy.

Step 8: Update the global optimal solution of the population.

Step 9: Check the termination condition, if it is satisfied, output the global optimal solution, otherwise go to step 3.

Algorithm 1 NDPSO-BN algorithm

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 Equation (5).

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 Equation (13).

    Update the position of particle X_i according to Equation (14).

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

      Remove cycles.

    end if

    By Equation (5), calculate the fitness value of particle X_i.

    Addition operator: generate a new solution X_{iadd}.

    if $f(X_i) < f(X_{iadd})$ **then**

      $X_i = X_{iadd}$.

      $f(X_i) = f(X_{iadd})$.

    end if

    Deletion operator: generate a new solution $X_{idelete}$.

    if $f(X_i) < f(X_{idelete})$ **then**

      $X_i = X_{idelete}$.

      $f(X_i) = f(X_{idelete})$.

    end if

    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*

4.5. The time complexity of NDPSO-BN algorithm

The computational complexity of the proposed algorithm is dominated by the $k2$ score computation and the depth first search algorithm. The population size is N, n is the number of nodes in a BN structure. For the database with M cases, the worst case time complexity of computing $k2$ score is $O(Mn)$ [9]. In each generation, the total number of computing $k2$ score is $3N$, then the overall time complexity becomes $O(NMn)$. After generating a new individual, the depth first search algorithm is implemented to identify whether the new individual is a directed cycle graph, the time complexity of performing depth first search algorithm is bounded by $O(n^2)$ [33]. Since the total number of implementing depth first search algorithm is $2N$ in each generation, then the computational complexity is $O(Nn^2)$. Thus, the overall time complexity of NDPSO-BN algorithm is bounded by $O(NnM)$.

5. Experimental evaluation

In order to evaluate the performance of the proposed algorithm in this paper, we first test the behavior of NDPSO-BN on databases sampled from well-known benchmark networks by probabilistic logic sampling. Then, we carry out the empirical comparisons of the proposed algorithm with other algorithms.

5.1. Experimental databases and performance measures

To test the behavior of the proposed algorithm for learning BN structures, five well-known benchmark networks are selected: Alarm [4], Asia [20], Child [31], Credit and Tank networks .[1] The Alarm network, which contains 37 nodes and 46 edges, is a network often used in the medical domain for on-line monitoring of patients in intensive care units. The Asia network is a simple graphical model useful in demonstrating basic concepts of Bayesian networks in diagnosis, it includes eight nodes and eight edges. The Child network contains 20 nodes and 25 edges, the aim of the Child network is to provide a mechanism so that both clinical expertise and available data can be properly exploited to produce a reasonably transparent diagnostic aid. The Credit network is available in GeNie software and used for assessing credit worthiness of an individual, it consists of 12 nodes and 12 edges. The Tank network has 14 nodes and 20 edges and is a simple network for diagnosing possible explosion in a tank, it is developed by Gerardina Hernandez as a class homework at the University of Pittsburgh and also available in GeNie software. All of the databases used in our experiments are sampled from these networks by probabilistic logic sampling. In Table 1, we give a brief description of the experimental databases, which include the original networks, the number of nodes, the number of edges and the $k2$ score of each network.

To evaluate the performance of the algorithms, we test the experimental results in terms of the $k2$ score, the structural difference, the number of iterations and the execution time. The evaluation metrics are listed as below:

- HKS: the highest $k2$ score resulting from all trials.
- 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.
- BAE: the biggest number of edges incorrectly added over all trials.
- SAE: the smallest number of edges incorrectly added over all trials.
- AAE: the average number of edges incorrectly added over all trials, it includes the mean and the standard deviation.
- BDE: the biggest number of edges incorrectly deleted over all trials.
- SDE: the smallest number of edges incorrectly deleted over all trials.
- ADE: the average number of edges incorrectly deleted over all trials, it includes the mean and the standard deviation.

Table 1. Databases used in experiments.

Database	Original network	Number of cases	Number of nodes	Number of arcs	Score
Asia-1000	Asia	1000	8	8	−1101.11
Asia-3000	Asia	3000	8	8	−3326.34
Credit-1000	Credit	1000	12	12	−5360.58
Credit-3000	Credit	3000	12	12	−15,822.36
Credit-5000	Credit	5000	12	12	−26,374.30
Tank-1000	Tank	1000	14	20	−1632.52
Tank-3000	Tank	3000	14	20	−4848.08
Tank-5000	Tank	5000	14	20	−8138.93
Child-1000	Child	1000	20	25	−6406.11
Child-3000	Child	3000	20	25	−18,731.68
Child-5000	Child	5000	20	25	−31,157.76
Alarm-1000	Alarm	1000	37	46	−5044.07
Alarm-2000	Alarm	2000	37	46	−9739.44
Alarm-3000	Alarm	3000	37	46	−14,512.89
Alarm-4000	Alarm	4000	37	46	−19,160.06
Alarm-5000	Alarm	5000	37	46	−23,780.46

- BRE: the biggest number of edges incorrectly reversed over all trials.
- SRE: the smallest number of edges incorrectly reversed over all trials.
- ARE: the average number of edges incorrectly reversed over all trials, it includes the mean and the standard deviation.
- BSD: the biggest structural difference (edges wrongly added, deleted and reversed) resulting from all trials.
- SSD: the smallest structural difference resulting from all trials.
- ASD: the average structural difference resulting from all trials, it includes the mean and the standard deviation.
- BIt: the biggest number of iterations needed to find a near-optimal network over all trials.
- SIt: the smallest number of iterations needed to find a near-optimal network over all trials.
- AIt: the average number of iterations needed to find a near-optimal network overall trials, it includes the mean and the standard deviation.
- AET: the average execution time over all trials.

5.2. Algorithms and parameter settings

We compare the proposed method with other algorithms. An artificial bee colony algorithm for learning Bayesian networks (ABC-B) [15] proposed by Ji et al., they used the extended ABC meta-heuristic to guide the search process to construct BN structures; BNC-PSO: Structure learning of Bayesian networks by particle swarm optimization [12], S.Gheisart et al. solved the problem of BN structures learning using the PSO algorithm, in which mutation and stochastic crossover operations are defined to design the position and velocity updating rules; The sparse candidate algorithm for learning Bayesian network structure from massive datasets (SCA) [11] was introduced for learning BNs by restricting the search space; The max-min hill-climbing (MMHC) algorithm [5] was designed by first using the conditional independence tests to identify the skeleton of the network and then using the greedy hill climbing search to orient the skeleton.

NDPSO-BN: the inertia weight $\omega = 0.7298$, the acceleration coefficients $c_1 = 1.49618, c_2 = 1.49618$, population size is 50; BNC-PSO: the inertia weight ω decreases linearly from 0.95 to 0.4, the acceleration coefficient c_1 decreases linearly from 0.82 to 0.5, the acceleration coefficient c_2 increases linearly from 0.4 to 0.83 and the population size is 50; ABC-B: weighted coefficients for the pheromone $\alpha = 1$ and the heuristic information $\beta = 2$, the parameter of the pheromone evaporation $\rho = 0.1$, the parameter that used to determine the relative importance of exploitation versus exploration $q_0 = 0.8$, the maximum number of solution stagnation $limit = 3$ and the population size

Table 2. *k*2 score, number of iterations and execution time of the proposed algorithm on different databases.

Database	HKS	LKS	AKS	Blt	Slt	Alt	AET (s)
Asia-1000	−1100.78	−1100.78	−1100.78±0.00	7	3	4.90±1.52	0.80
Asia-3000	−3325.88	−3325.88	−3325.88±0.00	7	4	5.30±1.05	1.30
Credit-1000	−5333.90	−5338.77	−5335.54±2.14	18	5	10.90±4.40	3.40
Credit-3000	−15,806.22	−15,808.37	−15,806.87±1.03	19	7	13.20±3.58	5.20
Credit-5000	−26,362.73	−26,363.18	−26,363.00±0.23	29	9	15.90±6.10	7.30
Tank-1000	−1630.60	−1631.77	−1631.13±0.42	96	28	57.60±28.92	23.40
Tank-3000	−4848.08	−4848.63	−4848.14±0.17	73	15	43.9±22.04	22.60
Tank-5000	−8138.93	−8139.47	−8139.02±0.21	63	15	33.10±15.50	44.40
Child-1000	−6379.39	−6392.12	−6382.30±3.81	39	14	27.40±8.93	35.00
Child-3000	−18,727.63	−18,727.63	−18,727.63±0.00	65	21	35.60±14.08	27.30
Child-5000	−31,157.76	−31,157.76	−31,157.76±0.00	70	18	39.10±17.92	85.00
Alarm-1000	−5032.21	−5054.92	−5040.17±6.24	112	51	89.70±19.21	165.70
Alarm-2000	−9721.84	−9745.84	−9729.18±7.29	111	75	91.70±14.55	190.00
Alarm-3000	−14,511.92	−14,518.92	−14,514.23±2.40	141	82	102.70±21.31	247.90
Alarm-4000	−19,149.88	−19,159.34	−19,152.73±2.84	128	62	94.80±22.83	261.60
Alarm-5000	−23,769.25	−23,777.61	−23,771.14±2.52	119	68	94.90±27.49	267.70

Table 3. Structural difference of the proposed algorithm on different databases.

Database	BAE	SAE	AAE	BDE	SDE	ADE	BRE	SRE	ARE	BSD	SSD	ASD
Asia-1000	0	0	0.00±0.00	1	1	1.00±0.00	1	0	0.90±0.31	2	1	1.90±0.31
Asia-3000	0	0	0.00±0.00	1	1	1.00±0.00	1	0	0.60±0.51	2	1	1.60±0.51
Credit-1000	0	0	0.00±0.00	1	1	1.00±0.00	1	0	0.70±0.48	2	1	1.70±0.48
Credit-3000	0	0	0.00±0.00	0	0	0.00±0.00	2	1	1.30±0.48	2	1	1.30±1.48
Credit-5000	1	0	0.40±0.51	0	0	0.00±0.00	1	1	1.00±0.00	2	1	1.40±0.51
Tank-1000	5	2	3.20±1.03	1	0	0.70±0.48	1	0	0.20±0.42	6	2	4.10±1.20
Tank-3000	2	0	0.20±0.63	0	0	0.00±0.00	1	0	0.10±0.32	3	0	0.30±0.95
Tank-5000	2	0	0.40±0.84	0	0	0.00±0.00	2	0	0.40±0.70	3	0	0.80±1.32
Child-1000	0	0	0.00±0.00	5	4	4.20±0.42	2	0	1.00±0.94	7	4	5.20±1.13
Child-3000	0	0	0.00±0.00	1	1	1.00±0.00	3	0	1.10±1.10	4	1	2.10±1.10
Child-5000	0	0	0.00±0.00	0	0	0.00±0.00	2	0	1.00±0.81	2	0	1.00±0.81
Alarm-1000	7	2	4.40±1.95	3	2	2.60±0.51	4	1	2.40±1.07	12	6	9.40±2.50
Alarm-2000	3	2	2.70±0.48	3	2	2.20±0.42	4	0	2.80±1.47	9	4	7.70±1.49
Alarm-3000	4	1	2.50±0.85	1	1	1.00±0.00	4	0	2.00±1.49	8	3	5.50±2.01
Alarm-4000	3	0	1.10±0.99	2	1	1.10±0.31	3	0	1.60±1.26	7	1	3.80±1.85
Alarm-5000	3	1	1.50±0.70	1	1	1.00±0.00	3	0	1.60±1.07	6	2	4.10±1.44

is 40; SCA and MMHC algorithms are implemented in the Causal Explorer system[2] and we use the default values in the software implementations for the parameters of two algorithms. The maximum number of iterations *MaxIteration* is set to 500.

5.3. Learning BNs using NDPSO-BN algorithm

To evaluate the performance of the proposed method, we employ the algorithm 10 times independently on each database. Table 2 shows the experimental results in terms of the *k*2 score, the number of iterations and the execution time. Table 3 presents the structural difference between the original network and the learned structure.

From the view of scoring metric, we observe that the differences between HKS and LKS for Asia-1000, Asia-3000, Child-3000 and Child-5000 are exactly the same and equal to 0, which means that NDPSO-BN algorithm is capable of obtaining the same *k*2 scores over ten executions on each database. In addition, the standard deviations of *k*2 scores for Alarm network are relatively larger than that for other networks, which indicates that the proposed algorithm is relatively stable for small networks. Meanwhile, we notice that the standard deviation decreases with the increasing of the number of cases. From the observations above, we conclude that the proposed algorithm performs well for

Table 4. *k*2 score comparisons for different algorithms.

Database	*k*2 Score	NDPSO-BN	BNC-PSO	ABC-B	SCA	MMHC
Asia-3000	HKS	−3325.88	−3325.88	−3325.88	−3325.87	−3325.88
	LKS	−3325.89	−3325.88	−3327.43	−3325.87	−3325.88
	AKS	−3325.88±0.00	−3325.88±0.00	−3326.04±0.49	**−3325.87±0.00**	−3325.88±0.00
Credit-3000	HKS	−15,806.22	−15,806.22	−15,815.69	−16,473.54	−15,806.22
	LKS	−15,808.37	−15,822.36	−15,815.69	−16,473.54	−15,806.22
	AKS	−15,806.87±1.03	−15,813.00±5.77	−15,815.69±0.00	−16,473.54±0.00	**−15,806.22±0.00**
Tank-3000	HKS	−4848.08	−4848.08	−4848.08	−7325.20	−4901.68
	LKS	−4848.63	−4849.74	−4850.02	−7325.20	−4901.68
	AKS	**−4848.14±0.17**	−4848.31±0.52	−4848.44±0.61	−7325.20±0.00	−4901.68±0.00
Child-3000	HKS	−18,727.63	−18,727.63	−18,727.63	−20,034.56	−18,727.63
	LKS	−18,727.63	−18,753.35	−18,727.63	−20,034.56	−18,727.63
	AKS	**−18,727.63±0.00**	−18,730.20±8.13	**−18,727.63±0.00**	−20,034.56±0.00	**−18,727.63±0.00**
Alarm-3000	HKS	−14,511.92	−14,511.59	−14,511.28	−24,686.66	−14,681.68
	LKS	−14,518.92	−14,525.96	−14,531.39	−24,686.66	−14681.68
	AKS	**−14,514.23±2.40**	−14,516.23±5.26	−14,516.54±7.08	−24,686.66±0.00	−14,681.68±0.00
Alarm-5000	HKS	−23,769.25	−23,769.12	−23,769.12	−38,389.35	−23,977.14
	LKS	−23,777.61	−23,787.67	−23,786.25	−38,389.35	−23,977.14
	AKS	**−23,771.14±2.52**	−23,775.80±6.52	−23,771.22±5.31	−38,389.35±0.00	−23,977.14±0.00

small networks, and is also able to find the near-optimal networks with high *k*2 scores for large networks with enough cases. From the perspective of the number of iterations and execution time, we observe that the proposed algorithm can successfully find the near-optimal BN structures in several iterations and spends less searching time for small networks.

Table 3 shows the experimental results in terms of the structural difference between the original network and the learned network. We observe that the AAE for Asia-1000, Asia-3000, Credit-1000, Credit-3000, Child-1000, Child-3000 and Child-5000 are equal to 0, which means that NDPSO-BN algorithm learns the Asia, Credit and Child networks on these databases with no edges accidentally added. For Tank and Alarm networks, the average number of edges incorrectly added decreases with the increase of the sample capacity, and the SAE for Tank-3000, Tank-5000 and Alarm-3000 are 0, which indicates that NDPSO-BN recovers Tank and Alarm networks on these databases with no incorrectly edges added in the best cases. The ADE for Credit-3000, Credit-5000, Tank-3000, Tank-5000 and Child-5000 are 0, which means that NDPSO-BN finds Credit, Tank and Child networks on these databases with no edges accidentally deleted. The BDE and SDE for Asia-1000, Asia-3000, Credit-1000, Child-3000, Alarm-3000 and Alarm-5000 are exactly the same and equal to 1, indicating that NDPSO-BN finds the network on each database always with one edge incorrectly deleted. The SRE for databases except Credit-3000, Credit-5000 and Alarm-1000 are 0, that is, in the best cases, NDPSO-BN learns networks on these databases with no edges incorrectly reversed. The SSD for Tank-3000, Tank-5000 and Child-5000 equal to 0, which means that NDPSO-BN is capable of finding the true structures on these databases in the best cases. The structural difference between the original network and the learned structure decreases as the sample capacity increases. The mean and standard deviations of the structural difference are relatively small for the databases generated from the benchmark networks. The results obtained above help us to show that the proposed algorithm is reliable for learning BNs.

5.4. Comparing NDPSO-BN with other algorithms

We compare the proposed algorithm with three score-based algorithms (BNC-PSO, ABC-B and SCA) and a hybrid algorithm (MMHC). Each algorithm is implemented on databases Asia-3000, Credit-3000, Tank-3000, Child-3000, Alarm-3000 and Alarm-5000, and then we summarize the experimental results in Tables 4 and 5. Each report in Tables 4 and 5 is the mean and standard

Table 5. Structural difference comparison for different algorithms.

Database	Structural difference	NDPSO-BN	BNC-PSO	ABC-B	SCA	MMHC
Asia-3000	AAE	**0.00±0.00(0)**	**0.00±0.00(0)**	0.20±0.63 **(0)**	**0.00±0.00(0)**	**0.00±0.00(0)**
	ADE	**1.00±0.00(1)**	**1.00±0.00 (1)**	1.10±0.32 **(1)**	**1.00±0.00(1)**	**1.00±0.00(1)**
	ARE	0.60±0.52 **(0)**	0.20±0.42 **(0)**	**0.00±0.00(0)**	1.00±0.00 **(1)**	1.00±0.00 **(1)**
	ASD	1.60±0.52 **(1)**	**1.20±0.42(1)**	1.30±0.95 **(1)**	2.00±0.00 **(2)**	2.00±0.00 **(2)**
Credit-3000	AAE	**0.00±0.00(0)**	**0.00±0.00(0)**	**0.00±0.00(0)**	7.00±0.00 **(7)**	**0.00±0.00(0)**
	ADE	**0.00±0.00 (0)**	0.50±0.53 **(0)**	1.00±0.00 **(1)**	3.00±0.00 **(3)**	**0.00±0.00(0)**
	ARE	1.30±0.48 **(1)**	0.60±0.70 **(0)**	**0.00±0.00(0)**	5.00±0.00 **(5)**	1.00±0.00 **(1)**
	ASD	1.30±0.48 **(1)**	1.10±0.57 **(0)**	**1.00±0.00 (1)**	15.00±0.00 **(15)**	**1.00±0.00(1)**
Tank-3000	AAE	**0.20±0.63(0)**	0.40±0.70 **(0)**	0.70±0.95 **(0)**	20.00±0.00 **(20)**	11.00±0.00 **(11)**
	ADE	**0.00±0.00(0)**	0.10±0.32 **(0)**	**0.00±0.00(0)**	13.00±0.00 **(13)**	3.00±0.00 **(3)**
	ARE	**0.10±0.32(0)**	0.30±0.48 **(0)**	1.10±0.74 **(0)**	3.00±0.00 **(3)**	5.00±0.00 **(5)**
	ASD	**0.30±0.95(0)**	0.80±1.32 **(0)**	1.80±1.66 **(0)**	36.00±0.00 **(36)**	19.00±0.00 **(19)**
Child-3000	AAE	**0.00±0.00(0)**	**0.00±0.00(0)**	**0.00±0.00(0)**	5.0±0.00**(5)**	**0.00±0.00(0)**
	ADE	**1.00±0.00(1)**	**1.00±0.00(1)**	1.10±0.32 **(1)**	9.00±0.00 **(9)**	**1.00±0.00(1)**
	ARE	1.10±1.10 **(0)**	**0.80±1.23(0)**	2.00±1.41 **(0)**	3.00±0.00 **(3)**	3.00±0.00 **(3)**
	ASD	2.10±1.10 **(1)**	**1.80±1.23 (1)**	3.10±1.37 **(1)**	17.00±0.00 **(17)**	4.00±0.00 **(4)**
Alarm-3000	AAE	2.50±0.85 **(1)**	4.50±1.08 **(3)**	**2.00±0.82 (1)**	17.0±0.00 **(17)**	2.00±0.00 **(2)**
	ADE	**1.00±0.00(1)**	**1.00±0.00(1)**	**1.00±0.00(1)**	28.00±0.00 **(28)**	**1.00±0.00(1)**
	ARE	**2.00±1.50(0)**	2.70±1.06 **(1)**	3.00±1.25 **(2)**	4.00±0.00 **(4)**	9.00±0.00 **(9)**
	ASD	**5.50±2.01(3)**	8.20±1.93 **(6)**	6.00±1.89 **(4)**	49.00±0.00 **(49)**	12.00±0.00 **(12)**
Alarm-5000	AAE	1.50±0.71 **(1)**	2.30±1.18 **(1)**	1.40±1.17 **(0)**	24.00±0.00 **(24)**	**1.00±0.00(1)**
	ADE	**1.00±0.00(1)**	**1.00±0.00(1)**	**1.00±0.00(1)**	27.00±0.00 **(27)**	2.00±0.00 **(2)**
	ARE	1.60±1.07 **(0)**	**1.20±1.66(0)**	1.30±1.64 **(0)**	6.00±0.00 **(6)**	6.00±0.00 **(6)**
	ASD	4.10±1.45 **(2)**	4.50±2.67 **(3)**	**3.70±2.50(1)**	57.00±0.00 **(57)**	9.00±0.00 **(9)**

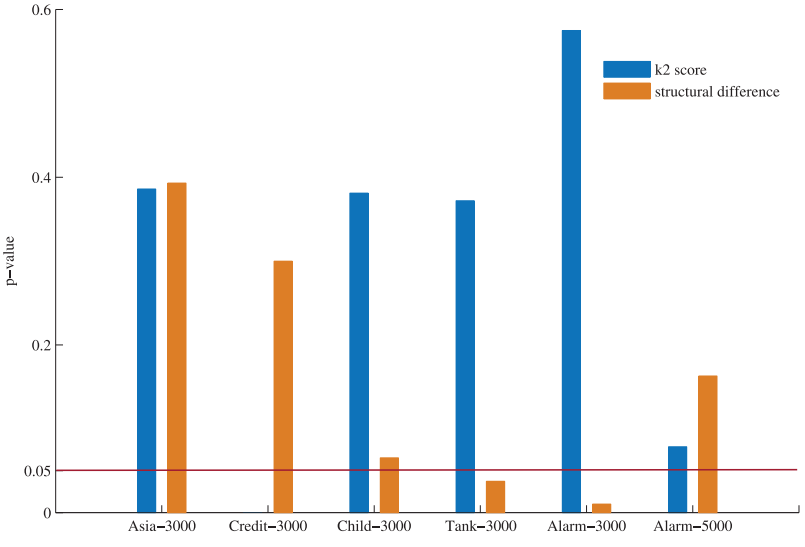

Figure 2. Analysis of variance.

deviation, the value inside (·) is the best result found along the experimentation by using the different algorithms, we also mark the best value obtained from different algorithm in bold. To illustrate the significant difference between the compared algorithms, Figure 2 presents the results of analysis of variance. Figure 3 shows the average execution time of NDPSO-BN in comparison to the BNC-PSO and ABC-B algorithms.

The *k*2 scores resulted from five algorithms on different databases are presented in Table 4. As for the score-based BN structures learning algorithms, the higher corresponding scores indicate the

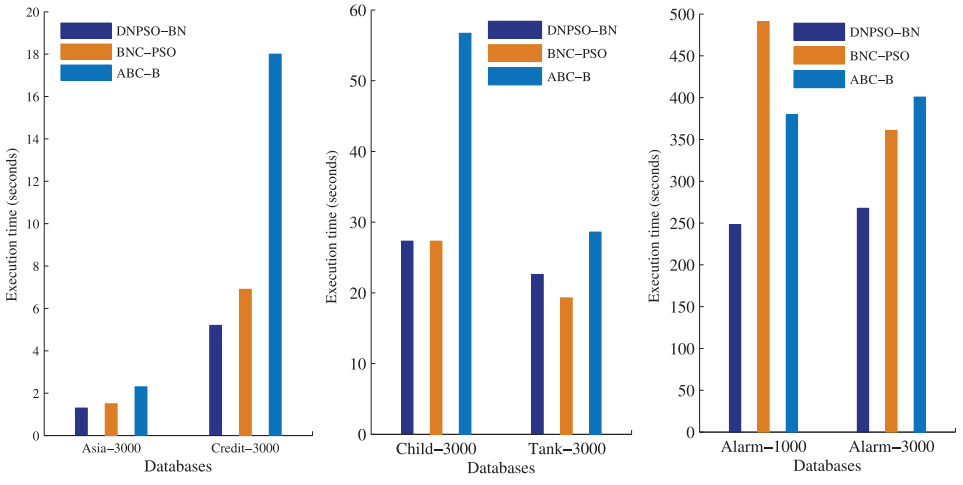

Figure 3. Time performance for three algorithms.

better quality of the learned networks. For Asia-3000, each algorithm except ABC-B always finds the BNs with the same $k2$ score over ten trials, and SCA achieves the best value (-3325.87) on the average score among five algorithms. For Credit-3000, MMHC learns the BNs with the best $k2$ score, and ABC-B always receives the same results over ten trials. For Tank-3000, Child-3000, Alarm-3000 and Alarm-5000, NDPSO-BN can find the BNs with higher or not lower scores than other algorithms. In addition, we notice that in comparison to the use of SCA and MMHC algorithms, three swarm-based algorithms (NDPSO-BN, BNC-PSO, ABC-B) can guarantee the discovery of better quality BNs especially for large networks. It is obvious that the proposed algorithm is superior to other algorithms in terms of the scoring metric.

The experimental results in terms of the structural difference are shown in Table 5, the smaller corresponding structural differences indicate the better quality of the learned networks. We observe that the swarm-based algorithms (NDPSO-BN, BNC-PSO and ABC-B) have smaller structural differences than other algorithms (SCA and MMHC) on all six databases except Credit-3000. With respect to the structural differences of the three swarm-based algorithms, mean and standard deviation values of AAE and ADE resulted from NDPSO-BN are relatively smaller for databases generated from benchmark networks. The results of ARE obtained by NDPSO-BN are relatively larger for databases Asia-3000, Credit-3000, Tank-3000, Child-3000 and Alarm-5000, which lead to the increase of structural difference of NDPSO-BN. However, in comparison to the use of two swarm-based algorithms (BNC-PSO and ABC-B), NDPSO-BN learns the BNs with lower standard deviations in the most cases, which indicates that the proposed algorithm is relatively stable, and the experimental results also demonstrate that the proposed algorithm is a competitive method in learning BN structures.

From the experimental results above, we observe that three heuristic algorithms (NDPSO-BN, BNC-PSO and ABC-B) perform well in terms of the $k2$ score and structural difference on all databases. However, SCA and MMHC algorithms, especially SCA algorithm is a failure in finding the optimal or near-optimal solution when learning large networks. Therefore, Figure 2 only shows the difference between three heuristic algorithms in learning BNs. To illustrate the difference among three heuristic algorithms, analysis of variance is used to examine the influence of each algorithm in experimental results of the $k2$ score and structural difference, the line in Figure 2 is benchmark (0.05). Based on the $k2$ score, the analysis of variance shows that the influence of NDPSO-BN, BNC-PSO and ABC-B algorithms are not significant (p-value $>$ 0.05) in all databases except Credit-3000. From the result of structural difference, we observe that three heuristic algorithms have no significant differences on databases Asia-3000, Credit-3000, Child-3000 and Alarm-5000.

To compare the time performance of the proposed algorithm, we run the NDPSO-BN, BNC-PSO and ABC-B algorithms 10 times respectively on databases Asia-3000, Credit-3000, Child-3000, Tank-3000, Alarm-1000 and Alarm-3000. Figure 3 shows the average execution time on each database. It is obvious that NDPSO-BN often runs faster than the other two algorithms, and ABC-B spends much time in finding a near-optimal solution. One reason is that in comparison to the use of BNC-PSO algorithm, the exploitation ability is enhanced when the neighbor searching strategy is performed in our proposed algorithm, it can improve the ability of quick discovery of optimal solution. Another reason is that in comparison to the use of ABC-B, the proposed algorithm keeps its advantage in ease of implementation. In addition, when ABC-B is implemented, to produce a new solution in the neighborhood of the current solution, it is necessary to carry out four simple neighbor operators and two knowledge-guided operators, which may be time-consuming.

6. Conclusion

In this paper, a novel discrete particle swarm optimization algorithm has been proposed for solving the problem of BN structures learning. To keep the search advantages of the classical PSO and match the characteristics of BNs, the new velocity and position updating rules have been designed. Meanwhile, in order to overcome the drawback of premature convergence and balance the exploration and exploitation abilities of PSO, the mutation and neighbor searching operators have been used in our proposed algorithm. We have tested the NDPSO-BN method on databases generated from well-known networks, and compared with three score-based algorithms and a hybrid algorithm. The experimental results demonstrate that our algorithm is superior in terms of the execution time and the quality of the solutions, and is an effective method for learning BN structures from data. For future research, we will continue to study the swarm-based algorithms, and solve the BN structures learning and the dynamic BN learning problems.

Notes

1. http://dslpitt.org/genie/.
2. http://www.dsl-lab.org/causal_explorer/.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The research is supported by the National Natural Science Foundation of China [grant number 61877046].

References

[1] J.R. Alcobé, Incremental hill-climbing search applied to bayesian network structure learning. In *Proceedings of the 15th European Conference on Machine Learning*, 2004, pp. 1–11.
[2] S. Aouay, S. Jamoussi, and Y.B. Ayed, Particle swarm optimization based method for bayesian network structure learning. In *Proceedings of the International Conference on Modeling, Simulation and Applied Optimization*, 2013, pp. 1–6.
[3] J. Bai and H. Liu, *Multi-objective artificial bee algorithm based on decomposition by PBI method*, Appl. Intell. 45(4) (2016), pp. 976–991.
[4] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper, *The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks*, Springer, Berlin, Heidelberg, 1989.
[5] L.E. Brown, I. Tsamardinos, and C.F. Aliferis, *A novel algorithm for scalable and accurate bayesian network learning*, Stud. Health. Technol. Inform. 107(1) (2004), pp. 711–715.
[6] L.M. De Campos, J.M. Fernández-Luna, J.A. Gámez, and J.M. Puerta, *Ant colony optimization for learning bayesian networks*, Int. J. Approx. Reason. 31(3) (2002), pp. 291–311.
[7] D.M. Chickering, *Learning bayesian networks is np-complete*, Networks 112(2) (1996), pp. 121–130.

[8] H. Chunyu, H. Liu, and P. Zhang, *Cooperative co-evolutionary artificial bee colony algorithm based on hierarchical communication model*, Chinese J. Electron. 25(3) (2016), pp. 570–576.
[9] G.F. Cooper and E. Herskovits, *A bayesian method for the induction of probabilistic networks from data*, Mach. Learn. 9(4) (1992), pp. 309–347.
[10] N. Friedman, M. Linial, I. Nachman, and D. Pe'Er, *Using bayesian networks to analyze expression data*, J. Comput. Biol. 7(3–4) (2000), pp. 601–620.
[11] N. Friedman, I. Nachman, and D. Peér, Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 1999, pp. 206–215.
[12] S. Gheisari and M.R. Meybodi, *Bnc-pso: structure learning of bayesian networks by particle swarm optimization*, Inf. Sci. (Ny) 348 (2016), pp. 272–289.
[13] D. Heckerman, G. Dan, and D.M. Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, Mach. Learn. 20(3) (1995), pp. 197–243.
[14] X.C. Heng, Z. Qin, X.H. Wang, and L.P. Shao, *Research on learning bayesian networks by particle swarm optimization*, Inf. Technol. J. 5(3) (2006), pp. 118–121.
[15] J. Ji, H. Wei, and C. Liu, *An artificial bee colony algorithm for learning bayesian networks*, Soft. comput. 17(6) (2013), pp. 983–994.
[16] C.F. Juang, *A hybrid of genetic algorithm and particle swarm optimization for recurrent network design*, IEEE Trans. Syst. Man Cybern. Part B Cybern. Publ. IEEE Syst. Man Cybern. Soc. 34(2) (2004), pp. 997–1006.
[17] J.I. Jun-Zhong, H.X. Zhang, H.U. Ren-Bing, and L.I. Chun-Nian, *A bayesian network learning algorithm based on independence test and ant colony optimization*, Acta Automatica Sinica 35(3) (2009), pp. 281–288.
[18] J. Kennedy and R. Eberhart, *Particle swarm optimization*, in *Encyclopedia of Machine Learning*, C. Sammut and G.I. Webb, eds., Springer, Boston, MA, 2011, pp. 760–766.
[19] P. Larrañaga, M. Poza, Y. Yurramendi, R.H. Murga, and C.M.H Kuijpers, *Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters*, IEEE. Trans. Pattern. Anal. Mach. Intell. 18(9) (1996), pp. 912–926.
[20] S.L. Lauritzen and D.J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, in *Readings in Uncertain Reasoning*, G. Shafer and J. Pearl, eds., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1990, pp. 415–448.
[21] J.Q. Li and Y.X. Pan, *A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem*, Int. J. Adv. Manuf. Technol. 66(1–4) (2013), pp. 583–596.
[22] J.Q. Li, Q.K. Pan, and P.Y. Duan, *An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping*, IEEE. Trans. Cybern. 46(6) (2016), pp. 1311–1324.
[23] J.Q. Li, H.Y. Sang, Y.Y. Han, C.G. Wang, and K.Z. Gao, *Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions*, J. Clean. Prod. 181 (2016), pp. 584–598.
[24] X.L. Li, S.C. Wang, and X.D. He, Learning bayesian networks structures based on memory binary particle swarm optimization. In *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*, 2006, pp. 568–574.
[25] F. Liu, S.W. Zhang, W.F. Guo, Z.G. Wei, and L. Chen, *Inference of gene regulatory network based on local bayesian networks*, PLoS Comput. Biol. 12(8) (2016), pp. 1–17.
[26] D. Margaritis, *Bayesian network induction via local neighborhoods*, Proc. Adv. Neural Inf. Proc. Syst. 12 (2000), pp. 505–511.
[27] Q.K. Pan, M.F. Tasgetiren, and Y.C. Liang, *A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem*, Comput. Oper. Res. 35(9) (2008), pp. 2807–2839.
[28] R.W. Robinson, Counting labeled acyclic digraphs. *New Directions in the Theory of Graphs*, 1973, pp. 239–273.
[29] Q. Shen, *Learning Bayesian network equivalence classes with ant colony optimization*, J. Artif. Intell. Res. 35 (2009), pp. 391–447.
[30] M. Shen, Z.H. Zhan, W.N. Chen, and Y.J. Gong, *Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks*, Industrial Electron. IEEE Trans. 61(12) (2014), pp. 7141–7151.
[31] D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, and R.G. Cowell, *Bayesian analysis in expert systems*, Stat. Sci. 8(3) (1993), pp. 219–247.
[32] P. Suchánek, F. Marecki, and R. Bucki, *Self-learning bayesian networks in diagnosis*, Procedia Comput. Sci. 35 (2014), pp. 1426–1435.
[33] R. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput. 1(2) (1972), pp. 146–160.
[34] I. Tsamardinos, C.F. Aliferis, and A.R. Statnikov, Algorithms for large scale markov blanket discovery. In *Proceedings of the FLAIRS Conference*, 2003, pp. 376–380.
[35] I. Tsamardinos, C.F. Aliferis, and A. Statnikov, Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 673–678.

[36] T. Wang and J. Yang, *A heuristic method for learning Bayesian networks using discrete particle swarm optimization*, Springer-Verlag, New York, 2010.
[37] H. Xing-Chen, Q. Zheng, T. Lei, and L.P. Shao, Learning bayesian network structures with discrete particle swarm optimization algorithm. In *IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 47–52.
[38] C. Yang, J. Ji, J. Liu, J. Liu, and B. Yin, *Structural learning of bayesian networks by bacterial foraging optimization*, Int. J. Approx. Reason. 69 (2016), pp. 147–167.
[39] L. Zhang and Q. Ji, *A bayesian network model for automatic and interactive image segmentation*, IEEE Trans. Image Proc. A Publ. IEEE Signal Process. Soc. 20(9) (2011), pp. 2582–2593.