Knowl Inf Syst (2010) 24:269–281
DOI 10.1007/s10115-009-0239-6

REGULAR PAPER

A heuristic method for learning Bayesian networks using discrete particle swarm optimization

Tong Wang · Jie Yang

Received: 10 September 2008 / Revised: 13 May 2009 / Accepted: 12 July 2009 /
Published online: 21 August 2009

Abstract Bayesian networks are a powerful approach for representing and reasoning under conditions of uncertainty. Many researchers aim to find good algorithms for learning Bayesian networks from data. And the heuristic search algorithm is one of the most effective algorithms. Because the number of possible structures grows exponentially with the number of variables, learning the model structure from data by considering all possible structures exhaustively is infeasible. PSO (particle swarm optimization), a powerful optimal heuristic search algorithm, has been applied in various fields. Unfortunately, the classical PSO algorithm only operates in continuous and real-valued space, and the problem of Bayesian networks learning is in discrete space. In this paper, two modifications of updating rules for velocity and position are introduced and a Bayesian networks learning based on binary PSO is proposed. Experimental results show that it is more efficient because only fewer generations are needed to obtain optimal Bayesian networks structures. In the comparison, this method outperforms other heuristic methods such as GA (genetic algorithm) and classical binary PSO.

Keywords Bayesian networks · Uncertainty · Structural learning · PSO · Discrete PSO · Genetic algorithms

1 Introduction

In data mining, researchers tend to build a model of the data being mined. To do so, probability-based approaches have been considered as an effective tool because the nature of the models is commonly uncertain. Unfortunately, high computational requirements and the lack of proper representation have hindered the building of probabilistic models. To cope with above two problems, probabilistic graphical models have been proposed. In the past decade,

T. Wang (✉) · J. Yang
Institute of Image Processing and Pattern Recognition,
Shanghai Jiaotong University, 200240 Shanghai, China
e-mail: tongwang@sjtu.org

many variants of probabilistic graphical models have been developed, with the simplest variant being BNs (Bayesian networks) [1].

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. This graphical model has several advantages [2]. One, because the model encodes dependencies among all variables, it readily handles situations where some data entries are missing. Two, a Bayesian network can be used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of intervention. Three, because the model has both a causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in causal form) and data. Four, Bayesian statistical methods in conjunction with BNs offer an efficient and principled approach for avoiding the overfitting of data [2]. As a result, it is widely applied in the field of medical diagnostics, classification systems, software agents for personal assistants, multisensor fusion, and legal analysis of trials [3].

Recently, learning BNs from data has become an increasingly active area of research [3]. Although, sometimes experts can create good BNs from their own experience, it can be a very hard task for large domains. Therefore, many methods have been investigated to automate the creation of BNs using cases collected from past experience [4]. The approaches developed to learn the BNs structure generally fall into two categories: search-and-scoring approach [5] and the dependency analysis approach [6]. The most commonly used search-and-scoring algorithm is the exhaustive search algorithm which can explore all possible structures of a dataset. But for large datasets, the number of possible structure is huge [7]. It is impossible to perform an exhaustive search [8]. Therefore, heuristic search methods [9] have been proposed.

There are two kinds of typical heuristic search algorithms for the structural learning problem, K2 algorithm [24] and hill-climbing algorithm [10]. K2 algorithm, which is something like greedy search method is proposed by Cooper and Herskovits. The idea of this algorithm is incrementally adding a node to a parent set and finding the best parent set to maximize the joint probability of the structure and the database. Although the search does not guarantee to find the globally best structure, it has been widely used, combining with some score metric, to learn a Bayesian network structure. Another most commonly used algorithm is Hill-climbing algorithm. A search space is defined in this algorithm firstly. The states in this space represent possible structures and the operators in this space denote the adjacency of structures. Then, this space is traversed looking for high-scoring functions to complete the optimization. The obvious operators in the search space are "add an edge", "delete an edge", and "reverse an edge". The search starts with some candidate network, which may be the empty one, or one that some expert has provided as a starting point. Then, the space is searched for a high-scoring network by applying those operators. Though these two search algorithms are more efficient, they are easy to be trapped in local optima.

To solve such a problem, a stochastic global optimization algorithm, EA (evolution algorithm) has been introduced by Larrañaga et al. [11] for learning BNs. Four evolutionary algorithmic methods are compared in their approach: steady state, hybrid steady state, elitist, and hybrid elitist. Then, Larrañaga et al. also proposed a genetic algorithm for learning BNs [12]. In their GA implementation, a DAG is represented by a connectivity matrix that is stored as a string (the concatenation of its rows). Recombination is implemented as one-point crossover on these strings, while mutation is implemented as random bit flipping. In a related work, Larrañaga et al. [13] employed a wrapper approach by implementing a GA that searches for an ordering that is passed on to K2. The results of the wrapper approach were comparable to those of their previous GA. Different crossover operators have been implemented in a GA to increase the adaptiveness of the learning problem with good results.

Lam et al. [14] proposed a hybrid evolutionary programming (HEP) algorithm that combined the use of independence tests with a quality based search. In the HEP algorithm, the search space of DAGS is constrained in the sense that each possible DAG only connects two nodes if they show a strong dependence in the available data. The HEP algorithm evolves a population of DAG to find a solution that minimizes the MDL score. The common drawback to the algorithms proposed by Larrañaga and Lam is that the crossover and mutation operators they used are complex and expensive both in memory and runtime.

Recently, PSO [15,16] has been successfully applied in many research and application areas such as Bayesian networks learning. Comparing with GA (genetic algorithm) [17] and SA (Simulated Annealing) [18], PSO's advantages not only lie in its easy implementation and few parameters to adjust but also lie in its strong search capability in the problem space and the ability of quick discovery of optimal solutions [19]. On the other hand, considering the classical PSO algorithm only operates in continuous and real-valued space, some methods are proposed to make it into discrete space and then applied it to Bayesian networks learning. For example, Heng et al. [20,21] still used alphabetic sequence to represent a candidate Bayesian network (a particle in PSO) and then defined some rules to make the computations of velocity updating and position updating possible. These methods are also successfully applied to dynamic Bayesian networks learning [22]. Li et al. [23] introduced memory binary PSO to prevent and overcome premature convergence for Bayesian networks learning. In this paper, binary encoding scheme is introduced to represent a BNs sample and then two modifications of particle moving operations are introduced. One is the velocity updating rules based on the binary representation, and the other is the position updating rules based on "stochastic mutation operation". As a result, continuous PSO is converted to binary discrete PSO. Because the particles are binary not alphabetic sequence, the operations defined on velocity updating and position updating are simpler and easier to implement. Experimental results show that the proposed binary PSOBNL (PSO-based Bayesian Networks Learning algorithm) outperforms other heuristic methods such as GABNL algorithm (GA-based Bayesian Networks Learning algorithm) and classical CPSO (Classical PSO-based Bayesian Networks Learning algorithm) method.

The rest of the paper is organized as follows. The concept of BNs and structural learning is introduced in the following section. Then, we give a brief introduction on classical PSO and then discuss the details of proposed binary PSO for structural learning of BNs in Sect. 3. Results and analysis are represented in Sect. 4. Discussions are given in Sect. 5. Finally, conclusions are summarized in Sect. 6.

2 Bayesian networks

2.1 Basic Bayesian networks

BNs [1] are represented as a directed acyclic graph. Vertices correspond to random variables and edges between the corresponding vertices correspond to the probable influence between random variables. The denomination 'BN' comes from the well-known Bayes theorem. In a BN, the joint probability can be written as follows (recursive factorization)

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

where, $x_1, x_2, \ldots, x_n$ are variables and π_i denotes the parents of the variable x_i.

2.2 Structural learning of BNs

As mentioned above, one category of BNs structural learning algorithms is the search-and-scoring algorithm. It defines a score that evaluating how well the dependencies in a structure match the data and searches for a structure that maximizing the score. The most common scoring approach to evaluating structures is by the posterior probability of the structure given the data. Maximizing the score is maximizing the posterior probability. The posterior probability of a candidate Bayesian network structure B_S can be obtained by applying Bayes rule:

$$p(B_S\,|D) = \frac{P(D\,|B_S)P(B_S)}{P(D)}. \tag{2}$$

where, $P(D|B_S)$ is the likelihood of the data given the BNs structureB_S, and $P(B_S)$ is the prior probability of the structure B_S, and $P(D)$ is the probability of the observed data D. We can ignore the probability $P(D)$ of the data since this value is constant and independent of any particular model B_S. If the prior probabilities $P(B_S)$ of the candidate hypothetical models B_S are equal, then the posterior probability of the model $p(B_S|D)$ is uniquely identified by the likelihood $P(D|B_S)$ of the data given the modelB_S. Cooper and Herskovits [24] showed that the likelihood $P(D|B_S)$ can be obtained by the following form:

$$P(D\,|B_S) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!. \tag{3}$$

Let Z be a set of n discrete variables, where a variable x_i in Z has r_i possible value assignments: $(v_{i1},\ldots,v_{ir_i})$. Let D be a database of m cases, where each case contains a value assignment for each variable in Z. Let B_S denote a belief-network structure containing just the variables in Z. Each variable x_i in B_S has a set of parents, which are represented with a list of variables π_i. Let ω_{ij} denote the jth unique instantiation of π_i relative to D. Suppose there are q_i such unique instantiations of π_i. Define N_{ijk} to be the number of cases in D in which variable x_i has the value v_{ik} and π_i is instantiated as ω_{ij}. Let

$$N_{ij} = \sum_{k=1}^{r_i}N_{ijk}. \tag{4}$$

With the Bayesian Dirichlet metric, we can now search for the structure that scores best.

3 Discrete PSO for Bayesian networks learning

3.1 Classical PSO

Particle swarm optimization (PSO) [15] is an evolutionary computation technique, which is inspired by social behavior of bird flocking and fish schooling. The theoretical frame work of PSO is very simple and PSO is easy to be coded and implemented with computer. Besides, PSO has a powerful exploration ability; it is a gradual searching process that approaches optimal solutions. Thus, nowadays PSO has gained much attention and wide applications in various fields. There, PSO has been used to solve combinatorial optimization problems such as the BNs learning problem. The position and the velocity of the ith particle in the d-dimensional search space can be represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$, respectively $(d = 1, 2, \ldots, N)$. Each particle has its own best

position $X_{best,i} = (x_{best,i1}, x_{best,i2}, \ldots, x_{best,id})$ corresponding to the personal best objective value obtained so far at t-th iteration. The global best particle is denoted by X_{global_best}, which represents the best particle found so far at t-th iteration in the entire swarm. N is the number of variables of the function to be optimized and P is the number of particles in the swarm. The new velocity of each particle is calculated as follows:

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_1 \cdot (x_{best,id}(t) - x_{id}(t)) + c_2 \cdot r_2 \cdot (x_{global\_best,d}(t) - x_{id}(t)), \quad i = 1, \ldots, P, d = 1, \ldots, N \tag{5}$$

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

The inertia weight w provides a balance between global and local exploration and exploitation. The particles maintain high velocities with a larger w, and low velocities with a smaller w. A larger w can prevent particles from becoming trapped in local optima, and a smaller w encourages particles exploiting the same search space area.

The constants c_1 and c_2 are used to decide whether particles x_i prefer moving toward a $X_{best,i}$ position or $X_{global\_best,i}$ position. Low values of them allow particles to roam far from target regions before being tugged back. While, high values result in abrupt movement toward, or past, the target. Generally, w was often set by starting at 0.9 and ends at 0.4 [25]. The acceleration constants c_1 and c_2 were often set to be 2.0 according to past experience. The r_1 and r_2 are random function in the range [0, 1].

The PSO approach utilizes a cooperative swarm of particles, where each particle represents a candidate solution to the problem, to explore the space of possible solutions to the optimization problem of interest. Initialize a population of particles with random positions and velocities and then to 'fly'. Evaluate the fitness at each step of the optimization. The fitness function can be defined in different formula according to different real applications. In this study, its formula is given in Eq. (12). For each particle, compare its current fitness value with the fitness value of its previous best position $X_{best,i}$. If current value is better, then update $X_{best,i}$ with the current value and position. Determine the global best particle of current swarm with the best fitness value. If the fitness value is better than the fitness value of global best position X_{global_best}, then update X_{global_best} with the current value and position of the current particle. Update the velocity and position of each particle according to (5) and (6). If a predefined stopping criterion is met, then output X_{global_best} and its fitness value, otherwise go back to evaluation step.

3.2 Binary PSO

The classical version of the PSO algorithm operates in a continuous search space. In order to solve optimization problems in discrete search spaces, several binary discrete PSO algorithms have been proposed. In a binary discrete space, the position of a particle is represented by an N-length bit string and the movement of the particle consists of flipping some of these bits.

Kennedy and Eberhart [26] propose the first binary version of PSO. This algorithm updates the velocity vector v_i according to Eq. (5), but variable v_{id} is interpreted as the probability of the bit at position d of particle i to become '1'. Since the computed velocity can be greater than 1.0 or even less than 0.0, a sigmoid function (Eq. 7) is applied to variable v_{id} in order to transform velocity values into the range [0.0, 1.0].

$$S(v_{id}) = \frac{1.0}{1.0 + e^{-v_{id}}} \tag{7}$$

The position of the ith particle in [26] is updated according to expression

$$x_{id} = \left\{ \begin{array}{ll} 1 & \text{if } (\text{ran} < S(v_{id})) \\ 0 & \text{otherwise} \end{array} \right\}. \tag{8}$$

where rand is a real random number uniformly distributed between 0 and 1.

3.3 BNs learning based on binary PSO

3.3.1 Representation of particle

Each particle represents a possible solution to the optimization problem. In this application, a particle expresses a candidate BN. We use a representation method used by Larranaga et al. [12] to express a candidate BN. In their researches, the network structure (composed of n nodes) is represented by an $n \times n$ adjacent matrix C. Each element C_{ij} in the matrix is defined as

$$C_{ij} = \left\{ \begin{array}{ll} 1, & \text{if node } j \text{ is a parent of node } i \\ 0, & \text{otherwise} \end{array} \right. \tag{9}$$

By flattening the matrix, the bit-string representation is obtained as following vector:

$$C_{11}C_{21}C_{31} \ldots C_{n1}C_{21}C_{22} \ldots C_{nn} \tag{10}$$

In PSO, it represents the position X_i of every particle i.

3.3.2 Velocity updating rules based on binary representation

The velocity vector V_i of every particle i is updated according to Eq. (5). Here, take an example to show the method of computing the difference between position vectors of X_{global_best}, $X_{best,i}$ and X_i in binary space:

$$X_{global\_best} - X_i = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T - \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T. \tag{11}$$

In Eq. (11), we can see the result must be composed of three real numbers 1, −1, and 0. The number "0" means the corresponding edge of one BN structure X_i is just the edge of the global best BN structure X_{global_best}. The number "1" indicates that the corresponding edge of X_i should be added to close to X_{global_best}. On the other hand, the number "−1" indicates that the corresponding edge of X_i should be removed to close to X_{global_best}. In this example, the number of nonzero elements in result of $X_{global\_best} - X_i$ is 3. That means there are three edges which result in the difference between a candidate BN X_i and the global best BN X_{global_best}. In the same way, the computation of $X_{best,i} - X_i$ can perform like above example. Finally, the number of nonzero elements in results of $X_{global\_best} - X_i$ and $X_{best,i} - X_i$ is substituted into Eq. (5) to get the updated value of velocityV_i.

3.3.3 Position updating rules based on "stochastic mutation operation"

After updating the velocity of particle i in Eq. (5), this particle's position will be updated by the new velocity in Eq. (6). While X_i is a binary vector and V_i is a real number, we can not get updated particle's position by simple addition. In this application, we are going to perform as below. Firstly, round the number of V_i to get an integer M_i. Secondly, randomly select M_i bits in X_i and inverse the number of selected elements (selected bits with a 1 are changed to 0 and a 0 is changed to 1). In this way, the updated particle's position is obtained [27]. After doing this updating operator, the particle has the tendency to fly towards the global best position rather than simply being equal to X_{global_best}.

3.3.4 Fitness function

The fitness function for a candidate network B_S is given by

$$Fitness = \log(P(D\,|B_S)); \tag{12}$$

and $P(D|B_S)$ can be obtained by Eq. (3).

3.3.5 Cyclic problems

When particles 'fly', they may evolve into cyclic graphs which are illegal BNs. A method is introduced to handle cyclic problems by two steps: locating the cycles, transforming the cyclic networks into acyclic networks.

Firstly, we try to find the cycles. A cycle is a path that the first and last node are the same. Identify a source which is a node with no incoming edges and delete it along with all edges outgoing from it. Repeat this process in a remaining diagraph until there is no node with incoming edges. The rest of the diagraph is the cycles if the network really has the cycles.

Next, we try to "break-cycles". The cycles are broken as follows. On the base of finding the two nodes which are each other's parent. One of the nodes in the cycle is randomly selected and its parental relationship to the other node is removed. According to the matrix, the corresponding bit is altered from 1 to 0.

To avoid the occurrence of bidirectional edges and the reflexive edges, we change the candidate network matrix into triangle matrix and let its diagonals are all 0.

4 Experiment

Our experiments are simulated in MatLab 7.0. The computer is Intel P4, 2.66 GHz CPU; 1024 MB RAM and the system is Windows XP Professional. The data sets are generated from the well-known benchmarks of BNs including the ALARM and the ASIA networks. The ASIA network, introduced by Lauritzen and Spiegelhalter to illustrate their method of propagation of evidence, considers a small piece of fictitious qualitative medical knowledge. The ALARM network was constructed by Beinlinch et al. [28] as a prototype to model potential anesthesia problems in the operating room. Because the network, with 37 nodes and 46 directed edges, has a complex structure, it is widely used for evaluating the performance of a Bayesian network learning algorithm. Several techniques exist for simulating BNs, we used probabilistic logic sampling [29], with which we generated a database of 3000 cases for ASIA network. For ALARM network, we will use the 3000 first cases of the database

Table 1 PSO and GA parameter settings

	Population	Generation	Crossover probability	Mutation rate	C_1	C_2	Weight
GA	50 or 100	1000 or 5000	0.5	0.1	–	–	–
PSO	50 or 100	1000 or 5000	–	–	2.0	2.0	0.9–0.4

Table 2 The evaluations of the original structures

Number of cases	log $(P(D\|B_S))$		Method
	ASIA	ALARM	
500	−548.56	−2646.1	GABNL
1000	−1080.0	−5034.5	GABNL
2000	−2154.1	−9729.1	GABNL
3000	−3243.7	−14412	GABNL

Table 3 The performance of GABNL

Data set	Cases	Population	AF	BF	AIF	ANG
ASIA	500	10	−545.7	−545.7	−548.56	283
	500	50	−545.7	−545.7	−548.56	475
	1000	10	−1076.1	−1076.1	−1080.0	240
	1000	50	−1076.1	−1076.1	−1080.0	413
	2000	10	−2154.0	−2154.0	−2154.1	193
	2000	50	−2154.0	−2154.0	−2154.1	388
	3000	10	−3243.7	−3243.7	−3243.7	235
	3000	50	−3243.7	−3243.7	−3243.7	338
ALARM	500	10	−2635.0	−2635.0	−2646.1	963
	500	50	−2635.0	−2635.0	−2646.1	2238
	1000	10	−5027.9	−5027.9	−5034.5	925
	1000	10	−5027.9	−5027.9	−503.45	2438
	2000	10	−9720.9	−9720.0	−9729.1	848
	2000	50	−9720.0	−9720.0	−9729.1	2125
	3000	10	−14411	−14404	−14412	798
	3000	50	−14404	−14404	−14412	2013

that were generated from it by Herskovits [30]. For both database we consider the different subsets consisting of the first 500, 1000, 2000 and 3000 cases.

To compare the performances of PSOBNL (PSO-based Bayesian Networks Learning algorithm) algorithm with GABNL (GA-based Bayesian Networks Learning algorithm) algorithm, we give an experiment of those two algorithms on the above two data sets. Our main target in these experiments is to determine whether PSOBNL is more efficient and effective than GABNL. Additional parameter settings for GABNL and PSOBNL algorithm are given in Table 1. The evaluations of the initial structures using GABNL algorithm for the two databases can be seen in Table 2.

Tables 3 and 4 represent the final result and the summary of the performance comparison between those two algorithms, respectively. Column 1 is the name of the data sets used in our experiment. Column 2 represents the number of the cases used in each data set. Column 3 is the population size used in our experiments. We estimate the performances of the two algorithm using 4 measures, which are represented from column 4 to column 7. Column 4 is the average fitness value of the final solutions (AF). The best fitness values (BF) obtained through the evolution process are in column 5. Column 6 is the average fitness value of the

Table 4 The performance of PSOBNL

Data set	Cases	Population	AF	BF	AIF	ANG
ASIA	500	10	−544.7	−544.7	−547.1	165
	500	50	−544.7	−544.7	−547.1	63
	1000	10	−1071.3	−1071.3	−1078.5	143
	1000	50	−1071.3	−1071.3	−1078.5	49
	2000	10	−2143.1	−2143.1	−2149.1	87
	2000	50	−2143.1	−2143.1	−2149.1	28
	3000	10	−3235.8	−3235.8	−3239.2	128
	3000	50	−3235.8	−3235.8	−3239.2	33
ALARM	500	10	−2624.6	−2624.6	−2635.8	824
	500	50	−2624.6	−2624.6	−2635.8	725
	1000	10	−5013.4	−5013.4	−5024.0	808
	1000	10	−5013.4	−5013.4	−5024.0	698
	2000	10	−9710.8	−9710.1	−9719.3	726
	2000	50	−9710.1	−9710.1	−9719.3	587
	3000	10	−14315	−14294	−14401	630
	3000	50	−14294	−14294	−14401	410

best network obtained in the first generation (AIF). The last column is the average generation that the best-so-far solution is obtained (ANG). Both of the two algorithms are executed 20 times for each data set. The maximum number of generations is 1000 in ASIA data set. The maximum number of generations is 5000 in ALARM data set. The allowable parent set size is limited to four.

From the comparison results, we can see that, the average results of the two algorithms are both better than the evaluation of the initial structure. Besides, the experimental results readily suggest that PSOBNL is more efficient as it use fewer generations to obtain similar or better solutions.

In Table 4, we test PSOBNL with 2 different population sizes: 10, 50. As compared both of the results, we observe that a large population would help to obtain the solution earlier. If we compare the average number of generation needed to obtain the best so far solution (ANG), we observe that it decreases for increasing population size. Because a large population means that more search points are employed, there are more different solutions examined than in a smaller population. On the other hand, the time consumed also increases proportionally with increasing population size.

For better to explain the differences among these three methods, in Fig 1, we compare the performance of proposed binary PSO, classical binary PSO and GA for the two data sets, ASIA-500 and ALARM-2000. For each algorithm, we measure the fitness value of the best-so-far solution averaged over 20 runs as the generations proceed. Although we are testing on two very different data sets, we obtain similar observation that proposed binary PSO converges much faster than GA. Besides, for the same number of generations, proposed binary PSO is observed to perform better than other methods in terms of the average fitness value of the final solution obtained. Though the fitness value obtained by PSOBNL is a little higher than others obtained by GABNL and CPSOBNL, the PSOBNL converges faster than GABNL and CPSOBNL.

5 Discussions

Compared with GA, PSO has many advantages. First, the concept of PSO algorithm is simple and can be implemented in a few lines of code. It requires only primitive mathematical

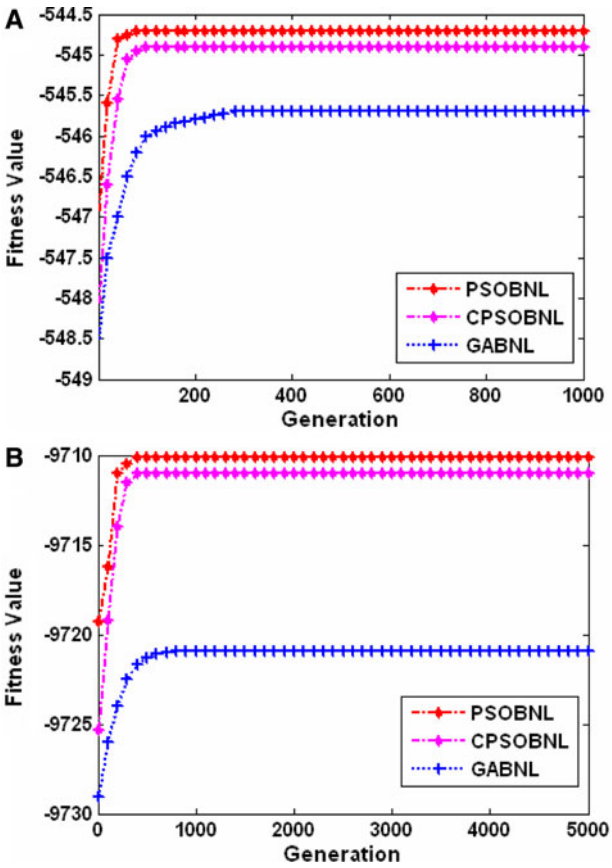

Fig. 1 **a** Typical runs of three algorithms on ASIA-500 data set and **b** typical runs of three algorithms on ALARM-2000 data set

operators, and is computationally inexpensive both in memory and runtime. But, the crossover and mutation operators GA used are complex. Second, a particle swarm system has memory, which the genetic algorithm does not have. A particle swarm also retains the knowledge of where in the search space it performed the best, a memory of a past experience. In a GA, if an individual is not selected for elitism or crossover, the information contained by that individual is lost. As for PSO, individuals who fly past optima are tugged to return towards them; knowledge of good solutions is retained by all particles [31]. But, changes in genetic populations may result in the destruction of previous knowledge of the problem. Third, every particle flies in the candidate problem space, adjusts their velocity and position according to the local best and global best. So, all the particles have a powerful search capability, which can help the swarm find the optimal solution. As for GA, after finding a sub-optimal solution, the GA cannot find a better one.

The running time of PSO is affected less by the problem dimension (feature numbers), but more by the size of data. The GA is affected greatly by the number of features (problem dimension). For ALARM network, because of the increasing dimension of the search space, as the number of cases increase it becomes more difficult to find better results. For

PSO, the computational time increases quickly with the number of generations. In case of GA, the computational time increases slowly with the number of generations. The higher computational time for PSO is due to the communication between the particles after each generation. Hence, in terms of computational time with high number of generations, the GA approach is faster. Fortunately, PSO converged fast. It can obtain the best solution in a given small number of iterations.

In the future, we would extend the structural learning approach based on binary PSO algorithm by trying to find out an optimal ordering of these variables, which is order-based PSOBNL algorithm. Moreover, we also can apply the BNs obtained by our method on a real-life data mining problem such as business information retrieval problem [32], or medical diagnosis problem. BNs have become increasingly popular for handling the uncertain knowledge involved in establishing diagnosis of disease and predicting treatment outcome in various different areas.

We also plan to adapt the proposed structural learning approach to dynamic BNs [33]. Some methods have been proposed for learning dynamic BNs such as [34]. We also plan to learn BNs from incomplete data.

6 Conclusions

In this paper, we have described a new algorithm for learning BNs based on binary PSO. Binary PSO is changed from the typical PSO by two modifications of updating rules for velocity and position. The algorithm is tested on two BNs learning problems to show that it can discover good BNs; we have compared PSOBNL with CPSOBNL and GABNL and found that PSOBNL is more efficient and effective than the other two algorithms. Moreover, PSOBNL requires fewer generations to converge and generates optimal BNs structures.

References

1. Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufman, San Mateo
2. Heckerman D (1997) Bayesian networks for data mining. Data Mining Knowl Discov 1:79–119
3. Heckerman D, Geiger D et al (1995) Learning Bayesian networks: the combination of knowledge and statistical data. Mach Learn 2:197–243
4. Etxeberria R, Larrañaga P, Picaza JM (1997) Analysis of the behaviour of genetic algorithms when learning Bayesian network structure from data. Pattern Recognit Lett 18:1269–1273
5. Suzuki J (1999) Learning Bayesian belief networks based on the minimum description length principle: basic properties. IEICE Trans Fundam Electron Commun Comput Sci E82-A:2237–2245
6. Cheng J, Greiner R, Kelly J, Bell D, Liu W (2002) Learning Bayesian network from data: an information-theory based approached. Artif Intell 137:43–90
7. Robinson RW (1977) Counting unlabeled acyclic digraphs. In: Little CHC (ed) Combinatorial mathematics. Springer Lecture Notes in Math., vol 622, pp 28–43
8. Chickering DM, Geiger D, Heckerman D (1994) Learning Bayesian networks is NP-hard. Microsoft Res., Redmond, WA, MSR-TR-94–17
9. Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Blackwell Scientific Publications, Oxford
10. Alcobé JR (2004) Incremental Hill-climbing search applied to Bayesian network structure learning. In: Proceedings of the 15th European Conference on Machine Learning, Pisa, Italy
11. Larrañaga P, Murga RH, Poza M, Kuijpers CMH (1995) Structure learning of Bayesian networks by hybrid genetic algorithms. In: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics, pp 310–316

12. Larrañaga P, Poza M, Yurramendi Y, Murga RH, Kuijpers CMH (1996) Structure learning of Bayesian networks by genetical algorithms: a performance analysis of control parameters. IEEE Trans Pattern Anal Mach Intell 18:912–926
13. Larrañaga P, Kuijpers CMH, Murga RH, Yurramendi Y (1996) Learning Bayesian network structures by searching for best ordering with genetic algorithm. IEEE Trans Syst Man Cybernet 26(4):487–493
14. Lam W, Bacchus F (1994) Learning Bayesian belief networks-an approach based on the MDL principle. Comput Intell 10:269–293
15. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Perth, pp 1942–1948
16. Senthil Arumugam M, Rao MVC, Chandramohan A (2008) A new and improved version of particles warm optimization algorithm with global–local best parameters. Knowl Inform Syst 16(3):331–357
17. Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press, Michigan
18. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 264(5163):1297–1301
19. Salman A, Ahmad I, Al-Madani S (2002) Particle swarm optimization for task assignment problem. Microprocessors Microsyst 26:363–371
20. Heng X-C, Qin Z, Wang X-H, Shao L-P (2006) Research on learning Bayesian networks by particle swarm optimization. Inform Technol J 5(3):540–545
21. Heng X-C, Qin Z, Tian L, Shao L-P (2007) Learning bayesian network structures with discrete particle swarm optimization algorithm. FOCI 2007, pp 47–52
22. Heng X-C, Qin Z, Tian L, Shao L-P (2007) Research on structure learning of dynamic bayesian networks by particle swarm optimization. CI-ALife 2007, pp 85–91
23. Li X-L, Wang S-C, He X-D (2006) Learning Bayesian networks structures based on memory binary particle swarm optimization. SEAL 2006, LNCS 4247, pp 568–574
24. Cooper GF, Herskovits EA (1992) A Bayesian method for the induction of probabilistic networks from data. Mach Learn 9:309–347
25. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. Proc. IEEE Int. Conf. On Evolutionary Computation. Anchorage, AK, USA, pp. 69–73.
26. Kennedy J, Eberhart RC (1997) A Discrete binary version of the particle swarm algorithm. In: Proceedings of IEEE conference on systems, man and cybernetics, pp 4104–4108
27. Wang XY, Yang J, Teng XL, Xia WJ, Jensen R (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recognit Lett 28:459–471
28. Beinlinch IA, Suermondt HJ, Chavez RM, and Cooper GF (1989) The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In: Proceedings of the Second European Conf. Artificial Intelligence in Medicine, pp. 247–256
29. Henrion M (1988) Propagating uncertainty in Bayesian networks by logic sampling. In: Lemmar J, Kanal L (eds) Proceedings of the 2nd conference on uncertainty artificial intelligence, Amsterdam, pp 149–163
30. Herskovits E (1991) Computer based probabilistic-network construction. Doctoral dissertation. Medical Information Sciences, Stanford University
31. Kennedy J, Eberhart RC (1995) A new optimizer using particle swarm theory. In: Sixth international symposium on micro machine and human science, Nagoya, pp 39–43
32. Wang Z, Wang Q, Wang D-W (2009) Bayesian network based business information retrieval model. Knowl Inform Syst 20(1):63–79
33. Kjaulff U (1992) A computational scheme for reasoning in dynamic probabilistic networks. In: Proceedings of the eighth conferebce on uncertainty in artificial intelligence, pp 121–129
34. Wang KJ, Zhang JY, Shen FS, Shi LF (2008) Adaptive learning of dynamic Bayesian networks with changing structures by detecting geometric structures of time series. Knowl Inform Syst 17(1):121–133

Author Biographies

Tong Wang received the PhD degree in the Institute of Image Processing and Pattern Recognition at Shanghai Jiao tong University in 2009. She is now a lecturer of Institute of Computer and Information, Shanghai Second Polytechnic University. Her research interests include bioinformatics, artificial intelligence and pattern recognition.

Jie Yang received a PhD in computer science from the University of Hamburg, Germany. He is now a professor and director of the Institute of Image Processing and Pattern Recognition, Shanghai Jiao tong University. He has performed more than 20 national and ministry scientific research projects in image processing, pattern recognition, data amalgamation, data mining, and artificial intelligence.