An Improved Evolutionary Approach-Based Hybrid Algorithm for Bayesian Network Structure Learning in Dynamic Constrained Search Space

Jingguo Dai · Jia Ren · Wencai Du · Vladimir Shikhin · Jixin Ma

Received: date / Accepted: date

Abstract Learning Bayesian network (BN) structures from data is a NP-hard problem due to the vastness of the solution space. To address this issue, hybrid approaches that integrate the constraint-based (CB) method and the score-and-search (SS) method have been developed in the literature, but when the constrained search space is fixed and inaccurate, it is very likely to lose the optimal solution, leading to low learning accuracy. Besides, due to the randomness and uncertainty of the search, it is difficult to preserve the superiority of the structures, resulting in low learning efficiency. Therefore, we propose a novel hybrid algorithm based on an improved evolutionary approach to explore BN structure with highest matching degree of data set in dynamic constrained search space. The proposed algorithm involves two phases, namely, the CB phase and the SS phase. In the CB phase, the mutual information (MI) is utilized as the restriction to limit the search space, and a binding parameter is introduced to the novel encoding scheme so that the search space can be dynamically changed in the evolutionary process. In the SS phase, a new operator is developed to pass on the excellent genes from generation to generation, and an update principle for the binding parameter is exploited for the dynamic selection of the search space. We conduct the comparative experiments on the benchmark network data sets and provide performance and applicability analysis of our proposed method. The experimental results show that the new algorithm is effective in learning the BN structures.

Keywords Bayesian networks · Structure learning · Mutual information · Genetic algorithm

Jingguo Dai
College of Information Science and Technology, Hainan University, Haikou, China
Tel.: +86-15248947656
E-mail: djgolivia_edu@126.com

Jia Ren (the corresponding author)
College of Information Science and Technology, Hainan University, Haikou, China
State Key Laboratory of Marine Resource Utilization in the South China Sea, Hainan University , Haikou, China
Tel.: +86-15289805781
Address: Room 216, College of Information Science and Technology, Hainan University, No. 58, Renmin Avenue, Haikou, Hainan Province, 570228, China
E-mail: renjia@hainu.edu.cn

Wencai Du
College of Information Science and Technology, Hainan University, Haikou, China
Faculty of International Tourism and Management, City University of Macau

Vladimir Shikhin
Moscow Power Engineering Institute, Moscow, Russia

Jixin Ma
Department of Computing and Information Systems, University of Greenwich, London, United Kingdom

1 Introduction

A Bayesian network (BN) is a probabilistic graphical model that depicts the uncertain relationships among random variables in the domain. This model is composed of two elements: a directed acyclic graph (DAG) and conditional probability tables (CPTs). The structure of a BN is defined as a DAG, where each node represents a variable and directed edges denote the conditional dependence relationships among nodes. And a CPT quantifies the degree of dependence between two connected variables. Because of the powerful reasoning ability and the visualized knowledge acquisition and representation, BNs have been widely applied in the fields of data mining and artificial intelligence [1,2].

Before using a BN to perform the inference for solving the real-world problems, the construction of the model structure needs to be completed first so that the probabilistic relationships among variables can be found. Hence, structure learning lays the groundwork for BN studies. In recent years, learning the structure of a BN automatically from collected data has gradually displaced the way of constructing a BN manually by domain experts, because the latter is time consuming and subjective [3]. However, finding the BN structure that best describes the dependencies among variables from data is proved to be NP-hard [4]. In the literature, many methods have been proposed to tackle with this structure learning problem, which are usually classified into three schemes with the varying model types.

The constraint-based (CB) method is one of the BN structure learning approaches, which usually makes use of the conditional independence (CI) tests [5,6] to estimate a set of conditional independence relations among variables. The PC-algorithm [7] is a typical example of this approach. The score-and-search (SS) method is another scheme of the BN structure learning approaches, which attempts to search for the optimal network that best fits the given data set in the space of all possible structures [8–10]. There are two important components in the SS method, including a scoring metric which is used to evaluate the quality of every candidate BN structure in terms of the given data set and a search strategy that is adopted to find the solution which has the best score by traversing the search space. Besides, according to the different demands for the structure learning problem, the solution space where the search can be conducted is divided into three categories. Most of the SS algorithms explore the DAG space, which contains the feasible DAGs [11–13]. And some other methods perform the search in the equivalence class space including only partially DAGs (PDAGs), where some edges are not directed [5,14]. Two DAGs belong to the same equivalence class if they define the identical conditional independence relations. Another kind of search space is the variable ordering space, which contains all possible permutations of the variables [15–18]. Since it is much smaller than the DAG space, researchers always consider the search of an optimal variable ordering as an effective way to turn the BN structure learning problem affordable based on the observation that the structure learning of BNs is not NP-hard if given the ordering [15].

However, both two types of the BN structure learning approaches mentioned above have drawbacks. In the CB method, an exponential number of the dependency analysis computations are required, especially for the large-scale networks. And the results of CI tests are unreliable when the sample is small, leading to the inaccurate structures [6]. For the SS method, researchers always adopt heuristic techniques. But since the search space grows exponentially with the increasing number of variables, the convergence speed of the SS approaches tends to be slow. Thus, hybrid methods combining the above two approaches have emerged to overcome these limitations [19–22]. In this type of methods, some CB approach is first used to reduce the search space, which always imposes the restrictions on the feasible solutions, such as the CI tests or the mutual information (MI), and in the subsequent SS stage, the final solution is found in the reduced space based on the defined scoring function.

In this paper, a hybrid algorithm based on an improved evolutionary approach is proposed for the task of BN structure estimation in dynamic constrained search space. In the first stage of the hybrid approach, to restrict the search space, we conduct the dependency analysis by employing the MI, and a new encoding scheme composed of three parts is designed, including a variable ordering, the corresponding upper triangular connectivity matrix, and a binding parameter which is used to complete the dynamic selection of the search space. In the second stage, considering the poor performance of conventional genetic operators, we present a new recombination procedure to inherit the excellent genes from the parental population. And a principle of updating the binding parameter is developed to modify the search space adaptively. A number of experiments on the various data sets of benchmark BNs have been conducted compared against different learning approaches. The empirical results demonstrate that the approach proposed in our paper works well in the BN structure learning problem, having remarkable performance in the convergence speed and the quality of the solution.

The rest of this paper is organized as follows. In Sect. 2, we give a brief introduction of BNs and present related structure learning problems of various methods utilizing evolutionary computation in the previous studies. Sect. 3 describes our new algorithm in detail. In Sect. 4, a comparison among the proposed algorithm and other structure learning methods is presented and the experimental results on the performance and applicability analysis of our new algorithm are reported. Finally, we draw the conclusions and discuss the potential extension for future work in Sect. 5.

2 Preliminaries

2.1 Bayesian networks

A BN is a multivariate statistical model based on probability and graph theory to capture the dependence relationships among a group of random variables $X = \{X_i\}_{i=1,2,\dots,n}$. The model of a BN consists of a graph structure G and a set of distribution parameters $\theta = \{\theta_i\}_{i=1,2,\dots,n}$. $G = (X, A)$ is a DAG with n nodes that denotes random variables X. A directed edge $a_{ij} \in A$ in G directed from X_i to X_j describes the conditional dependence between these two variables, and X_i is called a parent of X_j. All parents of X_j compose its parent set which is denoted by $pa(X_j)$. The parameter θ_i defines the conditional probability distribution $p(X_i \mid pa(X_i))$ for the variable X_i specifying the probability of each possible state of X_i given each possible configuration of $pa(X_i)$, displayed in a CPT. A BN satisfies the Markov condition [6], that is, every variable X_i is independent of its nondescendants given its parents $pa(X_i)$. Therefore, the joint probability distribution of the domain variables can be factored as:

$$p(X) = \prod_{i=1}^{n} p(X_i|pa(X_i)). \tag{1}$$

When we construct a BN for the real-life applications, two tasks need to be accomplished, including structure learning, which focuses on the identification of the topology of a BN, and parameter learning, which is the calculation of conditional probability based on a given BN structure. Thus, structure learning becomes the first issue before the estimation of the numerical parameters. However, due to the vast search space with respect to the increasing number of variables in a BN, it is knotty for the researchers to use exhaustive search through all possible structures. Besides, the computational complexity remains exponential with currently available exact algorithms, and such methods are ticklish for BNs with more than around 30 vertices given our actual computational capacity [23]. Hence many greedy local search techniques and heuristic methods have received much attention in the literature [24–26]. Moreover, in order to improve the search efficiency of the heuristic methods for BN structure learning, structural restrictions codifying the information about the observed data are defined, leading to the emergence of hybrid approaches.

2.2 Bayesian network structure learning

When a hybrid algorithm is applied to the task of the BN structure estimation, the modeling process can be separated into two phases: the CB phase and the SS phase.

2.2.1 The CB phase

It is a challenging problem to find a BN structure accommodating the given data set, which best represents the causal dependencies between variables. In fact, Robinson has proved that the number of all possible BN structures with respect to n nodes in the DAG space can be calculated by the following equation [27]:

$$f(n) = \begin{cases} 1, & n = 0; \\ \sum\limits_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i), & n > 0. \end{cases} \tag{2}$$

We can see that the number of possible BN structures has exponential growth with the increasing number of variables, making it intractable to learn the model directly and precisely in such enormous search space. Hence, when using the hybrid algorithms, many researchers impose some structural restrictions on the candidate BNs, including existence restrictions, absence restrictions and ordering restrictions [28], at the beginning of the exploration in order to narrow down the search space.

Existence and absence restrictions take into account the association relationships between variables, which respectively consider the existence and absence of arcs and/or edges. The CI test is treated as a typical way to make absence restrictions. Since the computation results of high-order CI tests may not be reliable [20], many methods in the literature only use order-0 and order-1 CI tests to identify the possible conditional independence relations [6,20]. Nevertheless, when performing CI tests, all possible separation sets may need to be examined to decide whether two variables are independent. It is obviously a difficult task for large-scale networks, even using low-order tests.

In other research works, researchers use the MI to establish existence restrictions [14,19]. The MI value between two variables X and Y can be defined as follows:

$$I(X;Y) = \sum_{i=1}^{r} \sum_{j=1}^{s} p(a_i, b_j) \log \frac{p(a_i, b_j)}{p(a_i)p(b_j)} \tag{3}$$

where r and s are the number of possible states of X and Y, respectively. a_i is the ith possible value of X, b_j

is the jth possible value of Y. The MI between two variables is symmetric in nature, i.e., $I(X;Y) = I(Y;X)$. It can be seen that if X is completely unrelated to Y, $I(X;Y) = 0$, and the higher value of the MI indicates the stronger relationship between two variables. However, it is difficult to determine whether there is an edge between two nodes by simply using Eq. 3 to evaluate the MI. Thus Li et al. [14] and Chen et al. [19] both introduced the constraint given by Eq. 4 to identify the relationship between a pair of variables more explicitly, which is composed of two conditions. If any of the conditions is satisfied, it is assumed that there is a strong dependence between the two variables, so an edge connecting these two nodes should be established.

$$I(X;Y) \geq \alpha_{MI} \cdot MMI(X) \; or \; I(Y;X) \geq \alpha_{MI} \cdot MMI(Y) \tag{4}$$

where $MMI(X)$ and $MMI(Y)$ denote the maximum MI (MMI) for node X and Y respectively, $0 \leq \alpha_{MI} \leq 1$ is a binding parameter. We can see from the above expression that to limit the search space, α_{MI} plays an important role in obtaining the number of connections which must be included in the candidate structures for the subsequent search phase. In the previous works, the value of the binding parameter was predefined, for example, it was set at 0.9 in [19]. It is noteworthy that if improper value of α_{MI} is adopted, there exists a great possibility that the reduced search space cannot contain the optimal structure due to the permanent residence of some wrong edges or the permanent loss of crucial connections in the fixed search space depending on the predefined value of α_{MI}.

Besides the previous two types of restrictions, another possibility is to use ordering restrictions. When given a node ordering, the feasible networks in the search space must satisfy the constraint that the possible parents of each node $X_i (i = 1, 2, \ldots, n)$ in G should be chosen from those nodes preceding X_i. It is easy to see that the predefined variable ordering can narrow down the size of the search space. In fact, taking the encoding scheme of a genetic algorithm (GA) as an example, the cardinality of the search space equals $2^{\binom{n}{2}}$ under the ordering assumption, while in the general case, it is 2^{n^2} using an adjacent matrix. K2 algorithm [29] is a well-known heuristic method of learning BN structures using a node ordering as input. However, obtaining an appropriate permutation of nodes requires a large amount of priori information that may be hard to acquire in many practical applications. So some concerns have been raised regarding the searches in the ordering space [15,30]. And the experimental results in [15] showed that the ordering-based search could find a better scoring network, compared with the structure-based search, when there were a large number of variables.

Table 1 Proportion of randomly generated adjacent matrices without any cycles

n	$ff(n)$	$f(n)$	$F(n)$	$p(n)$
1	1	1	2	50.000%
2	2	3	16	18.750%
3	8	25	512	4.883%
4	64	543	65536	0.829%

2.2.2 The SS phase

In this paper, considering that a GA, which is one type of the evolutionary approaches, has been regarded as an important heuristic technique due to the reliable global search capability [31,32], we exploit an improved GA so as to search for an appropriate BN structure that accommodates the given data set in the DAG space.

In the existing GA-based BN structure learning algorithms, to encode every candidate structure with n nodes in the search space, researchers often resort to an adjacent matrix $A = (a_{ij})_{n \times n}$, where

$$a_{ij} = \begin{cases} 1, \; if \; X_i \; is \; a \; parent \; of \; X_j; \\ 0, \qquad otherwise. \end{cases} \tag{5}$$

We can see that the adjacent matrix A is an n-dimensional Boolean matrix. Larrañaga et al. [24] rewrote the above matrix in the Eq. 6 as an individual:

$$a_{11}a_{12} \ldots a_{1n}a_{21}a_{22} \ldots a_{2n} \ldots a_{n1}a_{n2} \ldots a_{nn} \tag{6}$$

And in the algorithm proposed by Lee et al. [13], a chromosome was defined using upper and lower triangular matrices:

$$a_{12}a_{13} \ldots a_{1n}a_{23} \ldots a_{2n} \ldots a_{(n-1)n}a_{21}a_{31}a_{32} \ldots a_{n(n-1)} \tag{7}$$

However, the possibility of obtaining a DAG is not high when an arbitrary adjacent matrix is employed to act as an individual, as shown in Table 1, where $F(n) = 2^{n \times n}$ is the number of all possible n-dimensional Boolean matrices, $f(n)$ is defined in Eq. 2, and $p(n) = f(n)/F(n)$ denotes the proportion of randomly generated adjacent matrices without any cycles.

It can be seen from Table 1 that the proportion of randomly generated adjacent matrices which satisfy the acyclicity feature decreases significantly with the growth of the number of nodes. Thus a repair operator which makes the illegal graph acyclic has to be carried out, leading to high time complexity. Then many researchers tend to avoid generating illegal networks with the help of the node ordering [8,12,15]. The second column in Table 1 presents the number of structures $ff(n)$ with different number of nodes given the predefined node ordering. Compared against $f(n)$, it is obvious that the search space with the given ordering is much smaller than the DAG space.

As for the genetic operators that are utilized during the evolutionary process to produce the offspring, Santos et al. developed the random multi-point crossover operator (RMX) [16] and the distance-based mutation operator (DMO) [17] to explore the influence of a single operator on the search for suitable node orderings of BNs. Nevertheless, when a GA searches for the optimal BN structure during the evolutionary process, the crossover operator acting a significant part in the convergence speed is probably ineffective. In other words, since the newly generated BN structure may be markedly different from its parents after implementing the crossover, it is likely that the offspring cannot inherit the positive qualities from the parental population. Thus, in the hybrid evolutionary algorithm (HEA) proposed by Wong et al. [11], a novel operator called merge was introduced. It attempted to modify the structure with the parts of another one to increase the scores of the offspring, replacing the conventional crossover. Lee et al. [13] also designed new recombination operator to increase the possibility that the merits of the parental population can be passed on to the offspring.

When the search process is carried out, it is essential for such an optimization problem to define a scoring function that estimates the goodness of each candidate structure in regard to the given data set. In the light of different theoretical foundations, there are varying metrics used in the literature [11,14,15]. And considering that the Bayesian information criterion (BIC) is based on the penalized maximum likelihood approach [3], we employ this type of scoring metric derived from information theory in our work. The formula for the BIC metric is given by the following expression:

$$score_{BIC} = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ijk}\log\frac{m_{ijk}}{m_{ij*}} - \sum_{i=1}^{n}\frac{q_i(r_i-1)}{2}\log m. \tag{8}$$

where n is the number of nodes, q_i defines the count of the possible configurations for $pa(X_i)$, r_i is the number of the states for X_i, and m_{ijk} corresponds to the sample count of the kth possible value of X_i given the jth possible configuration of $pa(X_i)$. Besides, $m_{ij*} = \sum_{k=1}^{r_i} m_{ijk}$, m is the total sample count.

The BIC score consists of two parts: the log-likelihood calculating the matching degree of the candidate model with the samples and a penalty relevant to the dimension of the model itself as well as the size of the data set. Note that the BIC metric makes a tradeoff between the accuracy and the complexity of the model, thus it searches for a simple BN structure that best fits the observed data. Moreover, as a feature common to other metrics, the BIC score is node-decomposable [3] and can be rewritten as follows:

$$score_{BIC} = \sum_{i=1}^{n} score_{BIC}(i). \tag{9}$$

where

$$score_{BIC}(i) = \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ijk}\log\frac{m_{ijk}}{m_{ij*}} - \frac{q_i(r_i-1)}{2}\log m \tag{10}$$

In Eq. 10, $score_{BIC}(i)$ is defined as the family BIC score of X_i. Besides, a union of one node X_i and its parent set $pa(X_i)$ forms a substructure of a BN, a network can accordingly be viewed as an aggregation of $pa(X_i)(i = 1, 2, \ldots, n)$. Therefore, we only need to compute the combination of the scores for smaller factors like a single node whose parent set has a variance when the new individuals have been generated. On the other hand, the BIC metric as well as its first part encoding the log-likelihood is score equivalent [33], i.e., the structures in the same equivalence class are assigned the same score. It can cause the structure identification problem. And the merge operator designed in [11] was able to avoid the search spending much time on the BN structures in the same equivalence class.

The main focus of our algorithm is to obtain a BN structure that fits the given data set by using the effective genetic operators in dynamic constrained search space. More precisely, on the one hand, to restrict the search space, we exploit the MI to design a flexible constraint which is able to dynamically change during the evolutionary process by using an innovative updating principle. On the other hand, allowing for the risk of the disruption of the excellent substructures when the offspring are produced and the interference from the BN structures in the same equivalence class, we take advantage of the decomposability of the BIC metric to substitute a novel recombination operator for the conventional crossover.

3 Method

In this section, a hybrid approach using an improved GA is presented to deal with the BN structure learning problem. The learning ability of the proposed method can be enhanced with the help of several strategies. First, a novel encoding scheme is designed which can support the effective representation of candidate BN structures. Second, the knowledge of dependencies among the variables from the calculations of the MI is introduced during the evolutionary process to adaptively modify the search space. Third, a new recombination operator based on the BIC metric is provided so that the superior genes are able to be inherited to the offspring. Since the MI is employed in an improved GA for structure learning of BNs, the proposed approach in our paper is called MIIGA.

3.1 Representation and notation

To represent candidate BN structures, the following two elements have been taken into account in MIIGA. On the one hand, the conventional coding scheme where the BN structure is codified by an adjacent matrix has the limitation that newly generated networks may not satisfy the acyclic property, and considering that the node ordering assumes that parent nodes should appear before the children, thus the new representation of BN structures encodes the node ordering in the individuals to guarantee the valid networks. On the other hand, in order to limit the search space, the information provided by Eq. 4 is used in MIIGA, and recalling that an improper value of α_{MI} in Eq. 4 tends to obtain an unsatisfactory structure in the search phase, we attempt to change the binding parameter α_{MI} adaptively in the evolutionary process, instead of predefining its value. Therefore, we propose a new encoding scheme in which each chromosome consists of three parts, including: (1) the permutation of variables $\prec$, where if $X_i \in pa(X_j)$ then node $X_i \prec$ node X_j, (2) a string generated by flattening the upper triangular adjacent matrix which described the connections among the nodes consistent with the given ordering, and (3) the binding parameter α_{MI} which is supposed to be evolved as well during the evolutionary process. The detailed design is given as follows:

$$i_1 i_2 \dots i_n a_{12} a_{13} a_{23} a_{14} a_{24} a_{34} \dots a_{1n} a_{2n} \dots a_{(n-1)n} \alpha_{MI} \tag{11}$$

where there are n nodes in the network, i_k denotes the kth node in the ordering, a_{jk} describes whether there

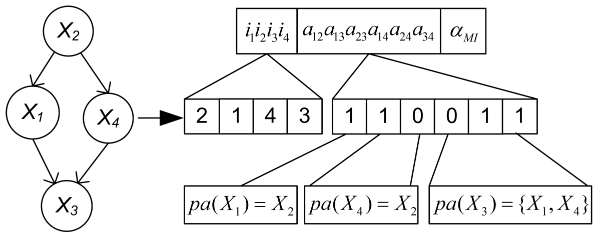

Fig. 1 An example of the new encoding scheme

is an edge directed from jth node to kth node based on the ordering which is defined in the first part of the chromosome, when it is true, $a_{jk} = 1$; otherwise $a_{jk} = 0$, and α_{MI} is the binding parameter raised in Eq. 4. An illustration is given in Fig. 1.

Since the node ordering is used to interpret the candidate structures, the genetic operators in the proposed method are closed operators [24]. Besides, unlike the fixed value of α_{MI} in other related works, the last locus in the chromosome treated as a identifier for α_{MI} is constantly changed in the closed interval $[0, 1]$, leading to the variation of the number of connections that should be involved in the candidate networks. Note that as the evolutionary search proceeds, individuals owning an improper value of α_{MI} will ultimately be eliminated. In fact, according to the previous analysis of the importance of α_{MI}, when its value is too large or too small, the BIC score of the BN structure will be low due to the absence of some crucial edges or the presence of many wrong connections. Then there is a greater chance that this individual will be eliminated because of its inferior fitness. Consequently, the individuals having more appropriate value of α_{MI} are able to survive for the further evolution.

3.2 The algorithm

3.2.1 The initialization

At the beginning of the proposed algorithm, an initial population needs to be defined. Note that if the initial population gains good performance to get closer to the optimal solution, it is very likely to take less time to find an appropriate network that accommodates the given data set. So in the initialization procedure, we adopt MI to recover useful information from the data set. More specifically, starting with an empty graph, as for each individual in the initial population, the node ordering is determined by the MMI. Recalling that the MI is symmetric, when there is the MMI between node $X_i (i = 1, 2, \dots, n)$ and another node, an undirected edge connecting these two nodes is added to the current graph, and the directions of these edges are given randomly. After obtaining the ordering, the search space is preliminarily reduced. Then the value of the binding parameter α_{MI} is stochastically initialized. Based on the matrix $Info$ which stores the MI values among variables, some connections are established in the existing network if they satisfy the constraint in Eq. 4, so the search space is further narrowed down corresponding to the value of α_{MI}. The initial individual is chosen from this reduced search space.

To select individuals for the reproduction of the offspring in the evolutionary process, the relative merits for different network structures are evaluated by means of the BIC metric in MIIGA. Besides the node-decomposable property, in this paper, we further decompose the BIC family score of each node into two parts. One is the log-likelihood function estimating how well the substructure of node X_i fits the given data set. Another part is the penalization function that encodes the complexity of the substructure. These two items of node X_i are stored respectively in the matrices $(llS_i)_{m\times n}$ and $(dpS_i)_{m\times n}$ during the evolutionary process, where m is the size of the population and n is the number of variables. The decomposition form of Eq. 10 is defined as follows:

$$score_{BIC}(i) = llS_i - dpS_i \tag{12}$$

where

$$llS_i = \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ijk}\log\frac{m_{ijk}}{m_{ij*}} \tag{13}$$

$$dpS_i = \frac{q_i(r_i-1)}{2}\log m \tag{14}$$

In Eq. 13, llS_i is a minus, and dpS_i in Eq. 14 is non-negative. According to the features of the above two items, we adopt different part of the BIC family score in the following subsections to find the highest scoring network by traversing the varying search space.

3.2.2 Priority-recombination operator

In view of the poor performance of the conventional crossover in exchanging information among the population members, we introduce a novel operator called priority-recombination (PR) in our proposed algorithm, in order to make the offspring successfully inherit the positive features from the parental population without the disruption of the superior substructures obtained so far in the evolutionary process. And considering that the log-likelihood function which is the first part of the BIC metric measures the degree of matching between BN structure and the data set, $(llS_i)_{m\times n}$ is adopted in the PR operator. According to Eq. 13, the greater llS_i means the higher match degree between the substructure of X_i and the data. Hence, we make use of this feature to find better offspring. Moreover, though the log-likelihood function is score equivalent, leading to a difficult identification problem for BN structures in the same equivalence class, the PR operator can handle it well by producing the offspring having different log-likelihood function values from those of the parental structures.

In the PR operator, two individuals are taken as input, i.e., a parental network *old_indiv* and a BN structure randomly selected from the previous generation, which is denoted as $former\_indiv$. Recalling that the BIC metric is node-decomposable and the BN structure is a combination of $pa(X_i)(i = 1, 2, \ldots, n)$, we attempt to create a better network which contains good substructures of both two inputs. That is, a new individual *new_indiv* can be generated by selecting the parent set $new\_pa(X_i)$ from $old\_pa(X_i)$ which denotes the parent set of the node X_i in the *old_indiv* and $former\_pa(X_i)$ which is defined as the parent set of the node X_i in the $former\_indiv$, so that the sum of the llS_i for the *new_indiv* ($\sum_{i=1}^{n} new\_llS(X_i)$) is greater than the sum of the llS_i for the $former\_indiv$ ($\sum_{i=1}^{n} former\_llS(X_i)$) as well as the sum of the llS_i for the *old_indiv* ($\sum_{i=1}^{n} old\_llS$ (X_i)). The pseudocode for the PR operator is displayed in Algorithm 1.

At the beginning of the operator, the computations of Eq. 15 are executed:

$$diff\_llS(X_i) = former\_llS(X_i) - old\_llS(X_i), \quad (i = 1, 2, \ldots, n) \tag{15}$$

And a node set $D = \{X_j\}$, $j \in \{1, 2, \ldots, n\}$ is produced by sorting $diff\_llS(X_i)$ in descending order. Since the positive value of $diff\_llS(X_i)$ supposes that $former\_pa(X_i)$ is better than $old\_pa(X_i)$, the replacement of $old\_pa(X_i)$ with $former\_pa(X_i)$ is taken into consideration when $diff\_llS(X_i) > 0$. Starting from the first node X_j in the set D, the PR operator invokes the subprogram getpaset to obtain a subset of nodes *mid_paset* and a modified node ordering *node_order*. Thus, old_pa (X_j) is replaced by $former\_pa(X_j)$ for every node $X_j \in mid\_paset$, and *node_order* is updated to guarantee the validity of the ordering, i.e., each node in $former\_pa(X_j)$ should precede the node X_j after substituting $former\_pa(X_j)$ for $old\_pa(X_j)$ in *old_indiv*. The pseudocode for the procedure getpaset is presented in Algorithm 2.

Algorithm 1 Priority-recombination (PR) operator

```
1: Calculate $diff\_llS(X_i)$=$former\_llS(X_i)$-$old\_llS(X_i)$, $i = 1, 2, \ldots, n$;
2: Sort $diff\_llS(X_i)$ in descending order and store the corresponding node set in $D = \{X_j\}$, $j \in \{1, 2, \ldots, n\}$;
3: Set $mark\_set = \emptyset$, $change\_paset = \emptyset$, $llS = old\_llS$ and $indiv = old\_indiv$;
4: while size($mark\_set$) $< n$ do
5:     Get a node $X_j \in D$, which has not been considered yet;
6:     Set $mid\_paset = \emptyset$ and $node\_order = indiv(1, 1 : n)$;
7:     Invoke the procedure getpaset to obtain $mid\_paset$ and $node\_order$;
8:     $paset = mid\_paset - change\_paset$;
9:     Compute $sum$=sum($diff\_llS(X_j{}')$), $X_j{}' \in paset$;
10:    if $sum > 0$ then
11:        Replace $llS(X_j{}')$ with $former\_llS(X_j{}')$, $X_j{}' \in paset$;
12:        Update $indiv$ according to $node\_order$;
13:        $change\_paset = change\_paset \cup paset$;
14:        $mark\_set = mark\_set \cup paset$;
15:    else
16:        $mark\_set = mark\_set \cup X_j$;
17:    end if
18: end while
19: if $llS > old\_llS$ then
20:     $new\_indiv = indiv$;
21: else
22:     $new\_indiv = old\_indiv$;
23: end if
```

Algorithm 2 Procedure getpaset

```
1: if $X_j \in mid\_paset$ then
2:     Return immediately;
3: end if
4: $mid\_paset = mid\_paset \cup X_j$;
5: for $X_j{}'' \in former\_pa(X_j)$ do
6:     if $X_j{}''$ lies behind $X_j$ in $node\_order$ then
7:         Update $node\_order$ by inserting $X_j{}''$ in front of $X_j$;
8:         Invoke getpaset;
9:     end if
10: end for
```

The subprogram getpaset is a recursive procedure. The input *mid_paset* begins with an empty set and *node_order* is initialized by the ordering of *old_indiv*. The node X_j is at first added to the set *mid_paset*. Then, every node $X_j{}'' \in former\_pa(X_j)$ is checked whether it is positioned behind X_j in *node_order*. If it is true, the procedure has to modify *node_order* by shifting $X_j{}''$ in front of X_j and recursively invoke itself for the node $X_j{}''$. Consequently, the elements in *mid_paset* are the nodes whose parent set needs to be displaced.

In our implementation, we randomly select half of the population to conduct the PR operator. When a better network structure is produced, it is regarded as the offspring. Otherwise, the parental network *old_indiv* which fails to give a better network is put back in the current population and performs the following operations.

3.2.3 Updating principle of the binding parameter

Allowing for the new encoding scheme proposed in Sect. 3.1, there is a significant locus in the chromosome, i.e., the binding parameter α_{MI}, which needs to be adaptively changed during the evolutionary process to implement the dynamic selection of the search space. In our algorithm, for the unselected individuals and the selected ones which are unable to produce the better networks in the PR operator, the value of α_{MI} should be updated. We store the edges that satisfy the Eq. 4 in the matrix $Infomat$. It can be seen that if the value of α_{MI} is high, the size of $Infomat$ is small. While if that value is too low, the worst case is that $Infomat$ contains all possible connections. In MIIGA, we use $(dpS_i)_{m\times n}$ to renovate α_{MI}. Recalling that the more complex substructure of X_i gains the higher dpS_i, we provide an updating principle based on *best_dpS* which is the sum of dpS_i of the best individual obtained so far in the evolutionary process, presented as follows:

$$\begin{cases} new\_\alpha_{MI} = old\_\alpha_{MI} - \Delta\alpha, & \sum\limits_{i=1}^{n} dpS_i < best\_dpS; \\ new\_\alpha_{MI} = old\_\alpha_{MI}, & \sum\limits_{i=1}^{n} dpS_i = best\_dpS; \\ new\_\alpha_{MI} = old\_\alpha_{MI} + \Delta\alpha, & \sum\limits_{i=1}^{n} dpS_i > best\_dpS. \end{cases} \quad (16)$$

Eq. 16 describes the strategy used to adjust α_{MI} of an individual by comparing against the current optimal network. Specifically, if the sum $\sum\limits_{i=1}^{n} dpS_i$ for one network is less than *best_dpS*, we consider that the complexity of this structure is lower than that of the best network, so in order to increase the number of connections, the value of α_{MI} is supposed to be smaller, then we give α_{MI} a decrement by $\Delta\alpha$. Otherwise, it is assumed that the sum $\sum\limits_{i=1}^{n} dpS_i$ for the candidate individual is bigger than *best_dpS*, α_{MI} should be assigned an increment by $\Delta\alpha$ because the number of connections in the structure has to be reduced. But if $\sum\limits_{i=1}^{n} dpS_i$ equals *best_dpS*, there is no need to change the value of α_{MI}. Therefore, the search space has been refined by altering the binding parameter α_{MI}. The next step is to complete the mutation for the networks in the modified search space.

3.2.4 Information-driven mutation operator

In our algorithm, considering the randomness of the conventional mutation, we provide an operator called information-driven mutation (IDM) to produce the offspring. There are two steps in the operator. In the beginning, one of the mutation types, including the addition, deletion and reverse of a directed edge, is randomly chosen, and then the BIC score and the MI which have been stored before are used to implement the mutation. Briefly, when the IDM decides to execute the deletion or reverse in the network, an existing directed edge will be selected with larger probability if its deletion or reverse leads to a higher scoring structure. And when the IDM decides to add an edge to the network, the connection with larger MI value will be first considered.

3.2.5 Structure learning phases of MIIGA

The algorithm of MIIGA is shown in Algorithm 3. It consists of 2 phases, including 7 steps:

1) The CB phase:

Step 1 Calculation of the MI and preprocessing the results;

2) The SS phase:

Step 2 Initialization;

Step 3 Evaluation of the fitness;

Step 4 Tournament selection;

Step 5 Implementation of the PR operator;

Step 6 Binding parameter α_{MI} update;

Step 7 Information-driven mutation.

3.2.6 Time complexity analysis

In this section, we analyze our algorithm in a worst-case scenario for time complexity. As the proposed method contains 7 steps, we discuss the time complexity for each phase. Assume that the population size is m, the maximum number of generations is g. Let N and n be the number of data samples and number of variables, respectively. Let r be the maximum number of possible states for any variable. In Step 1, we need to compute the MI values for each pair of variables. Owing to the symmetry of the MI, we need to only calculate $n(n-1)/2$ MIs. The complexity of computing each MI value is $O(4r^2+N)$ or simply $O(N)$, since r is generally much smaller than N. Thus, the time complexity of estimating all the MIs is $O(Nn^2)$. To sort MI values for each variable, we adopt quicksort algorithm, and the time complexity for that in the worst case is $O(n^2)$. Because there are n variables, the complexity of sorting MI values for all nodes is $O(n^3)$. Therefore, the overall complexity for Step 1 is $O(Nn^2)+O(n^3)$. For Step 2, creating an individual based on the node ordering by using the MMI requires $O(n^2)$ time, so the complexity for the initial population is $O(mn^2)$. Additionally, the inner loop of the proposed algorithm, from Step 3 to Step 7, includes fitness evaluation, tournament selection, recombination, binding parameter update and mutation. In Step 3, we calculate the fitness values according to the BIC metric, and the time required for computing BIC scores is $O(Nn^2)$, so the complexity for fitness evaluation is $O(mNn^2)$. For Step 4, suppose there are s competitors in the tournament, we need to first select s individuals from the current population, that is, the number of alternatives is $\binom{m}{s}$ and then sort the fitness values of the selected individuals, thus the complexity for tournament selection is $O(m^3s^2)$ or simply $O(m^3)$ as s is normally much smaller than m. In Step 5, in view of the pseudocode for the PR operator displayed in Algorithm 1, we use $O(n^2+nq)$, where q is the maximum number of parents for any variable, to implement the recombination for each pair of individuals. Note that there are only half of the population that participate the recombination, so the time complexity for Step 5 is $O(mn^2/2+nqm/2)$. Besides, in Step 6, we update the binding parameter according to Eq. 16, and then modify the second part of the chromosome for the corresponding individual based on Eq. 4. Note that the worst case to select individuals which execute binding parameter update requires there is none of the selected networks which successfully produce the better structure in Step 5, thus the time complexity for Step 6 is $O(mn^2)$. For Step 7, we need $O(m)$ to implement the mutation operation. Therefore, the complexity of the inner loop is $O(mNn^2+m^3+mn^2/2+nqm/2+mn^2+m)$. Moreover, the outer loop is related to the number of iterations g, consequently, the time complexity of the proposed algorithm is $O(Nn^2)+O(n^3)+O(mn^2)+O(g\cdot(mNn^2+m^3+mn^2/2+nqm/2+mn^2+m))$.

4 Experimental results and discussion

In this section, to study the performance of the proposed algorithm, we select four benchmarks of BNs including the ASIA, CAR DIAGNOSIS2, Alarm and Child5 networks to conduct the experimental evaluation. Comparative studies are also performed with the following existing methods: conventional genetic algorithm (CGA) [24], dual genetic algorithm (DGA) [12], Bayesian Network PowerConstructor (BNPC) [34], K2 greedy algorithm [29], BN Construction algorithm using Particle Swarm Optimization (BNC-PSO) [10], Improved K2 algorithm via Markov Blanket (IK2vMB) [18] and Max-Min Hill Climbing (MMHC) [22]. Besides, the performance analysis is also presented through computer simulations to demonstrate the validity of the proposed method. Finally, we design a set of experiments aimed at testing the applicability of the algorithms presented in this paper.

Algorithm 3 MIIGA Algorithm

```
1: Calculate the MI for each node and store the results in a matrix $Info$;
2: Sort $Info$ in a descending order;
3: store the descending MI values and the corresponding index matrix in two matrices $IV$, $IO$ respectively, and store the MMI in a vector $MMI$;
4: for each individual in the initial population $pop(0)$ do
5:     Obtain a node ordering using $MMI$;
6:     Get the value of $\alpha_{MI}$ randomly to restrict the search space;
7:     Store the edges that satisfy Eq. 4 in the matrix $Infomat$ by checking $IV$, $IO$;
8:     Create an individual based on $Infomat$ in a reduced solution space;
9: end for
10: Evaluate the BIC score for each individual in $pop(0)$ and store two parts of BIC score, i.e., the log-likelihood and the penalization function value for every node in each individual in the matrices $(llS_i)_{m\times n}$ and $(dpS_i)_{m\times n}$, respectively;
11: Find the current optimal individual and store its penalization function value in $best\_dpS$;
12: Produce the population $pop(1)$ for the first generation using the tournament selection;
13: for $G$=1:$generation\_size$ do
14:     Randomly pick $pop\_size/2$ individuals in the $pop(G)$, the rest are marked as $np$;
15:     for each picked individual $old\_indiv$ do
16:         Randomly select an individual $former\_indiv$ from $pop(G-1)$;
17:         Invoke the procedure priority-recombination;
18:         if a better individual is successfully produced then
19:             Regard the new individual as the offspring;
20:         else
21:             Put back $old\_indiv$, and mark it as $np$;
22:         end if
23:     end for
24:     for each individual marked as $np$ do
25:         Update $\alpha_{MI}$ by an increment or decrement of $\Delta\alpha$, compared with $best\_dpS$;
26:         Renovate the search space according to the value of $\alpha_{MI}$;
27:         Perform the information-driven mutation operator to alter the structure in the new search space;
28:         Regard the new individual as the offspring;
29:     end for
30:     Evaluate the BIC scores for the new offspring;
31:     Sort the BIC scores of individuals, including all new offspring and $pop(G)$, in a descending order;
32:     Store the $pop\_size$ individuals with top $pop\_size$ highest BIC scores in a intermediate population $pop'$;
33:     Select the population $pop(G+1)$ from $pop'$ using the tournament selection and update $best\_dpS$;
34: end for
35: Return the individual with highest BIC score in any generation as output;
```

4.1 Experimental setup

To assess the performance of the methodology, we carry out the experiments on six different synthetic data sets, which are generated from four well-known BNs. The ASIA network [35] is a small BN that consists of eight nodes and eight edges. The network represents a fictitious medical example of whether a patient has tuberculosis, lung cancer, or bronchitis, related to the chest clinic. All random variables are discrete and each can take two discrete states, as shown in Fig. 2. A database of 1000 cases is utilized to train the BN. CAR_DIAGNOSIS2 (Brent Boerlage, `http://www.norsys.com`) is a network of moderate size used for diagnosing why a car does not move based on spark plugs, headlights, main fuse, among others. It contains 18 discrete nodes and 20 edges. The random variables can take two or three states. Fig. 3 shows the structure of the CAR_DIAGNOSIS2 network depicted by Netica from which three data sets of different sizes are collected, including 500, 1000 and 2000 sample cases respectively. The Alarm network [22] presented in Fig. 4 is a medical diagnostic system for patient monitoring, which contains 37 nodes and 46 edges. The maximum number of possible states for any variable in Alarm is four. Another BN is Child5 [22]. It is a relatively large network containing 100 nodes and 126 edges. The variables in Child5 take a varying number of states from 2 to 6. Data sets sampled from the latter two BNs were released by the authors in [22], and we select 5000 cases to perform the task of structure recovery for both two BNs. Table 2 provides the general information about the data sets used in our experiments.

The experimental platform used in our paper is a personal computer with Intel Core i5-5300U, 2.30 GHz, 32 bits architecture, 4 GB RAM memory and under Windows 7. The programs are all compiled using the Matlab software release R2014a. In order to prevent the exponential growth in the number of possible configurations of the parent sets with the number of parents,

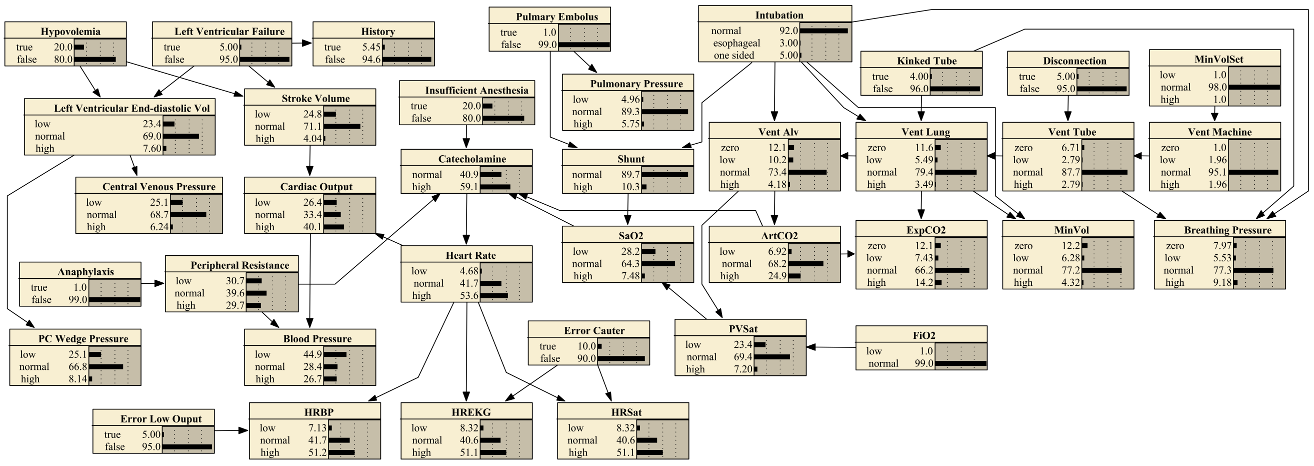
Hypovolemia true 20.0 false 80.0
Left Ventricular Failure true 5.00 false 95.0
History true 5.45 false 94.6
Pulmary Embolus true 1.0 false 99.0
Intubation normal 92.0 esophageal 3.00 one sided 5.00
Kinked Tube true 4.00 false 96.0
Disconnection true 5.00 false 95.0
MinVolSet low 1.0 normal 98.0 high 1.0
Pulmonary Pressure low 4.96 normal 89.3 high 5.75
Insufficient Anesthesia true 20.0 false 80.0
Left Ventricular End-diastolic Vol low 23.4 normal 69.0 high 7.60
Stroke Volume low 24.8 normal 71.1 high 4.04
Catecholamine normal 40.9 high 59.1
Shunt normal 89.7 high 10.3
Vent Alv zero 12.1 low 10.2 normal 73.4 high 4.18
Vent Lung zero 11.6 low 5.49 normal 79.4 high 3.49
Vent Tube zero 6.71 low 2.79 normal 87.7 high 2.79
Vent Machine zero 1.0 low 1.96 normal 95.1 high 1.96
Central Venous Pressure low 25.1 normal 68.7 high 6.24
Cardiac Output low 26.4 normal 33.4 high 40.1
SaO2 low 28.2 normal 64.3 high 7.48
ArtCO2 low 6.92 normal 68.2 high 24.9
ExpCO2 zero 12.1 low 7.43 normal 66.2 high 14.2
MinVol zero 12.2 low 6.28 normal 77.2 high 4.32
Breathing Pressure zero 7.97 low 5.53 normal 77.3 high 9.18
Anaphylaxis true 1.0 false 99.0
Peripheral Resistance low 30.7 normal 39.6 high 29.7
Heart Rate low 4.68 normal 41.7 high 53.6
PC Wedge Pressure low 25.1 normal 66.8 high 8.14
Blood Pressure low 44.9 normal 28.4 high 26.7
Error Cauter true 10.0 false 90.0
PVSat low 23.4 normal 69.4 high 7.20
FiO2 low 1.0 normal 99.0
Error Low Ouput true 5.00 false 95.0
HRBP low 7.13 normal 41.7 high 51.2
HREKG low 8.32 normal 40.6 high 51.1
HRSat low 8.32 normal 40.6 high 51.1

Fig. 4 The Alarm network

Table 2 General information about data sets

Data set	Original network	Size	No.nodes	No.edges	Domain	Max In/Out -degree	BIC score
ASIA-1000	ASIA	1000	8	8	2	2/2	-2.3105e+03
CAR_DIAGNOSIS2-500	CAR_DIAGNOSIS2	500	18	20	2-3	5/3	-3.3253e+03
CAR_DIAGNOSIS2-1000	CAR_DIAGNOSIS2	1000	18	20	2-3	5/3	-6.2745e+03
CAR_DIAGNOSIS2-2000	CAR_DIAGNOSIS2	2000	18	20	2-3	5/3	-1.2093e+04
Alarm-5000	Alarm	5000	37	46	2-4	4/5	-4.8724e+04
Child5-5000	Child5	5000	100	126	2-6	2/7	-3.1752e+05

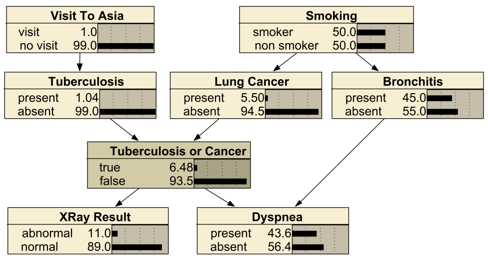
Visit To Asia visit 1.0 no visit 99.0
Smoking smoker 50.0 non smoker 50.0
Tuberculosis present 1.04 absent 99.0
Lung Cancer present 5.50 absent 94.5
Bronchitis present 45.0 absent 55.0
Tuberculosis or Cancer true 6.48 false 93.5
XRay Result abnormal 11.0 normal 89.0
Dyspnea present 43.6 absent 56.4

Fig. 2 The ASIA network

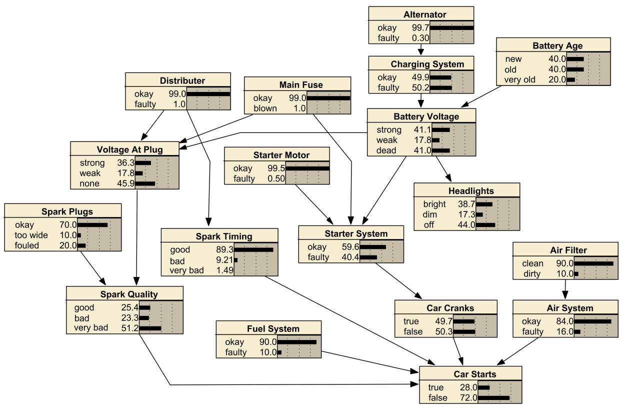
Alternator okay 99.7 faulty 0.30
Battery Age new 40.0 old 40.0 very old 20.0
Charging System okay 49.9 faulty 50.2
Distributer okay 99.0 faulty 1.0
Main Fuse okay 99.0 blown 1.0
Battery Voltage strong 41.1 weak 17.8 dead 41.0
Voltage At Plug strong 36.3 weak 17.8 none 45.9
Starter Motor okay 99.5 faulty 0.50
Headlights bright 38.7 dim 17.3 off 44.0
Spark Plugs okay 70.0 too wide 10.0 fouled 20.0
Spark Timing good 89.3 bad 9.21 very bad 1.49
Starter System okay 59.6 faulty 40.4
Air Filter clean 90.0 dirty 10.0
Spark Quality good 25.4 bad 23.3 very bad 51.2
Car Cranks true 49.7 false 50.3
Air System okay 84.0 faulty 16.0
Fuel System okay 90.0 faulty 10.0
Car Starts true 28.0 false 72.0

Fig. 3 The CAR_DIAGNOSIS2 network

the maximum number of parents of each node is limited to five for simplicity. Additionally, the same BIC metric is employed in CGA, DGA and BNC-PSO to evaluate the fitness of the candidate BN structures, so that it is convenient to compare their performance with that of MIIGA.

Allowing for the stochastic nature of GA, PSO and MMHC, it is necessary to execute more than one run to verify the final solution in the simulations. Thus, for MIIGA, CGA, DGA, BNC-PSO and MMHC, we run 30 times for each testing instance and average the results. While BNPC is a deterministic algorithm using dependency analysis approach, we only execute once for each data set. And since the relevant program provided by Causal Explorer (`http://www.dsl-lab.org`) can only obtain a PDAG as structure-learning result, we give a random direction for those undirected edges. Meanwhile, K2 algorithm obtains the same output even for repeated runs due to the deterministic property with the same node ordering. Therefore, in all experiments, to ensure a fair comparison, the node ordering used as input in K2 algorithm is given stochastically without the prior information, and we execute this approach 30 times with different orderings. Furthermore, since IK2vMB employs a maximum weight spanning tree (MWST) to establish the initial search graph, which needs to assign the root node first, we still conduct the related experiments 30 times with different root node.

4.2 Comparison among MIIGA and other algorithms

In the following, we compare the proposed algorithm MIIGA with the other seven different existing methods, i.e., CGA, DGA, BNPC, K2, BNC-PSO, IK2vMB and MMHC, based on all of the data sets. The final experimental parameters are confirmed by large numbers of experiments. Both GA-based algorithms and PSO-based approach, including MIIGA, CGA, DGA and BNC-PSO, use the same population size and the same maximum number of generations, which are 100 and 500 respectively. For MIIGA, the tournament size is 2 in a tournament selection. In particular, we set $\Delta\alpha$ to be 0.02 in the Eq. 16 for updating the value of the binding parameter α_{MI}. For CGA, the crossover rate is 0.5, and the mutation rate is 0.1. And for DGA, the crossover rate is 0.65, the mutation rate is 0.05. Besides, the parameter values of BNC-PSO are the same as those used in [10].

To examine the effectiveness and efficiency of these eight algorithms, we adopt the following four measures:

1) ABB. The average BIC score of the final individuals.

2) ART. The average running time in seconds.

3) AGB. The average generation that the best-so-far individual is obtained.

4) AHD. The average structural Hamming distance (SHD) between the best individual and the original BN structure, including the number of mistakenly added, deleted and reversed directed edges.

In Table 3 and Table 4, there are the experimental results of the comparison among the proposed method and the other seven different algorithms in four measures. Notice that structure estimation of BNs from data using exact algorithms is a computational hard task with more than around 30 variables, Table 4 provides the summary of the structure-learning results carried out on the Alarm and Child5 networks, which are much larger than the two benchmarks of BNs used in Table 3, to further evaluate the validity of the proposed algorithm. It is noteworthy that except BNPC which only obtains the SHD, the other seven approaches have been executed 30 times for each data set, so the figures in the table are the averages of 30 trials, but for K2, IK2vMB and MMHC, only the average of the BIC scores and the SHD have been recorded because these three methods deal with BN structure learning problem without using population-based approach, thus AGB is not available and ART is neither comparable. Moreover, numbers in parentheses below the name of data sets are the BIC scores of the original networks used as reference and the standard deviations over 30 executions are the numbers in parentheses presented under the figures in Table 3 and Table 4. Comparing with BNPC, K2 and IK2vMB, MIIGA shows almost better results in learning BN structures according to the AHD statistics. However, when reconstructing the Child5 network, IK2vMB performs better than the proposed algorithm in terms of both BIC score and structural difference. As for MMHC, we can clearly observe from the AHD that its simulation results come closest to those of MIIGA for all data sets, and MIIGA performs slightly better than MMHC in structure recovery.

In Fig. 5, considering that the figures of the experimental results for the Child5 network are much larger than those of other BNs, which are reported in Table 4, so to make the comparison easily, three figures are plotted for clarity to show the comparison of three of the measures defined above, i.e., AHD, AGB and ART, among four algorithms including MIIGA, CGA, DGA and BNC-PSO for five different data sets except for the Child5-5000, using the error bars where the confidence level is set to be 95%. In the figures, CAR_DIAGNOSIS is abbreviated to CAR for the convenience.

Note that the smaller value of AHD implies that the estimated BN structure is more similar to the original network, it can be seen from Fig. 5a and Fig. 5b that for all experimental data sets, MIIGA spends fewer generations to find better structures which are closer to the original networks in terms of the number of generations that are taken to find best-so-far solution (AGB) and structural difference (AHD), compared with other three methods. Besides, we observe in Fig. 5c that MIIGA takes the least time to finish the execution among all comparative methods on first four data sets except for the Alarm-5000 under the same termination condition (ART) even though the proposed algorithm needs to use part of the running time to initialize the population by employing the MI, rather than the random initialization like other methods. Therefore, in view of the AHD and AGB statistics, we can draw the conclusion that MIIGA has better performance in learning the BN structures over its competitors.

Different from CGA and BNC-PSO which both use an adjacent matrix to represent an individual, MIIGA adopts a node ordering and the corresponding upper triangular matrix in the encoding scheme. Thus, during the evolutionary process, there is no need to implement the repairing operator since the genetic operators are closed, while CGA and BNC-PSO have to spend a significant part of the total execution time in transforming

Table 3 Comparison of the proposed and other existing methods on the Asia and CAR_DIAGNOSIS2 networks

Data set	Method	ABB	ART	AGB	AHD
ASIA-1000 (-2.3105e+03)	MIIGA	-2.3130e+03 (4.0081)	74.7182 (47.6697)	49.6667 (31.2535)	1.2000 (0.6103)
	CGA	-2.3174e+03 (4.6447)	91.1733 (6.8727)	61.6333 (17.7151)	6.0333 (1.4259)
	DGA	-2.3180e+03 (7.3747)	102.4410 (9.7233)	76.9333 (21.9135)	4.1333 (1.4794)
	BNC-PSO	-2.3144e+03 (4.2952)	56.2261 (3.0005)	25.7333 (15.9740)	5.4000 (1.6938)
	MMHC	-2.3191e+03 (6.1148)	/	/	2.0667 (0.2582)
	K2	-2.3319e+03 (14.2652)	/	/	7.9333 (2.6514)
	IK2vMB	-2.3278e+03 (10.9498)	/	/	5.2000 (1.5844)
	BNPC	/	/	/	2
CAR_DIAGNOSIS2-500 (-3.3253e+03)	MIIGA	-3.1228e+03 (14.1615)	140.0882 (34.4109)	145.5333 (121.1231)	8.5667 (0.7739)
	CGA	-3.1762e+03 (49.1105)	179.7834 (4.8787)	363.1667 (65.8573)	19.9000 (2.1391)
	DGA	-3.1111e+03 (28.0405)	217.1188 (4.3239)	427.4667 (50.9339)	13.3000 (1.5790)
	BNC-PSO	-3.1202e+03 (31.6380)	141.0358 (1.4320)	69.3333 (13.0736)	15.9333 (1.3374)
	MMHC	-3.1933e+03 (46.2161)	/	/	12.2000 (1.4736)
	K2	-3.2571e+03 (73.3665)	/	/	20.9000 (3.9071)
	IK2vMB	-3.1584e+03 (24.9771)	/	/	18.4667 (2.7258)
	BNPC	/	/	/	11
CAR_DIAGNOSIS2-1000 (-6.2745e+03)	MIIGA	-6.0417e+03 (12.9550)	162.7560 (54.0104)	33.0000 (17.1464)	7.5000 (0.9738)
	CGA	-6.1357e+03 (52.5363)	199.7532 (5.6645)	376.8667 (55.3937)	18.8000 (2.1399)
	DGA	-6.0937e+03 (50.1974)	220.4229 (11.6764)	416.5667 (48.2220)	14.3667 (1.8286)
	BNC-PSO	-6.0957e+03 (47.5138)	143.3565 (1.2525)	77.5333 (27.1506)	16.8000 (1.3995)
	MMHC	-6.1477e+03 (89.8395)	/	/	9.6000 (1.1212)
	K2	-6.2950e+03 (107.9888)	/	/	23.5333 (3.9105)
	IK2vMB	-6.1977e+03 (90.2832)	/	/	18.3333 (3.4773)
	BNPC	/	/	/	8
CAR_DIAGNOSIS2-2000 (-1.2093e+04)	MIIGA	-1.1859e+04 (3.6043)	111.8929 (45.9301)	75.9000 (46.6087)	5.0667 (0.6915)
	CGA	-1.1978e+04 (53.4192)	129.7303 (1.6758)	399.2333 (65.9460)	18.3000 (1.8223)
	DGA	-1.1932e+04 (58.6156)	222.7345 (6.3109)	430.0333 (51.6503)	13.4000 (2.1592)
	BNC-PSO	-1.1964e+04 (49.6195)	148.3080 (1.8731)	78.7667 (20.9378)	18.1000 (2.0736)
	MMHC	-1.1989e+04 (213.2586)	/	/	6.5333 (1.2459)
	K2	-1.2215e+04 (209.4246)	/	/	23.8333 (5.0452)
	IK2vMB	-1.1991e+04 (58.1766)	/	/	20.6333 (2.4703)
	BNPC	/	/	/	7

Table 4 Comparison of the proposed and other existing methods on the Alarm and Child5 networks

Data set	Method	ABB	ART	AGB	AHD
Alarm-5000 (-4.8724e+04)	MIIGA	-4.9995e+04 (211.1703)	1.0397e+03 (394.1418)	65.9000 (74.8347)	9.0667 (5.7950)
	CGA	-5.4279e+04 (104.0603)	1.2157e+03 (114.1812)	488.3333 (15.1483)	33.6000 (5.5746)
	DGA	-5.6796e+04 (856.2856)	790.9612 (19.6425)	496.2333 (3.0477)	28.8667 (3.3501)
	BNC-PSO	-4.9487e+04 (392.8225)	865.0046 (45.8914)	385.1667 (61.0438)	14.2333 (4.2725)
	MMHC	-4.9329e+04 (431.5650)	/	/	10.2667 (3.7123)
	K2	-5.8978e+04 (3.4295e+03)	/	/	60.7000 (6.5292)
	IK2vMB	-5.0676e+04 (1.1062e+03)	/	/	20.2000 (4.6713)
	BNPC	/	/	/	10
Child5-5000 (-3.1752e+05)	MIIGA	-3.2240e+05 (2.5277e+03)	1.0299e+04 (2.4004e+03)	42.4667 (51.3452)	36.8000 (11.9146)
	CGA	-3.2323e+05 (5.7673e+03)	5.6261e+03 (1.1545e+03)	489.8400 (15.1676)	81.1600 (13.3157)
	DGA	-3.2448e+05 (3.4676e+03)	6.0752e+03 (77.9467)	498.3333 (2.4398)	60.5333 (9.3034)
	BNC-PSO	-3.2440e+05 (2.8613e+03)	9.3081e+03 (1.5113e+03)	496.6667 (4.0478)	54.4074 (12.2608)
	MMHC	-3.2085e+05 (109.8116)	/	/	39.3333 (1.5430)
	K2	-3.3089e+05 (2.0785e+03)	/	/	115.4000 (13.8330)
	IK2vMB	-3.1974e+05 (902.3164)	/	/	26.2000 (6.2555)
	BNPC	/	/	/	55

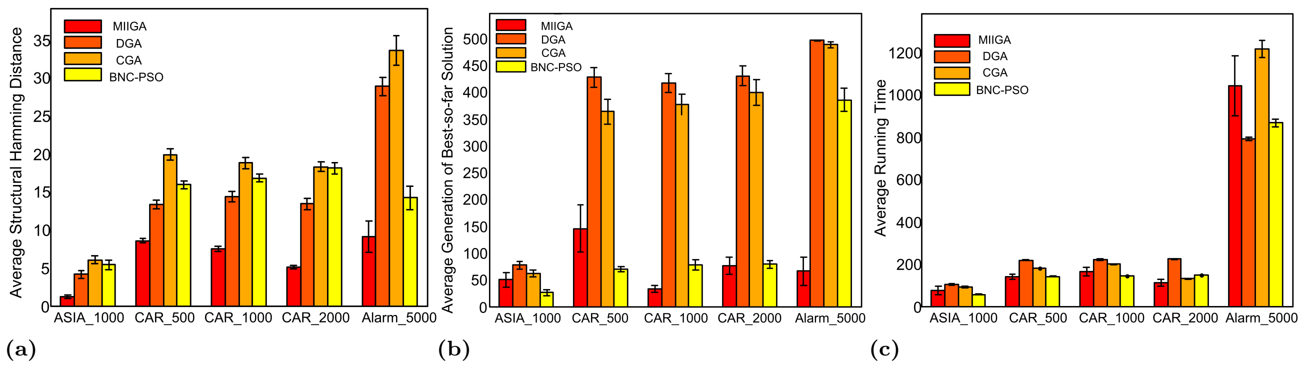

Fig. 5 Three measures for the four algorithms tested with five different data sets. (a) The AHD statistics. (b) The AGB statistics. (c) The ART statistics

the illegal offspring into DAGs. Besides, the PR operator takes two existing individuals as input to generate a better network that inherits the superior substructures from the parental individuals, so we can directly reuse the previous computational results of the BIC scores to save time. Therefore, in terms of the ART statistics, MIIGA performs better than CGA and BNC-PSO with relatively small networks. But when we compare the larger BNs like the Alarm and Childs networks in Table 4, taking into account that MIIGA needs to alter the value of the binding parameter based on the update principle provided in Sect. 3.2.3 to refine the search space as the evolutionary search proceeds, it is time-consuming with the increasing number of nodes. So MIIGA costs more time than other methods to finish the simulation using the Alarm-5000 and Child5-5000 data sets.

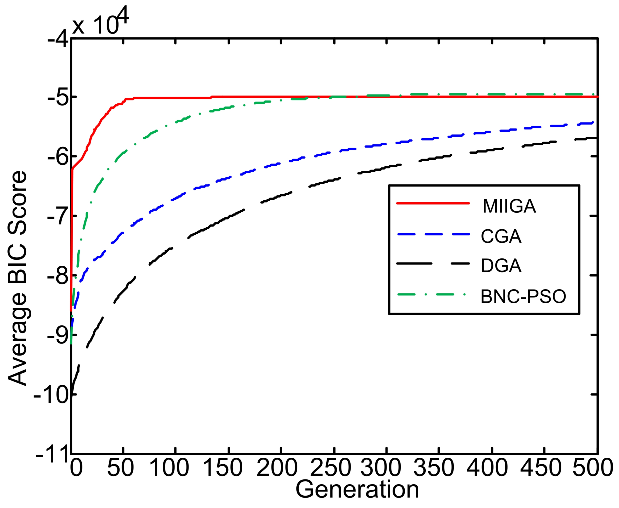

Fig. 7 The BIC score of the best-so-far structure averaged over 30 executions on the Alarm-5000 data set for four algorithms

Both DGA and MIIGA use the node ordering as the restriction which is a strong constraint to reduce the search space. When an inappropriate permutation of variables is given, it is very likely that some fundamental edges will be deleted. But MIIGA is able to dynamically change the search space by updating the binding parameter α_{MI} in the evolutionary process. Moreover, recalling that the log-likelihood function is score equivalent, it is inefficient for a GA to consume much time in searching for the networks that belong to the same equivalence class. Unlike DGA, MIIGA introduces the PR operator to replace the conventional crossover, which can produce the offspring having different log-likelihood function values from those of the parental population. According to the above two virtues, the PR operator is able to explore more BN structures in different equivalence classes and in varying search space, making MIIGA more likely to find better solutions. Consequently, comparing the count of the structural difference (AHD) and the BIC scores (ABB), we observe that MIIGA shows better performance than DGA.

As shown in Fig. 6 and Fig. 7, we compare four algorithms including MIIGA, CGA, DGA and BNC-PSO more intuitively for the three different data sets of the CAR_DIAGNOSIS2 network and the Alarm-5000 data set, respectively. For each method, we evaluate the BIC score of the best-so-far structure averaged over 30 executions as the generations proceed. In all figures, the solid lines present the results of MIIGA, the short-dashed lines record the learning results of CGA, the long-dashed lines represent the results of DGA and the dash-and-dot lines show the results of BNC-PSO. It can be seen that apart from the CAR_DIAGNOSIS2-500 data set, MIIGA initially outperforms the other three methods and continues to find the network having the highest BIC score among all four approaches for the same number of generations until the termination. Moreover, we can see that MIIGA has an obvious advantage in the convergence speed and simultaneously always converges to a better structure in terms of the higher BIC score with the larger data sets.

Comparing these four algorithms, we believe that the excellent performance of the proposed method could be attributed both to the dynamic selection of the reduced search space and to the introduction of the PR operator. For CGA, DGA and BNC-PSO, randomly selected initial population is used at the beginning of the evolution. Because the random synthetic networks cannot reflect the knowledge of the given data set, it is hard to ensure the rapid convergence to the optimal solution. While for MIIGA, it takes advantage of the MI to filter the useless structures when implementing the initialization, making the initial population closer to the optimal structure. In the evolutionary process, the search space is adaptively changed by updating the binding parameter, thus the proposed method is less likely to get trapped in a local optimum. Furthermore, instead of stochastically exchanging some part of the parental individuals like the other three methods, the PR operator in MIIGA identifies the excellent substructures and passes on them to the offspring, resulting in the better quality of the individuals for the subsequent generation.

4.3 Performance analysis of MIIGA

To further study the performance of MIIGA, we conduct the following experiments focusing on the factors that affect the efficiency and the effectiveness of the proposed algorithm. On one hand, we want to investigate the contributions of the MI and the PR operator. On the other hand, we desire to validate the effect of the binding parameter α_{MI}. All experiments are carried out on the CAR_DIAGNOSIS2-2000 data set and the results are the averages of 30 trials. The default settings we assume in the simulations are the population size is 100, the maximum number of generations is 500, and the tournament size is 2 in a tournament selection. Besides, the Mann-Whitney test [36] at the 0.05 significance level is applied to the estimation of the difference between MIIGA and other methods in the diverse measures.

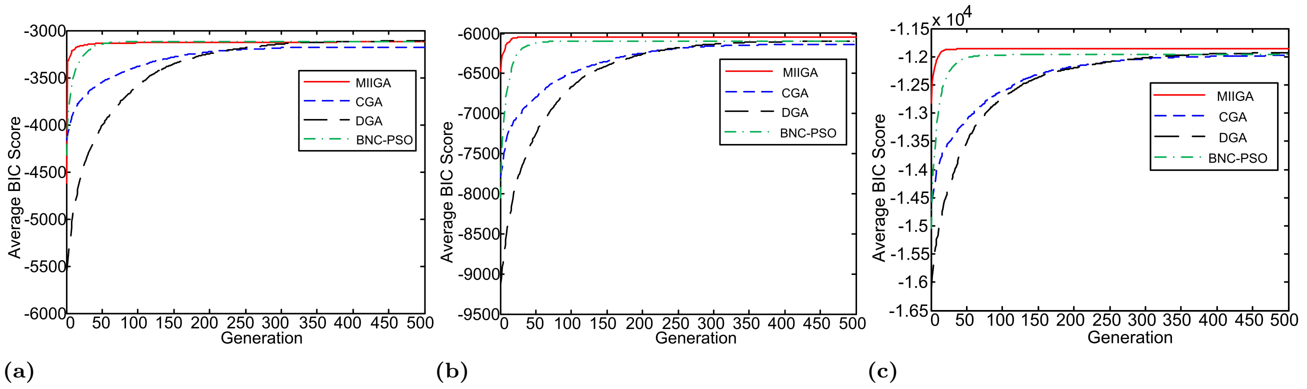

Fig. 6 The BIC score of the best-so-far structure averaged over 30 executions on three different data sets for four algorithms (a) CAR_DIAGNOSIS2-500 data set. (b) CAR_DIAGNOSIS2-1000 data set. (c) CAR_DIAGNOSIS2-2000 data set

4.3.1 Influence of the MI

In order to limit the search space for the subsequent SS phase, before starting the evolutionary iteration, it is necessary to impose the restriction on the candidate networks to filter some invalid BN structures. Considering that the MI implies the degree of the dependence between two random variables, we use the MI to select the networks that satisfy the constraint presented in Eq. 4 as the initial population. In this subsection, we modify MIIGA by initializing the population randomly without the help of the MI. This implementation is called "MIIGA-InitMI". Table 5 provides a summary of the comparative results of these two approaches. Note that apart from the four measures mentioned above to evaluate the performance of the methods, there is a new item recorded in the table, which is denoted as ABI that represents the average BIC score of the best network obtained in the first generation.

From Table 5, we observe that the ART, AGB and AHD of MIIGA are significantly lower than those of MIIGA-InitMI (p- values are 0.0019, 9.8892e-11, and 4.5357e-10 respectively). Thus, MIIGA does outperform the modified method MIIGA-InitMI in terms of the efficiency and the effectiveness. According to the ABI statistics, it is obvious that the initial population in MIIGA has much better quality (p-value is 3.0199e-11), compared against MIIGA-InitMI, and it evidently affects the accuracy performance of the final solution, as the ABB statistics suggest (p-value is 1.5543e-6). Hence, we believe that the initialization by means of the MI is able to effectively enhance the search efficiency for the best structure (AGB) and meanwhile make the final solution much closer to the original network. In fact, because the MI is a brilliant way to interpret the dependence between variables, we utilize it to initially produce the population. Each individual in the initial population has to satisfy the constraint which contains a large amount of information about the original network based on the given data set. Therefore, MIIGA can avoid the necessity of selecting the individual from the entire search space of possible BN structures and simultaneously keep the performance of the individuals at the high level in the SS phase.

4.3.2 Influence of the PR operator

When the conventional crossover is applied to the population, since it always tends to recombine two individuals arbitrarily, the newly generated offspring may not inherit the superior features from the parents. Thus, we present the PR operator based on the log-likelihood function to produce better offspring without the split of the existing best-so-far substructures. In this experiment, we concentrate on the influence of the PR operator on the performance of MIIGA. Hence, we compare the proposed algorithm with another method called " MIIGA-PR", and the only difference between two approaches is that we perform the conventional crossover, instead of the PR operator in MIIGA-PR. The comparison results are provided in Table 6.

From the table, it can be observed that MIIGA needs less iteration to find better solutions according to the AGB and AHD statistics. Apart from the average values, the results of the Mann-Whitney test indicate that the difference between MIIGA and MIIGA-PR is significant (p-values are 2.4238e-09 and 9.7213e-04, respectively). Since the PR operator can preserve the excellent genes when exchanging information among the population members, it is easier for MIIGA to obtain higher scoring structures. While in view of the randomness of the conventional crossover, it is very likely that MIIGA-PR has to repeatedly recover the information that has been lost because of the arbitrary recombi-

Table 5 Performance of modified MIIGA with the random initialization

	ABB	ART	AGB	AHD	ABI
MIIGA	-1.1859e+04 (3.6043)	111.8929 (45.9301)	75.9000 (46.6087)	5.0667 (0.6915)	-1.2830e+04 (584.4365)
MIIGA-InitMI	-1.1904e+04 (69.8365)	151.5861 (51.0457)	267.6000 (89.7935)	8.6667 (2.4821)	-1.6015e+04 (299.6605)

Table 6 Performance of modified MIIGA without the PR operator

	ABB	ART	AGB	AHD	ABI
MIIGA	-1.1859e+04 (3.6043)	111.8929 (45.9301)	75.9000 (46.6087)	5.0667 (0.6915)	-1.2830e+04 (584.4365)
MIIGA-PR	-1.1871e+04 (18.2177)	250.3739 (44.8972)	241.3667 (101.3612)	6.4333 (1.7943)	-1.2955e+04 (519.6785)

nation. Therefore, the best-so-far individual obtained by MIIGA-PR always appears later than MIIGA. Besides, the PR operator produces the offspring by selecting parent sets from the parental population, thus there is no need to invoke the fitness computation procedure. Consequently, MIIGA takes significantly less time to finish than MIIGA-PR in terms of the ART (p-value is 1.3289e-10).

4.3.3 Effect of the binding parameter α_{MI}

According to the encoding scheme described in Sect. 3.1, different value of the binding parameter α_{MI} is placed on the last locus of every chromosome. As the evolutionary search proceeds, the value of α_{MI} is going to be adaptively changed based on Eq. 16. It is obvious that due to the varying value of α_{MI}, each individual in the population attempts to launch the search in the different search space. In this experiment, we assess the effect of α_{MI} from two aspects. On one hand, compared against the method called "MIIGA+FixA" which uses the fixed value of α_{MI} to restrict the search space, we have a better understanding of the importance of dynamic update capability of α_{MI} in the proposed algorithm. In MIIGA+FixA, we set the value of α_{MI} to be 0.9 as [14] and [19] did. On the other hand, we need to figure out the necessity of the construction of Eq. 16 by comparing another approach named "MIIGA+RdA" in which the value of α_{MI} is altered stochastically during the evolutionary process. We present our results in Table 7.

From the table, we notice that the differences between MIIGA and MIIGA+FixA in all the measures, including the ABB (p-value is 1.5212e-12), the ART (p-value is 4.5043e-11), the AGB (p-value is 2.8954e-11), the AHD (p-value is 1.6579e-11) and the ABI (p-value is 2.7818e-07), are statistically significant. Despite that MIIGA+FixA takes less time to finish the execution under the same termination condition due to the lack of the update operation for α_{MI}, the structural difference between the final solution and the original network is much greater, compared with MIIGA. Hence, we can conclude that using varying value of α_{MI} provides the actual improvement on the effectiveness of BN structure learning. In fact, when the value of α_{MI} is predefined, the search space is also settled. It is very likely that some crucial edges are deleted permanently and many false connections are included in the final solution. Besides, because the search space stays the same during the evolutionary process, MIIGA+FixA has higher chance of being trapped in a local optimum, thus the best-so-far solution obtained by MIIGA+FixA appears much earlier than MIIGA in terms of the AGB value. While for MIIGA, it becomes evident that the search space can be modified dynamically by changing the value of α_{MI}. Thus, the individuals in the new generation are able to explore more possible structures in different search space to get higher chance of obtaining the global optimal solution.

When comparing MIIGA and MIIGA+RdA in Table 7, we find that the difference in the ABI statistics (p-value is 0.1223) is not statistically significant, which indicates that the best individuals having similar quality are obtained in the initial population due to the same initialization routine. While the ABB value of MIIGA is evidently higher than that of MIIGA+RdA (p-value is 0.0101) and the AHD value of MIIGA is significantly smaller than that of MIIGA+RdA (p-value is 1.5396e-07), which implies that MIIGA is able to find better solution that is closer to the original network, compared with MIIGA+RdA. Since Eq.16 guides the update operation of the binding parameter α_{MI}, it is more efficient and more accurate to change the search space in MIIGA, rather than randomly altering the value of

Table 7 Performance of modified MIIGA with different uses of α_{MI}

	ABB	ART	AGB	AHD	ABI
MIIGA	-1.1859e+04 (3.6043)	111.8929 (45.9301)	75.9000 (46.6087)	5.0667 (0.6915)	-1.2830e+04 (584.4365)
MIIGA+FixA	-1.2147e+04 (10.8180)	55.3026 (6.5850)	6.8333 (3.4749)	15.0333 (1.8659)	-1.2255e+04 (142.5399)
MIIGA+RdA	-1.1863e+04 (10.9366)	177.8257 (18.2734)	278.4000 (54.1553)	6.3667 (1.2726)	-1.2854e+04 (224.5912)

α_{MI} in the evolutionary process, making MIIGA+RdA more likely to lose the optimal solution.

4.4 Applicability analysis of MIIGA

It can be observed from Table 4 that MIIGA is un-able to obtain the best result when reconstructing the Child5 network which contains 100 nodes, defeated by IK2vMB. Therefore, we further conduct comparative experiments on several artificial networks in different scale to study the applicability of MIIGA.

For this experimental comparison we select 10 networks as the original BNs, and consider structure size from 37 up to 130 in increments of around 10 nodes, including the Alarm, Child3, Child5 networks used in [22] and other randomly-generated networks using toolbox of FullBNT-1.0.4 in Matlab. For each of these networks, we work with data sets sampled from them with 5000 instances. More detailed information about these networks is provided in Table 8.

Since MIIGA possesses an apparent superiority of the number of generations that are taken to find best-so-far solution over other competitors based on the AG-B statistics in Table 3 and Table 4, as performance indicator we adopt only one of the measures defined above, i.e., AHD, considering that this measure indicates the difference between the estimated structure and the original BN.

The AHD statistics for all experiments are summarized in Table 9. All results are the average of 30 tri-als and the best result is in bold. It can be seen that compared against other algorithms, MIIGA can always discover the best structures with less structural difference with respect to the original BNs containing less than 100 nodes. However, for the networks in large scale, such as BN110, BN120 and BN130, in terms of the accuracy of the learned model, the performance of MMHC is better than that of MIIGA, but with relatively small differences. Furthermore, when we consider the ratio of true arcs learnt, which is called coverage, we notice that apart from K2, any other algorithm tends to be more ineffective as the number of nodes increases.

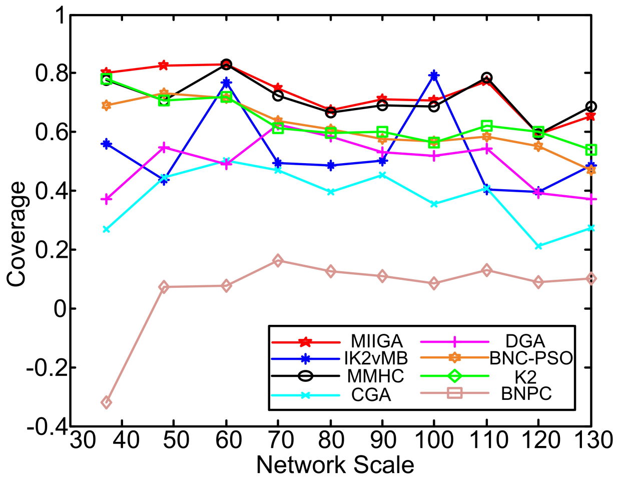

Fig. 8 Coverage of learning ten artificial networks for MIIGA and other existing methods

Fig. 8 shows the ratio of true arcs learnt employing MIIGA and other existing methods for ten data sets, and in order to clarify the relation between the results of all networks in different scale for each algorithm, results from the same method are linked up.

According to the AHD statistics in Table 9, MIIGA is better at recovering causal relations for networks between 30 and 90 nodes in virtue of the preprocessing in which it tries to identify the connections with highest possibility for each node and the dynamic constrained search space in case of being stuck in a local optimum. Nevertheless, as shown in Fig. 8, the ratio of true arcs learnt using MIIGA decreases with the increasing number of nodes, that is, the accuracy of the estimated model becomes lower when we deal with large scale networks. This result is justified because as the number of nodes increases, even though some invalid structures can be eliminated to constrain the search space with the help of MI, the scale of the search space is still enormous contrary to the fixed size of the population used in the evolutionary process. Besides, in view of the available memory, 5000 cases are provided to train BNs in different scale, so there is a strong possibility that the sample size is not big enough to offer sufficient in-

Table 8 Detailed information about artificial networks

Network	Size	No.nodes	No.edges	Max In/Out -degree
Alarm	5000	37	46	4/5
BN48	5000	48	68	3/5
Child3	5000	60	79	3/7
BN70	5000	70	96	3/8
BN80	5000	80	104	3/8
BN90	5000	90	125	3/7
Child5	5000	100	126	2/7
BN110	5000	110	150	3/10
BN120	5000	120	161	3/7
BN130	5000	130	202	3/8

Table 9 Comparison of the proposed and other existing methods in terms of AHD

Network	MIIGA	IK2vMB	MMHC	CGA	DGA	BNC-PSO	K2	BNPC
Alarm	**9.0667**	20.2000	10.2667	33.6000	28.8667	14.2333	60.7000	10
BN48	**11.9667**	38.3667	19.9667	37.6000	30.8000	18.3333	63.1333	20
Child3	**13.4667**	18.3333	13.5333	39.2800	40.3000	22.5667	72.9667	22
BN70	**24.2667**	48.6000	26.4000	50.8000	36.0667	34.9333	80.2667	37
BN80	**33.8333**	53.4000	34.6333	62.7667	43.0588	40.8000	90.7000	42
BN90	**36.1000**	62.2000	38.8667	68.3000	58.6000	53.1000	111.5000	50
Child5	36.8000	**26.2000**	39.3333	81.1600	60.5333	54.4074	115.4000	55
BN110	34.1667	89.4333	**32.2667**	88.8000	68.3333	62.5667	130.6000	57
BN120	65.8667	96.9333	**65.6333**	126.8000	97.6000	71.9333	146.4333	64
BN130	69.7333	104.0333	**63.2333**	146.6667	127.2000	107.1000	181.7667	93

formation about the original networks when we tackle with large scale BNs.

It may be possible to develop method to further increase the applicability of MIIGA, making it scale up to BNs with hundreds of nodes. In particular, authors in [37] have recently proposed a new BN structure learning algorithm called SAR (Separation And Reunion), which decomposes the task of constructing a large BN into learning some relatively small networks, and the final structure can be obtained by recombining these small BNs. Motivated by this kind of study, we plan to adopt the decomposition as the first step before using MIIGA to implement the task of learning large BNs, so as to improve the accuracy of structure estimation, which is discussed in an uncompleted paper.

5 Conclusion

In this paper, a hybrid algorithm called MIIGA for structure learning of BNs is presented based on an improved GA and information theory. We first design a new encoding scheme, where an individual consists of three parts: a node ordering, an upper triangular adjacent matrix and a binding parameter. The value of the parameter is subjected to be adaptively changed during the evolutionary process to guarantee that the search is able to traverse the varying search space, increasing the possibility of obtaining better alternatives. Besides, thanks to the permutation of variables, there is no need to prevent or repair any cycles generated in the child networks as the iteration proceeds, which can decrease the execution time. Then, the MI is used to filter the useless networks and improve the quality of the individuals, thus the convergence performance of the proposed algorithm is significantly better than the methods using randomly initial population. Moreover, we develop the PR operator to obtain better individuals as the offspring so that the superior genes are able to be passed on from generation to generation more efficiently without the disruption of the excellent substructures, compared with the conventional crossover. This strategy has two advantages. On one hand, more different equivalence classes are explored because the PR operator can produce the offspring that belong to the different equivalence classes from the parental population. On the other hand, owning to the reuse of the BIC scores calculated in the previous generations, fewer BIC metric evaluations are invoked. The proposed algorithm is tested on the four standard BN structures, and the experimental results demonstrate that our algorithm tends to work better in terms of the quality of the final solutions, the convergence speed and the running time, compared against other three existing population-based methods. Our future work concentrates on applying our proposed algorithm to the real-life problems, as well as large-scale BN structure learning problems.

Acknowledgements This study was funded by the National Natural Science Foundation of China (61562018), the International S&T Cooperation Projects of China (2015DFR10510), the Natural Science Foundation of Hainan Province, China (20166209) and the Research Projects of Hainan Institutes of Higher Education, China (HNKY2014-04).

Conflict of Interest: The authors declare that we have no conflict of interest.

References

1. Yue K, Wu H, Fu XD, Xu J, Yin ZD, Liu WY (2017) A data-intensive approach for discovering user similarities in social behavioral interactions based on the Bayesian network. Neurocomputing 219:364-375.
2. Ramírez-Noriega A, Juárez-Ramíez R, Martínez-Ramírez Y (2017) Evaluation module based on Bayesian networks to Intelligent Tutoring Systems. International Journal of Information Management 37(1):1488-1498.
3. Larrañaga P, Karshenas H, Bielza C, Santana R (2013) A review on evolutionary algorithms in Bayesian network learning and inference tasks. Information Sciences 233:109-125.
4. Chickering DM, Heckerman D, Meek C (2004) Large-sample learning of Bayesian networks is NP-hard. Journal of Machine Learning Research 5(10):1287-1330.
5. Kalisch M, Bühlmann P (2016) Estimating high-dimensional directed acyclic graphs with the PC-algorithm. Journal of Machine Learning Research 8(3):613-636.
6. Villanueva E, Maciel CD (2014) Efficient methods for learning Bayesian network super-structures. Neurocomputing 123:3-12.
7. Spirtes P, Glymour C (1991) An algorithm for fast recovery of sparse causal graphs. Social Science Computer Review 9(1):62-72.
8. Yuan C, Malone B (2013) Learning Optimal Bayesian Networks: A Shortest Path Perspective. Journal of Artificial Intelligence Research 48(10):23-65.
9. O'Gorman B, Babbush R, Perdomo-Ortiz A, Aspuru-Guzik A, Smelyanskiy V (2015) Bayesian network structure learning using quantum annealing. The European Physical Journal Special Topics 224(1):163-188.
10. Gheisari S, Meybodi MR (2016) BNC-PSO: structure learning of Bayesian networks by particle swarm optimization. Information Sciences 348:272-289.
11. Wong ML, Leung KS (2004) An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach. IEEE Trans Evol Comput 8(4):378-404.
12. Lee J, Chung W, Kim E (2008) Structure learning of Bayesian networks using dual genetic algorithm. IEICE Trans on Information and Systems 91(1):32-43.
13. Lee J, Chung W, Kim E, Kim S (2010) A new genetic approach for structure learning of Bayesian networks: Matrix genetic algorithm. International Journal of Control, Automation and Systems 8(2):398-407.
14. Li BH,Liu SY,Li ZG (2012) Improved algorithm based on mutual information for learning Bayesian network structures in the space of equivalence classes. Multimedia Tools and Applications 60(1):129-137.
15. Teyssier M, Koller D (2005) Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In: Proceedings of 21st conference on uncertainty in artificial intelligence, pp 584-590.
16. dos Santos EB, Hruschka Jr ER, Ebecken NFF (2010) Evolutionary algorithm using random multi-point crossover operator for learning Bayesian network structures. In: Proceedings of 9th international conference on machine learning and applications, pp 430-435.
17. dos Santos EB, Hruschka Jr ER, Hruschka ER, Ebecken NFF (2010) A distance-based mutation operator for learning bayesian network structures using evolutionary algorithms. In: Proceedings of IEEE congress on evolutionary computation, pp 1-8.
18. Jiang J, Wang J, Yu H, Xu H (2013) Poison Identification Based on Bayesian Network: A Novel Improvement on K2 Algorithm via Markov Blanket. In: Proceedings of 4th international conference in swarm intelligence, pp 173-182.
19. Chen XW, Anantha G, Lin X (2008) Improving Bayesian network structure learning with mutual information-based node ordering in the K2 algorithm. IEEE Trans Knowl Data Eng 20(5):628-640.
20. Ji JZ, Zhang HX, Hu RB, Liu CN (2009) A Bayesian network learning algorithm based on independence test and ant colony optimization. Acta Automatica Sinica 35(3):281-288.
21. Masegosa AR, Moral S (2013) New skeleton-based approaches for Bayesian structure learning of Bayesian networks. Applied Soft Computing 13(2):1110-1120.
22. Tsamardinos I, Brown LE, Aliferis CF (2006) The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning 65(1):31-78.
23. Kojima K, Perrier E, Imoto S, Miyano S (2010) Optimal search on clustered structural constraint for learning Bayesian network structure. Journal of Machine Learning Research 11(1):285-310.
24. Larrañaga P, Poza M, Yurramendi Y, Murge RH, Kuijpers CMH (1996) Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. IEEE Trans Pattern Anal Mach Intell 18(9):912-926.
25. Faulkner E (2007) K2GA: heuristically guided evolution of Bayesian network structures from data. In: Proceedings of 1st IEEE symposium on computational intelligence and data mining, pp 18-25.
26. Liu K, Ng JKY, Lee VCS, Son SH, Stojmenovic I (2016) Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network. IEEE/ACM Trans Networking 24(3):1759-1773.
27. Robinson RW (1977) Counting unlabeled acyclic digraphs. In: Little CHC (ed) Lecture Notes in Mathematics: Combinatorial Mathematics V. Springer, Berlin, pp 28-43.
28. de Campos LM, Castellano JG (2007) Bayesian network learning algorithms using structural restrictions. International Journal of Approximate Reasoning 45(2):233-254.
29. Cooper GF,Herskovits E (1992) A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9(4):309-347.
30. Friedman N, Koller D (2003) Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks, Machine Learning 50(1):95-125.
31. Bac FQ, Perov VL (1993) New evolutionary genetic algorithms for NP-complete combinatorial optimization problems. Biological Cybernetics 69(3):229-234.
32. Zhang H, Cao X, Ho JKL, Chow TWS (2017) Object-Level Video Advertising: An Optimization Framework, IEEE Trans Ind Informat 69(3):229-234.

33. Chickering DM (1995) A transformational characterization of equivalent Bayesian network structures. In: Proceedings of 11th conference on uncertainty in artificial intelligence, pp 87-98.
34. Cheng J, Greiner R, Kelly J, Bell D, Liu W (2002) Learning Bayesian networks from data: An information-theory based approach. Artificial Intelligence 137(1-2):43-90.
35. Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society. Series B (Methodological) 50(2):157-224.
36. Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics 18(1):50-60.
37. Liu H, Zhou S, Lam W, Guan J (2017) A new hybrid method for learning bayesian networks: Separation and reunion. Knowledge-Based Systems 121:185-197.