Received: 29 September 2021 | Accepted: 18 January 2022

DOI: 10.1002/int.22833

RESEARCH ARTICLE

WILEY

Bayesian network structure learning with improved genetic algorithm

Baodan Sun | **Yun Zhou**

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China

Correspondence
Yun Zhou, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China.
Email: zhouyun@nudt.edu.cn

Funding information
Training Program for Excellent Young Innovators of Changsha, Grant/Award Number: KQ2009009; Huxiang Youth Talent Support Program, Grant/Award Number: 2021RC3076; National Natural Science Foundation of China, Grant/Award Number: 61703416

Abstract

As an important model of machine learning, Bayesian networks (BNs) have received a lot of attentions since they can be used for classification via probabilistic inference. However, since it is a complicated combination optimization problem, BN structure learning cannot be solved with classic convex optimization algorithms. Hence, evolutionary algorithms provide an alternative way to find a global solution to BN structure learning problem. In this paper, we improve the biased random-key genetic algorithm to solve the BN structure learning problem. Meanwhile, we apply a local optimization model as its decoder to improve the performance of the proposed algorithm. Finally, we conduct our experiments on nine benchmark networks and a real dataset of cross-site scripting (XSS) attack. Experimental results show that the proposed algorithm can obtain more accurate solutions than other state-of-the-art algorithms and achieve a good performance in XSS attack detection for web security.

KEYWORDS

Bayesian networks, biased random keys, genetic algorithms, structure learning

1 | INTRODUCTION

Bayesian network (BN) is a generative model but can also be used for classification. BNs are consisted of structure and parameters, which means that the learning algorithms can be partitioned into structure learning and parameter learning. Structure learning is the basis of

subsequent parameter learning and inference, which is important for BN learning algorithms. The structure of BNs is represented by directed acyclic graphs (DAGs), which can be learnt by expert knowledge and data learning. However, constructing the structure with expert knowledge is inefficient and human resources. Hence, in this paper, we will mainly introduce the method of constructing the structure of BN with data learning.

1.1 | Related works

In the past 11 years, many researchers have been working on BN structure learning and some progress has achieved. Grüttemeier et al.[1] proposed a BN structure learning algorithm with structural constraints. Gu et al.[2] learnt large-scale BNs (>50 nodes) by partitioning nodes into clusters to learn subgraphs separately and combining all these subgraph to obtain the final complete BN. de Campos et al.[3] used Bayesian Dirichlet score to learn exact BN structure and analyzed its property. Bartlett et al.[4] converted the BN structure learning problem into an integer linear programming problem and used cutting planes to solve it. Li et al.[5] incorporated comprehensive knowledge into the hybrid structure learning algorithm. Oyen et al.[6] learnt the structure of BNs via transfer learning. Yu et al.[7] proposed a deep generative model and learnt the DAG with structural constraint. Correia et al.[8] applied pruning techniques to score-based structure learning algorithms.

For more complicated application cases, such as large networks (>50 nodes), automatic causal discovery with data learning is more difficult but there are still some works researching in this field. Chu et al.[9] proposed causal time-varying dynamic Bayesian network (cTVDBN) to efficiently discover the structure of trajectory-based networks. Yu et al.[10] proposed causal discovery from streaming features (CDFSF) and its variant–S-CDFSF and conducted experiments on some large networks, such as Gene (801 nodes). Zhou et al.[11] proposed a causal discovery algorithm to discover causal rules in large databases. Hong et al.[12] proposed the LSCD framework, which uses a graph-partitioning method and infer the causality of each subdataset. Ma et al.[13] proposed two efficient algorithms to mine the combined causes from large data sets.

However, these works require rigorous mathematical derivation and might not completely apply to practical scenes. Thus, heuristic algorithms provide an alternative way to solve this problem. Previous works have applied different heuristic algorithms for BN structure learning. Table 1 lists three types of heuristic algorithms that can be used to learn BN structure, including swarm intelligence, GA, and others.

Swarm intelligence is inspired by studying behaviors of birds and bees. In the nature, social animals work collectively in a unified dynamic system, their performances in solving problems and making decisions will surpass most individual members. As shown in Table 1, swarm intelligence is frequently used for solving Bayesian network structure learning (BNSL) problem. Khanteymoori et al.[14] proposed a new BNSL algorithm based on Breeding Swarm algorithm, which combines GA with PSO. Ji et al.[15] compared three swarm intelligence algorithms including ABC-B,[16] BFO-B,[17] and ACO-B[18] applying to BNSL. Experiments show that ABC-B achieves better K2 score than the other algorithms, and BFO-B can obtain more reliable structures, and ACO-B can not achieve better performance than ABC-B and BFO-B. Zhang et al.[19] proposed to combine PSO algorithm with the Chao theory to search the ordering of nodes for the K2 algorithm. Wang et al.[20] proposed the NDPSO-BN algorithm, which includes the mutation and neighbor searching operators in PSO. Gheisari et al.[21] proposed BNC-PSO

TABLE 1 Different heuristic algorithms solving Bayesian network structure learning

Heuristic types	References	Algorithm	Technical toutine
Swarm Intelligence	Gheisari et al.[21]	BNC-PSO	PSO+GA
	Khanteymoori et al.[14]	Breeding Swarm	PSO+GA
	Ji et al.[16]	ABC-B	ABC
	Yang et al.[17]	BFO-B	BFO
	de Campos et al.[18]	ACO-B	ACO
	Wang et al.[20]	NDPSO-BN	Neighbor searching+PSO
	Zhang et al.[19]	Chaotic-PSO	Chao theory+PSO
GA	Fukuda et al.[22]	PBIL	GA
	Lee et al.[23]	MGA	GA
	Khanteymoori et al.[25]	ARO	ARO
	Carvalho[26]	CCGA	GA
	Shetty et al.[27]	SGA	GA
Others	Constantinou[29]	Saiyan	Associational heuristic
	Baioletti et al.[28]	DEBN	Algebraic differential evolution

algorithm, which combines crossover operator and mutation operator into PSO. Yang et al.[17] proposed BFO-B algorithm, which apply BFO to search the optimal BN structure.

Genetic algorithm (GA) searches for the optimal solution by simulating the natural evolution process, which converts the problem-solving process into the crossover and mutation of chromosomal genes in biological evolution. Researchers also apply GA and its extensions to solve BNSL problem. Fukuda et al.[22] proposed to apply Population-based Incremental Learning (PBIL) to BNSL, which can evolve a probability vector. Lee et al.[23] proposed MGA algorithm, which represents each individual in the population into a matrix. Vafaee[24] proposed a hybrid algorithm, which combines mutual dependencies to reduce the searching space with GA to search the remaining solution space. Khanteymoori et al.[25] proposed ARO algorithm and the mechanism of ARO is based on asexual reproduction optimization. Carvalho[26] proposed a cooperative coevolutionary genetic algorithm (CCGA) to solve BNSL problem, which can be divided into two parts: determining the ordering of nodes and their corresponding matrix. Shetty et al.[27] used semantic genetic algorithm (SGA) to learn the structure of BNs.

In recent years, there are also other evolutionary algorithms applied to BNSL. Baioletti et al.[28] used a novel evolutionary algorithm, DEBN, to learn the structure of BNs, which applied Algebraic Differential Evolution to solve BNSL problem. Constantinou[29] used an associational heuristic algorithm, Saiyan, to learn the structure of BNs.

1.2 | Motivations and contributions

In this paper, we utilized an improved GA called biased random-key genetic algorithm (BRKGA)[30] to solve the BNSL problem, which is a general framework used to solve combination optimization problems. BRKGA contains a decoder to decode the random keys into the

initial solutions to BNSL problem. In this paper, our improvement is exploiting NOTEARS algorithm as the decoder, which can obtain a local optimal solution to BNSL problem to improve the performance of the proposed algorithm. In this way, we can obtain the initial solutions to BNSL problem and improve the random generated initial solutions, which can improve the final performance of GAs effectively. To deal with the cycles appearing in the crossover and mutation operations, we use the property of adjacency matrix to ensure the acyclicity of DAGs.

Our contributions of this paper are: (1) we introduce biased random-key genetic algorithm into BN structure learning; (2) we use NOTEARS algorithm as both the decoder and local optimizer to solve BNSL problem; (3) we apply our algorithm into the practical cross-site scripting (XSS) attack detection to verify the validity of our algorithm.

This paper is an extended version of the proceeding paper[31] with additional experiments and discussions. To be specific, we extend our comparing experiments with other heuristic algorithms with more data and evaluation metrics and add the practical application of our proposed method in XSS attack detection.

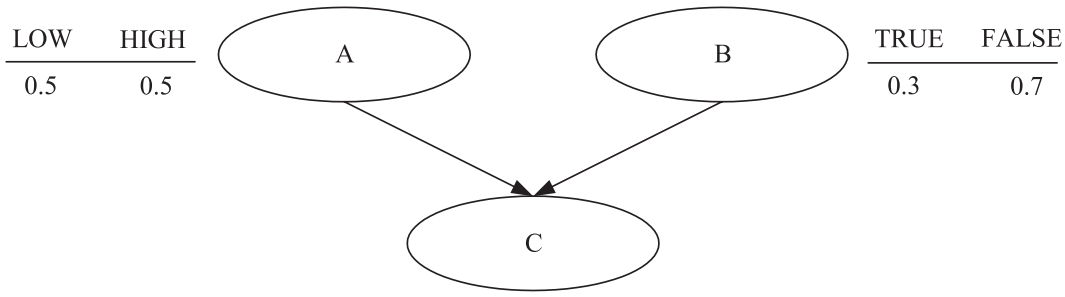
The paper is organized as follows. In Section 2, we introduce the basic knowledge of BNs and BRKGA. In Section 3, we introduce our improvements for using BRKGA to solve BNSL, especially for the decoder designed to relate random generated solutions to BNSL problem. Section 4 show the experiment results of the proposed algorithm and two heuristic algorithms.[32,33] Section 5 concludes the whole paper and discusses potential future works.

2 | BACKGROUND

2.1 | Bayesian networks

As a kind of probabilistic graphical model, BNs encode probabilistic knowledge that can be used for reasoning the causal and conditional relationships between nodes in a graph, which is different from other decision models. BNs are consisting of two components: the network structure and conditional probability distribution (or conditional probability table when variables are discrete). In the above, we have mentioned that the structure of BNs is represented by DAGs, where each node in a DAG represents a variable and the arcs between two nodes represent their possible causal relationships. BN uses this graphic structure to specify a set of conditionally independent statements and numerical values of conditional probability to describe the conditional probability of dependent strength. Here is an example of a BN, where node A and node B are parents of node C, and their conditional probability tables are shown in Figure 1.

BNSL is the most important part of BN learning algorithms because both of the parameters and the structure are unknown in this process. The task of structure learning algorithms is to find the most fitted network structure with data in the samples. When the structure of BN is certain, parameter learning is a relatively simple parameter estimation problem. The accuracy of BNSL is depended on the learning target: knowledge discovery and density estimation. Knowledge discovery is to find out the dependency of correlated variables through studying the dependency between nodes in the learnt structure. These dependent relationships are reflected by edges in the graph, which means that we need to find the most accurate structure to recover the true network structure. The density estimation is to estimate the true distribution of statistical models, and it needs a good generalization of the learnt structure. Thus, we can predict posterior probabilities of new observed data.



A	B	YES	NO
LOW	TRUE	0.5	0.5
HIGH	TRUE	0.3	0.7
LOW	FALSE	0.1	0.9
HIGH	FALSE	0.15	0.85

FIGURE 1 An example of Bayesian networks

Usually, there are two ways to construct a BN: domain experts and data learning. In practice, we usually combine both of them, using expert knowledge to provide priors of the structure ensuring the accuracy of modeling and data learning to specify the complete DAG quickly. However, expert knowledge cannot be obtained easily and the accuracy of expert knowledge should also be considered. Hence, data learning appears to be particularly important when we are stuck in this dilemma. There are three approaches to learn DAGs from data: score-based approach, constraint-based approach and hybrid approach. The score-based approach contains scoring functions and searching algorithms. Searching algorithms take an important role in the score-based approach because BNSL is a NP-hard problem[34] so that we cannot score all the possible structures and compare them. In practice, we usually use searching algorithms to search in the possible topological structure space according to the scoring functions to reduce the computation complexity. One of the widespread searching algorithms is heuristic searching algorithms, which is an effective way to obtain a global best solution quickly. However, heuristic algorithms may suffer from a low accuracy since its mechanism is a totally random searching process. So in this paper, we propose our algorithm to combine a local optimization process with a GA to improve the performance of our method.

In this paper, we will first define the mathematic representations of BNs. Given a DAG $G = \langle V, E \rangle$, V represents the set of nodes in the DAG and E represents the set of edges between these nodes. The nodes in V are corresponding to random variables, $X = \{X_1, X_2, ..., X_n\}$ and $n = |V|$ is the number of nodes. According to the implicit independence hypothesis in the structure, given its parents nodes, X_i is conditionally independent of its non-children nodes, and the joint probability distribution can be decomposed into the product of multiple conditional probability distributions:

$$P(X_1, X_2, ..., X_n) = \prod_{j=1}^{p} P(X_j \mid Pa(X_j)) \tag{1}$$

where, $P(X_j \mid Pa(X_j))$ represents the conditional distribution for X_j given its parents $Pa(X_j)$ with directed edges from each node in $Pa(X_j)$ to X_j in G.

The structure of a BN represents the conditional independence property between any two nodes in the graph and the nodes in DAGs represent random variables. In the traditional methods, an adjacency matrix $A = \{a_{ij}\}^{n*n}$ is usually used to represent a DAG. If there is an edge from i to j, the corresponding a_{ij} in the adjacency is set to 1, and a_{ji} is set to 0. It should be noticed that the elements in the diagonal are 0 since it is acyclic. The important property of the adjacency matrix is that elements in the diagonal of the power function of the adjacency matrix should be 0. We will use the property to check and modify the final results.

In traditional methods, scoring functions are used to evaluate the learnt structure of BN, and famous scoring functions include MDL, BDeu, BIC, and so on. In this paper, we use the loss function proposed in NOTEARS algorithm to evaluate all the possible solutions and the details can be seen in the next section.

2.2 | Biased random-key genetic algorithm

GA was designed and proposed according to the evolutionary laws of organisms in the nature. GA is a computational model that simulates the biological evolution process of natural selection and genetic mechanism of Darwin's biological evolution theory. It is a method to search for the optimal solution by simulating the natural evolution process. GA transforms the problem-solving process into a process similar to the crossover and mutation of chromosomal genes in biological evolution by means of mathematics and computer simulation operations. When solving more complex combinatorial optimization problems, compared with some conventional optimization algorithms, better optimization results can usually be obtained faster.

BRKGA is an improved random-key genetic algorithm (RKGA), which was firstly introduced by Bean[35] and it is designed for solving combination optimization problem. RKGA and BRKGA contain mutation operator and cross operator the same as GA, but RKGA and BRKGA add a block called decoder to convert random keys into initial solutions to optimization problem. Random keys are consisting of real numbers in the interval [0, 1] and its length is the same as the final solution. And the decoder can be designed according to different optimization problems, which can be very simple or act as a local optimizer. When choosing individuals to execute the crossover operator, BRKGA selects one parent from the elite individuals and the other from the whole population while RKGA randomly selects two individuals in the whole population. To explain this process with mathematical theory, the mechanism of the decoder is to map the solution from n-dimensional unit hypercube to the original solution space.

We can notice that BRKGA only improve the original GA slightly, but the performance can be changed a lot and the previous work[36] compares them in detail. However, we need to adjust the original BRKGA to utilize it to obtain the solution of the discrete optimization problem. To further improve the performance of the proposed algorithm, we use a local optimizer to obtain better initial solutions. And the flowchart of our improved algorithm is shown in Figure 2.

As we can see in Figure 2, the framework of BRKGA can be divided into two blocks: the problem-dependent block and the problem-independent block. The problem-dependent block is corresponding to the decoder, and the problem-independent block is corresponding to the other operations shown in Figure 2. Since BRKGA is a general framework, we can only design

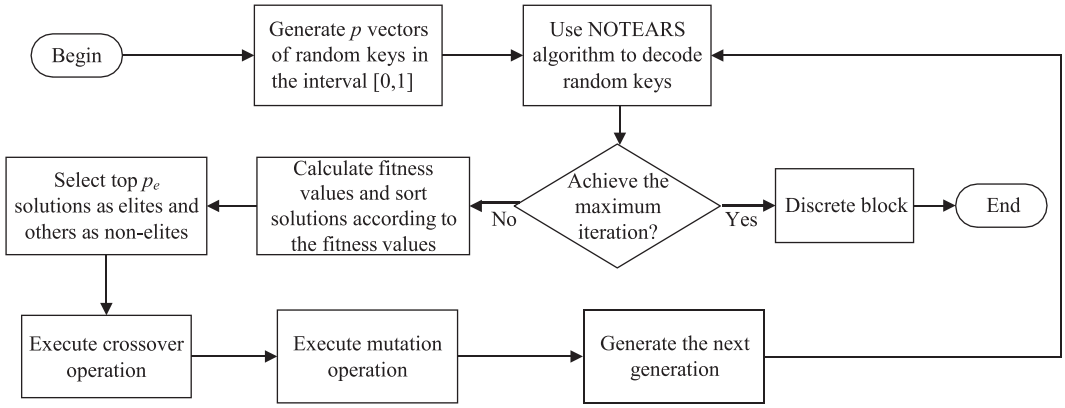


FIGURE 2 The flowchart of using BRKGA to solve the Bayesian network structure learning problem

the decoder according to the unsolved problem and the other operations can be kept the same. The most beneficial part of BRKGA is that we only need to adjust the decoder according to different combination problems and the other parts still can be used. NOTEARS algorithm is acted as both a decoder and local optimizer in the problem-dependent block of BRKGA. Since the original NOTEARS algorithms can obtain a local solution to BNSL problem, our algorithm can also be seen as the improvement of NOTEARS algorithm and the proposed algorithm can obtain a global solution compared to the local solution of NOTEARS. In this paper, we exploit NOTEARS algorithm as the decoder, which can obtain a local optimal solution to BNSL problem to improve the performance of the proposed algorithm.

3 | BN STRUCTURE LEARNING

3.1 | Problem definition

Many researchers solve BNSL problem with optimization algorithms. Among these algorithms, NOTEARS[37] has received widespread attentions because of its simple form, which can be easily understood.

In NOTEARS, it sets a real matrix $W = (w_{ij}) \in \mathbb{R}^{d\times d}$ as a collection of parameters satisfying $A(W) \in \{0,1\}^{d\times d}$ such that the elements in $A(W)$ are set to 1 if $w_{ij} \neq 0$ and 0 otherwise.[37] Thus, $A(W)$ is an adjacency matrix of a directed acyclic graph. In traditional approaches, the structure of BN is evaluated with scoring functions, which calculates the likelihood of the estimated DAG from data. A higher value of scoring functions means a better estimated structure. In NOTEARS, the loss function shown in Equation (2) is proposed to evaluate the estimated DAG rather than scoring functions, which can re-construct the BNSL problem as the equality constrained problem to simplify solving process. The loss function evaluates the difference between the estimated adjacency matrix and the ground truth, and the scoring functions counts the instances in the observed data for candidate structures. The BNSL problem can be written as[37]:

$$min\ \ell(W;X) \quad s.t.\ A(W) \in DAGs \tag{2}$$

where ℓ is the loss function and the constraint represents that $A(W)$ is the adjacency matrix of DAGs. This equation aims to find the minimum of the loss function defined by a real matrix W and its corresponding data X. The constraint illustrates that the adjacency matrix $A(W)$ converted from W is corresponding to a directed acyclic graph.

3.2 | Implementation

3.2.1 | Decoder

As mentioned above, the most important component of BRKGA is the decoder of random keys and it is designed according to the specific optimization problem. For example, Bean[35] uses the decoder to sort random keys and uses their sorted indices to represent a sequence. In this paper, we use NOTEARS algorithm[37] as the decoder since it can generate initial solutions and obtain a local minimum, simultaneously. Therefore, it is not only a decoder but also a local optimizer which can effectively improve the performance of GA.

In the above section, we have already shown the equality constrained problem (ECP)[37] of BNSL. In NOTEARS, the constraint in (2) can be replaced by matrix exponential as followings[37]:

$$min\ \ell(W;X) = \frac{1}{2n}\|\mathbf{X} - \mathbf{X}W\|^2 \quad s.t.\ \ h(W) = 0 \tag{3}$$

$$h(W) = tr(e^{W\circ W}) - d \tag{4}$$

where $\mathbf{X}$ is the observed data, and n is the sample size. $W \circ W$ represents the Hadamard product and $e^{W\circ W}$ represents the matrix exponential. $h(W) = 0$ guarantees the acyclicity of DAGs, which actually derivates from the property of the adjacency matrix: the n-th power of adjacency matrix a_{ij} means that there are n steps from i to j. Then, this optimization problem can be solved by augmented Lagrangian method with an augmented quadratic penalty[37]:

$$min\ \ell(W;X) + \frac{\rho}{2}|h(W)|^2 \quad s.t.\ h(W) = 0 \tag{5}$$

Then the dual problem can be written as[37]:

$$D(\alpha) = min\ L^{\rho}(W,\alpha) \tag{6}$$

$$min\ L^{\rho}(W,\alpha) = \ell(W;X) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{7}$$

$$\alpha \leftarrow \alpha + \rho h(W_{\alpha}^{*}) \tag{8}$$

where W_{α}^{*} is a local minimum, α is the Lagrange multiplier and ρ is the step size. W_{α}^{*} can be easily obtained by optimization algorithms, and L-BFGS[38] is used in NOTEARS. We should make it clear that all the equations above and its derivation process are cited from Zheng et al.[37] and the demonstration of NOTEARS and its further explanation can be seen in Wei et al.[39]

Although the above equations seem quite concise and can be easily understood, we should notice that the original BNSL is a combination optimization problem and its constraint is non-convex even if one can design a convex loss function. Actually, BNSL problem is a non-convex optimization problem and can not find the exact solution by the above process. Hence, BRKGA is utilized for solve the non-convex optimization problem and obtain a global solution to BNSL. Moreover, the above process can be seen as a decoder since NOTEARS can achieve a local minimum to improve the totally stochastic process here.

3.2.2 | Mutation and crossover

In GA, mutation and crossover operations are used to change the current solutions to search more feasible solutions. In this paper, we use Parameterized Uniform Crossover to execute crossover operation and the mechanism of mutation operator is the same as that of initial solutions. To execute the crossover operation, we should choose two parents to generate the new offspring. In BRKGA, the mechanism of choosing two parents is: we randomly select one parent in the elite population and the other in the non-elite population. Parameterized uniform crossover selects genes in the elite parent and the non-elite parent according to random numbers and reassembles genes of two parents into a new individual, which is shown in Figure 3.

Parameterized uniform crossover randomly generates a vector consisting of random numbers in the interval [0, 1]. Each random number in the vector is compared to a pre-defined parameter, ρ_e. If it is not smaller than ρ_e, the new generated individual inherits the corresponding element in the elite parent. Otherwise, the new generated individual inherits the corresponding element in the non-elite parent, and more details can be found in Figure 3. In this way, the new genome is generated and added into the next generation.

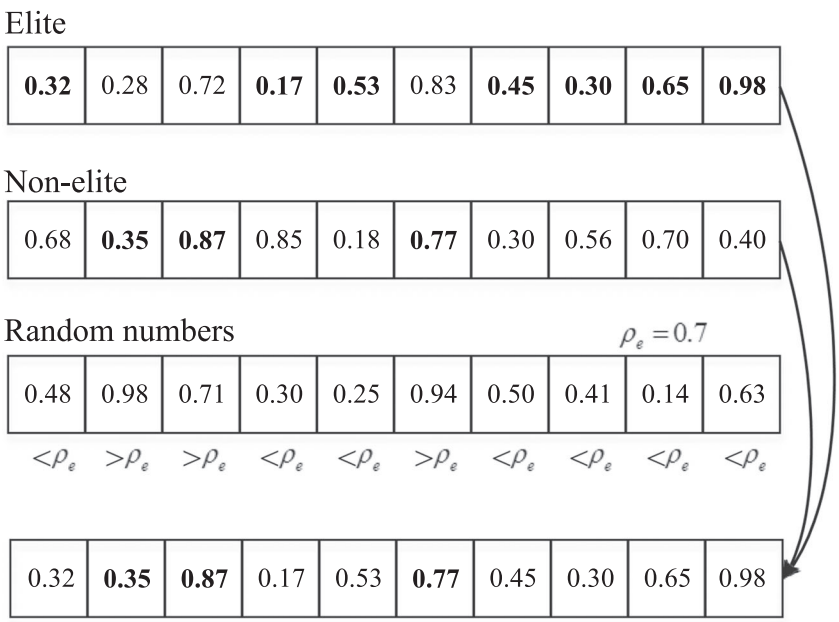


FIGURE 3 The idea of parameterized uniform crossover

The mechanism of mutation operation is the same as that of generating initial solutions at the start of BRKGA, which generates p_m vectors of n random numbers in the interval $[0, 1]$ and decodes them as initial solutions to the optimization problem. After executing the crossover and mutation operations, a new population is thus generated. The whole steps for BRKGA solving BNSL can be found in Algorithm 1. The first step is to generate p vectors of random keys of size n. Each random key is in the real interval $[0, 1]$ and random keys are the input of the decoder. The decoder interprets random keys as initial solutions to the optimization problem and the fitness values are then calculated using loss function defined in the above section. We sort the fitness values of the population and divide it into two parts: $p_e = 0.2*p$ portion of elite individuals and $p - p_e$ portion of non-elite individuals. To search the solution space, the mutation operator and the crossover operator are needed. The mechanism of mutation operator is mentioned above and p_m mutants are generated. The elite individuals are directly copied to the next generation and the crossover operator is executed. Next, these new individuals are combined together to form a new generation and their fitness values are re-calculated. The process will be executed iteratively until the algorithm reaches the maximum iteration, which is a pre-defined number (here is set to 100) by users. The convergence of our proposed method is the same as GA because we only use the decoder to generate initial solutions to BNSL problem and the overall mechanism is the same as the original GA.

Algorithm 1 BRKGA for Solving BNSL

1: **Input**: the population size p, maximum iteration max_iter, the elite population p_e, the mutant population p_m. 2: Initialize p vectors of random keys of size n.
3: Use the decoder to decode random keys into initial solutions to BNSL.
4: **repeat**
5: Calculate fitness values of all the solutions according to (3) and sort them.
6: Select $p_e = 0.2*p$ elite individuals and copy them to the next generation.
7: **for** $i = 1$ to $p - p_e - p_m$ **do**
8: Execute the crossover operator.
9: **end for**
10: **for** $p - p_m + 1$ to p **do**
11: Execute the mutation operator.
12: **end for**
13: **for** $p_e + 1$ to p **do**
14: Use the decoder to re-calculate local solutions to BNSL.
15: **end for**
16: **until** max_iter
17: Execute post-processing process.
18: **Output**: W

3.2.3 | Post-processing

In this paper, we use BRKGA to solve discrete BNSL problem. Therefore, we should add a discretization process to the original BRKGA to solve the BNSL problem. In this paper, after the execution of the local optimizer, the obtained adjacency matrix cannot represent a DAG, given the returned value in a matrix is continuous rather than discrete. First, it is because the returned result

of our algorithm is a matrix of continuous values, which are not 0 or 1. Since then, we need to convert these continuous values into discrete values. In BNSL, we usually define an indicator function to deal with this problem. In this paper, we pre-defined a threshold ω: if the elements in W_star are smaller than ω, the elements are set to 0; otherwise, the elements are set to 1. In this way, we can obtain the adjacency matrix of a BN. ω is chosen to be small enough that can preserve the sparsity of BNs. It should be noticed that we do not need to satisfy the acyclicity condition since the mutation operation and crossover operation will also induce cycles in the graph. So we will process the cycles contained in the final solution at the end of the proposed algorithm.

3.2.4 | Acyclicity

After the post-processing, the 0–1 matrix cannot ensure the acyclicity of the directed graph. Hence, we need to design an algorithm to delete the cycles contained in the graph. Here, to ensure the acyclicity of BNs, we use an important property of DAGs: the n-th power of adjacency matrix a_{ij} means that there are n steps from i to j. The first step for our algorithm is to calculate the trace of the adjacency matrix, A. If the trace of the adjacency matrix is not equal to zero, we randomly choose two nodes–V_x, V_y corresponding to the non-zero elements of the adjacency matrix and we delete the corresponding edges or reverse them. We repeat the recursive procedure until the trace of the adjacency is equal to zero. The details are shown in Algorithm 2.

Algorithm 2 Acyclicity of DAGs

1: **Input**: adjacency matrix, A.
2: **repeat**
3: Initialize $C = A$
4: **if** The diagonal of C does not contain non-zero elements **then**
5: $C = C^*A$
6: $i = i + 1$
7: **else**
8: Randomly choose two nodes V_x, V_y corresponding to the non-zero elements of C, delete the corresponding edges or reverse the edges in A.
9: **end if**
10: **until** $i = n$ and There is no non-zero elements in the diagonal of C.
11: **Output**: A

4 | EXPERIMENT

4.1 | Benchmark networks

4.1.1 | Experimental parameters

In our experiment, we implement our algorithm in R and execute the above algorithm on realistic networks downloaded from BN Repository*. The details for networks utilized in this paper are listed in Table 1. The sample sizes of these networks are 100, 500, 1000, 1500, 2000.

The population size p is set to 50, and the maximum iteration is 100. The sizes of elite individuals and mutant individuals are $p_m = 0.2*p$ and $p_e = 0.2*p$, respectively. ρ_e is set to 0.7. All the results are the averages of 10 times experiments. Structural hamming distance (SHD), TP, TN, FP, FN, F1, FPR, TPR, Precision, and Recall are used to compare the estimated DAG to the true graph and a smaller SHD means that two directed acyclic graphs are more similar. The definitions of these evaluation metrics can be seen in Constantinou.[40] All the parameters sensitivity analysis is given in the previous work.[36] The source code could be found in our homepage.[†]

4.1.2 | Experimental results

In this section, we list the experimental results of our algorithm conducting on these BNs in Table 1 with sample sizes of 100, 500, 1000, 1500, and 2000. We compare our algorithm to the original NOTEARS algorithm, the standard GA, and the mixture of PSO and GA algorithm. The reason why we choose GA and PSO-GA to conduct these experiments is that GA and PSO-GA are classic heuristic algorithms and often utilized in BNSL, which can be seen in Table 1. The details for NOTEARS are given in Section 3. For NOTEARS, we set the maximum iteration $t = 100$, tolerance $\epsilon = 1e - 8$, $c = 0.25$. The population size of the standard GA is 50, and the crossover probability is 0.9, and the mutation probability is 0.1. The crossover probability and mutation probability of PSO-GA is the same as the standard GA in this paper. Figure 4 shows an example of CHILD network to compare the experimental results with the original network. In the figure, the left network is the original CHILD network and the right network is our estimated result. The experimental results are shown in Tables 2–11. The smallest SHD values are presented in bold text.

4.1.3 | Analysis

Table 2 lists our experiment results on nine benchmark BNs, which are corresponding to small networks (<20 nodes), medium networks (20–50 nodes) and large networks (>50 nodes). We can see that SHD values increase with the arcs of these networks. Also, it is obvious that NOTEARS achieves better performance than the other three algorithms on CANCER network. On EARTHQUAKE network, BRKGA achieves better SHD value than the other three networks. On SURVEY network, GA obtains smaller SHD values than NOTEARS, PSO-GA, and BRKGA with sample sizes are set to 500. When the sample sizes are set to 100, 1000, 1500, and 2000, our proposed algorithm achieves smaller SHD values than the other three methods. On ASIA network, PSO-GA can obtain smaller SHD values than other algorithms when the samples are set to 100, 1500, 1500, and 2000. Our proposed algorithm achieves the smallest SHD value on SACHS network with sample size is set to 1500 and PSO-GA obtains the smallest SHD values with the other sample sizes on SACHS network. Our proposed method also can achieve better performance than the other three algorithms on CHILD and ALARM networks. Likewise, our proposed method can obtain smaller SHD values than other algorithms except that the sample size is set to 1500 on INSURANCE network. On HAILFINDER network, our proposed algorithm achieves the smallest SHD values with sample sizes are set to 100 and 2000. Meanwhile, NOTEARS achieve better performances than the other three algorithms with sample sizes are set to 500, 1000, and 1500.

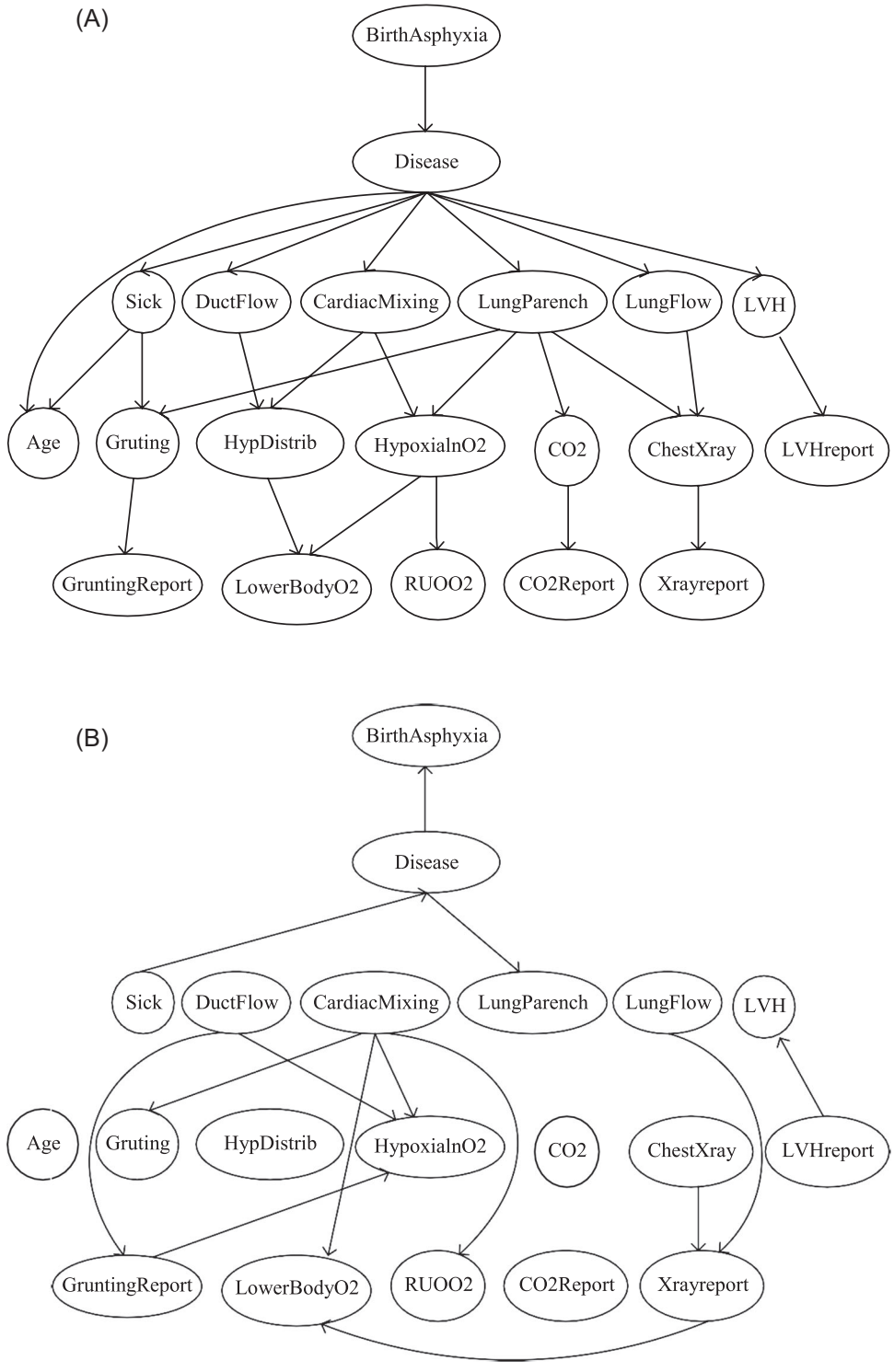


FIGURE 4 The learnt structure of CHILD network (A) the original CHILD network; (B) the CHILD network learnt with BRKGA

TABLE 2 The SHD values of four BN learning algorithms on nine standard networks

Network	Nodes	Arcs	Samples	NOTEARS	GA	PSO-GA	BRKGA
CANCER	5	4	100	**2.8**	7.1	7.0	5.5
			500	**3.3**	5.8	6.6	5.3
			1000	**3.1**	6.7	7.2	5.1
			1500	**3.1**	6.4	6.7	4.8
			2000	**2.8**	6.9	6.6	5.2
EARTHQUAKE	5	4	100	5.7	5.7	6.0	**4.8**
			500	5.7	6.3	5.8	**5.6**
			1000	5.8	6.2	5.6	**5.7**
			1500	5.7	5.6	5.7	**5.6**
			2000	6.0	5.8	6.4	**5.9**
SURVEY	6	6	100	9.4	7.9	8.3	**7.1**
			500	9.9	**6.7**	7.4	6.8
			1000	10.0	8.2	7.9	**6.9**
			1500	9.9	8.1	8.4	**7.4**
			2000	9.2	7.3	8.1	**7.2**
ASIA	8	8	100	12.3	11.3	**10.7**	10.9
			500	12.1	11.5	10.7	**10.5**
			1000	12.9	10.3	**10.1**	10.9
			1500	12.6	11.1	**10.8**	**10.8**
			2000	12.7	12.3	**10.5**	10.9
SACHS	11	17	100	28.0	21.9	**20.3**	20.6
			500	28.2	22.5	**19.3**	21.4
			1000	28.3	21.9	**19.9**	21.3
			1500	29.1	22.8	20.7	**19.6**
			2000	28.8	22.8	**20.1**	20.2
CHILD	20	25	100	38.2	39.5	40.7	**32.3**
			500	38.7	41.5	37.6	**31.4**
			1000	40.7	40.4	36.2	**31.9**
			1500	40.2	39.2	38.2	**31.1**
			2000	42.8	41.1	38.5	**30.6**
INSURANCE	27	52	100	79.5	74.9	67.7	**66.1**
			500	84.3	71.0	66.0	**65.9**
			1000	82.9	71.5	68.5	**66.2**
			1500	84.1	69.1	**64.9**	66.9

TABLE 2 (Continued)

Network	Nodes	Arcs	Samples	NOTEARS	GA	PSO-GA	BRKGA
			2000	86.7	71.3	67.3	**66.1**
ALARM	37	46	100	79.0	77.4	70.3	**62.7**
			500	83.6	75.8	71.0	**64.3**
			1000	84.5	75.2	72.6	**63.8**
			1500	85.1	75.6	71.3	**63.5**
			2000	83.6	75.2	72.0	**63.9**
HAILFINDER	56	66	100	96.6	114.6	113.1	**94.1**
			500	**97.9**	114.8	114.2	102.3
			1000	**98.5**	112.9	113.3	99.0
			1500	**96.9**	115.2	112.6	100.2
			2000	111.2	115.5	112.5	**99.6**

TABLE 3 The experimental results of four BN learning algorithms on CANCER network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	1.40	4.10	2.10	2.40	0.38	0.34	0.37	0.40	0.37
	NOTEARS	1.60	5.90	1.90	0.60	0.56	0.24	0.73	0.46	0.73
	GA	1.60	2.60	3.40	2.40	0.36	0.57	0.40	0.32	0.40
	PSO-GA	1.90	3.00	3.00	2.10	0.43	0.50	0.48	0.39	0.48
500	BRKGA	1.70	4.50	1.80	2.00	0.47	0.29	0.46	0.49	0.46
	NOTEARS	1.50	6.00	1.60	0.90	0.55	0.21	0.63	0.48	0.63
	GA	1.30	3.70	2.30	2.70	0.34	0.38	0.33	0.36	0.33
	PSO-GA	2.10	3.40	2.60	1.90	0.48	0.43	0.53	0.45	0.53
1000	BRKGA	1.70	4.50	2.00	1.80	0.47	0.31	0.49	0.46	0.49
	NOTEARS	1.50	6.00	1.50	1.00	0.55	0.20	0.60	0.50	0.60
	GA	1.20	2.90	3.10	2.80	0.29	0.52	0.30	0.28	0.30
	PSO-GA	1.50	2.80	3.20	2.50	0.34	0.53	0.38	0.32	0.38
1500	BRKGA	1.50	4.60	2.30	1.60	0.43	0.33	0.48	0.39	0.48
	NOTEARS	1.80	6.00	1.20	1.00	0.62	0.17	0.64	0.60	0.64
	GA	1.80	3.10	3.00	2.10	0.41	0.49	0.46	0.38	0.46
	PSO-GA	2.10	3.00	3.00	1.90	0.46	0.50	0.53	0.41	0.53
2000	BRKGA	1.50	4.60	1.80	2.10	0.43	0.28	0.42	0.45	0.42
	NOTEARS	1.40	6.00	1.70	0.90	0.52	0.22	0.61	0.45	0.61
	GA	2.00	3.10	2.90	2.00	0.45	0.48	0.50	0.41	0.50
	PSO-GA	2.10	3.40	2.60	1.90	0.48	0.43	0.53	0.45	0.53

TABLE 4 The experimental results of four BN learning algorithms on EARTHQUAKE network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	1.40	4.70	2.20	1.70	0.42	0.32	0.45	0.39	0.45
	NOTEARS	0.80	4.20	2.50	2.50	0.24	0.37	0.24	0.24	0.24
	GA	1.60	3.60	2.40	2.40	0.40	0.40	0.40	0.40	0.40
	PSO-GA	1.20	3.80	2.20	2.80	0.32	0.37	0.30	0.35	0.30
500	BRKGA	1.40	4.40	2.10	2.10	0.40	0.32	0.40	0.40	0.40
	NOTEARS	0.60	4.20	2.50	2.70	0.19	0.37	0.18	0.19	0.18
	GA	1.20	3.50	2.50	2.80	0.31	0.42	0.30	0.32	0.30
	PSO-GA	1.40	3.80	2.50	2.30	0.37	0.40	0.38	0.36	0.38
1000	BRKGA	1.40	4.20	2.20	2.20	0.39	0.34	0.39	0.39	0.39
	NOTEARS	0.60	4.10	2.50	2.80	0.18	0.38	0.18	0.19	0.18
	GA	1.80	3.20	2.80	2.20	0.42	0.47	0.45	0.39	0.45
	PSO-GA	1.50	3.90	2.20	2.40	0.39	0.36	0.38	0.41	0.38
1500	BRKGA	1.10	4.30	2.50	2.10	0.32	0.37	0.34	0.31	0.34
	NOTEARS	0.40	4.10	2.60	2.90	0.13	0.39	0.12	0.13	0.12
	GA	1.80	3.70	2.30	2.20	0.44	0.38	0.45	0.44	0.45
	PSO-GA	1.10	4.00	2.30	2.60	0.31	0.37	0.30	0.32	0.30
2000	BRKGA	1.00	4.10	2.50	2.40	0.29	0.38	0.29	0.29	0.29
	NOTEARS	0.50	4.00	2.50	3.00	0.15	0.38	0.14	0.17	0.14
	GA	1.50	3.70	2.30	2.50	0.38	0.38	0.38	0.39	0.38
	PSO-GA	1.30	3.40	2.70	2.60	0.33	0.44	0.33	0.33	0.33

When the networks are small, we notice that NOTEARS can achieve better or similar SHD values than our algorithm. For smaller test models, NOTEARS has a chance to obtain better results than our proposed method, but when the networks are larger, our proposed method can also achieve better results. However, with the increase of nodes, NOTEARS is tended to more and more difficult to find a better solution than our algorithm on CHILD and ALARM networks. We must point out that our algorithm is designed to get a continuous solution and we discretize it through an indicator function. So even if the best solution obtained by our algorithm achieves a smaller loss value, their adjacency matrices may be the same.

For Table 3, we can notice that the recall of NOTEARS algorithm is better than our proposed algorithm and the precision of NOTEARS is higher than our proposed method except when the sample size are set to 500. We can also see GA and PSO-GA can achieve a good performance on CANCER and EARTHQUAKE network considering these evaluation metrics shown in Tables 3 and 4. For CANCER and EARTHQUAKE networks, the network size is small, which is easily for GA and PSO-GA conducting the random searching in the solution space. However, when the network size is increasing, we can notice that the performances of GA and PSO-GA become worse for the reason that the solution space become very large and it is very difficult to search for the global solution. For Tables 4–6 and 10, we can see that the

TABLE 5 The experimental results of four BN learning algorithms on SURVEY network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	1.10	7.60	2.20	4.10	0.26	0.22	0.21	0.33	0.21
	NOTEARS	0.50	5.30	4.90	4.30	0.10	0.48	0.10	0.09	0.10
	GA	2.10	5.70	3.30	3.90	0.37	0.37	0.35	0.39	0.35
	PSO-GA	1.30	6.60	2.80	4.30	0.27	0.30	0.23	0.32	0.23
500	BRKGA	1.00	7.70	2.00	4.30	0.24	0.21	0.19	0.33	0.19
	NOTEARS	0.30	4.90	5.00	4.80	0.06	0.51	0.06	0.06	0.06
	GA	2.70	6.60	2.40	3.30	0.49	0.27	0.45	0.53	0.45
	PSO-GA	2.10	7.00	2.80	3.10	0.42	0.29	0.40	0.43	0.40
1000	BRKGA	1.20	7.60	1.80	4.40	0.28	0.19	0.21	0.40	0.21
	NOTEARS	0.50	4.90	5.20	4.40	0.09	0.51	0.10	0.09	0.10
	GA	2.40	5.60	3.40	3.60	0.41	0.38	0.40	0.41	0.40
	PSO-GA	2.20	6.30	3.00	3.50	0.40	0.32	0.39	0.42	0.39
1500	BRKGA	0.60	7.40	2.50	4.50	0.15	0.25	0.12	0.19	0.12
	NOTEARS	0.30	5.10	5.20	4.40	0.06	0.50	0.06	0.05	0.06
	GA	2.40	5.90	3.10	3.60	0.42	0.34	0.40	0.44	0.40
	PSO-GA	1.10	6.20	3.30	4.40	0.22	0.35	0.20	0.25	0.20
2000	BRKGA	0.90	7.30	2.50	4.30	0.21	0.26	0.17	0.26	0.17
	NOTEARS	0.50	5.70	4.30	4.50	0.10	0.43	0.10	0.10	0.10
	GA	1.80	6.90	2.10	4.20	0.36	0.23	0.30	0.46	0.30
	PSO-GA	1.90	6.20	3.00	3.90	0.36	0.33	0.33	0.39	0.33

overall performance of our proposed method is outstanding than NOTEARS. For Table 7, the TP of NOTEARS is larger than our proposed method when the sample is set to 1500, and the overall performance of our poposed method is still better than NOTEARS in the other situations. For Table 8, the TP of our proposed method is smaller than NOTEARS when the sample size is set to 100 and the overall performance of our poposed method is still better than NOTEARS in the other situations. For Table 9, the overall performance of our proposed method is better than NOTEARS when the sample size is set to 500 and 1500. For Table 11, the overall performance of our proposed method is better than NOTEARS when the sample size is set to 100, 500, and 2000.

4.2 | Real dataset

In this section, we apply our algorithm to detect XSS attack in cyber space to test its accuracy in the practical application. XSS is a common web application vulnerability. XSS attack is not easily detected since it is a kind of passive attacks utilized on the client side. The attacker inserts malicious payload (script code) into the web page. When the user browses the page, the

TABLE 6 The experimental results of four BN learning algorithms on ASIA network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	1.30	16.20	5.80	4.70	0.20	0.26	0.22	0.18	0.22
	NOTEARS	0.70	15.20	6.20	5.90	0.10	0.29	0.11	0.10	0.11
	GA	2.10	14.90	5.10	5.90	0.28	0.26	0.26	0.29	0.26
	PSO-GA	1.80	16.10	3.90	6.20	0.26	0.20	0.23	0.32	0.23
500	BRKGA	1.90	15.90	5.70	4.50	0.27	0.26	0.30	0.25	0.30
	NOTEARS	0.90	15.30	6.10	5.70	0.13	0.29	0.14	0.13	0.14
	GA	1.60	15.50	4.50	6.40	0.23	0.23	0.20	0.26	0.20
	PSO-GA	1.80	16.50	3.60	6.10	0.27	0.18	0.23	0.33	0.23
1000	BRKGA	2.10	16.00	5.10	4.80	0.30	0.24	0.30	0.29	0.30
	NOTEARS	0.70	14.60	6.50	6.20	0.10	0.31	0.10	0.10	0.10
	GA	2.40	16.10	3.90	5.60	0.34	0.20	0.30	0.38	0.30
	PSO-GA	1.90	16.60	3.50	6.00	0.29	0.17	0.24	0.35	0.24
1500	BRKGA	2.30	15.80	5.20	4.70	0.32	0.25	0.33	0.31	0.33
	NOTEARS	0.70	14.80	6.40	6.10	0.10	0.30	0.10	0.10	0.10
	GA	1.80	15.50	4.50	6.20	0.25	0.23	0.23	0.29	0.23
	PSO-GA	1.50	16.40	3.80	6.30	0.23	0.19	0.19	0.28	0.19
2000	BRKGA	1.70	15.60	5.80	4.90	0.24	0.27	0.26	0.23	0.26
	NOTEARS	0.90	14.60	6.30	6.20	0.13	0.30	0.13	0.13	0.13
	GA	1.50	14.70	5.30	6.50	0.20	0.27	0.19	0.22	0.19
	PSO-GA	1.90	16.30	3.90	5.90	0.28	0.19	0.24	0.33	0.24

malicious script code embedded in the web page will be executed. In this paper, XSS payload data set is obtained from Zhou and Wang,[41] which contains 30 features and 151,658 records, including 16,151 malicious samples (XSS payload) and 135,507 normal URLs. Moreover, we add another node representing the label of XSS attack corresponding to X31 in the learnt network. The features of XSS attack are listed in Table 12.

In this experiment, we use our proposed algorithm to learn the structure of BN from the XSS payload data set; then, we obtain the parameters in the learnt network by parameter learning; finally, we use the inference algorithm to predict labels of the test data set. In the experiment, we use our proposed method to learn the structure of XSS attack by randomly sampling 30,000 records as training data and 5000 records as test data in XSS payload data set. And parameter learning and inference algorithm are realized with functions contained in the bnlearn package in R.[#] Experimental results show that the accuracy of our proposed method can achieve 99.48% in XSS attack detection, which proves that the learnt network can effectively detect XSS attack compared to the accuracy shown in Zhou and Wang.[41] The structure of the learnt BN is shown in Figure 5.

As shown in Figure 5, the learnt DAG contains 19 nodes in the graph and X31 corresponds to the label of XSS attack. The other 12 nodes are not contained in the graph because they are

TABLE 7 The experimental results of four BN learning algorithms on SACHS network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	2.20	32.70	7.30	12.80	0.18	0.18	0.15	0.23	0.15
	NOTEARS	2.10	26.60	13.10	13.20	0.14	0.33	0.14	0.14	0.14
	GA	0.10	32.20	8.50	14.20	0.01	0.21	0.01	0.01	0.01
	PSO-GA	0.10	32.90	7.80	14.20	0.01	0.19	0.01	0.01	0.01
500	BRKGA	2.40	32.50	7.70	12.40	0.19	0.19	0.16	0.24	0.16
	NOTEARS	2.20	26.30	13.70	12.80	0.14	0.34	0.15	0.14	0.15
	GA	0.20	31.30	9.60	13.90	0.02	0.23	0.01	0.02	0.01
	PSO-GA	0.50	33.40	7.00	14.10	0.05	0.17	0.03	0.07	0.03
1000	BRKGA	2.60	32.10	7.60	12.70	0.20	0.19	0.17	0.25	0.17
	NOTEARS	2.10	26.70	13.50	12.70	0.14	0.34	0.14	0.13	0.14
	GA	0.40	31.60	8.60	14.40	0.03	0.21	0.03	0.04	0.03
	PSO-GA	0.20	33.00	7.90	13.90	0.02	0.19	0.01	0.02	0.01
1500	BRKGA	2.20	33.10	7.90	11.80	0.18	0.19	0.16	0.22	0.16
	NOTEARS	2.30	25.90	14.00	12.80	0.15	0.35	0.15	0.14	0.15
	GA	0.40	31.50	9.10	14.00	0.03	0.22	0.03	0.04	0.03
	PSO-GA	0.10	32.40	7.90	14.60	0.01	0.20	0.01	0.01	0.01
2000	BRKGA	2.90	33.00	6.50	12.60	0.23	0.16	0.19	0.31	0.19
	NOTEARS	1.60	26.20	13.90	13.30	0.11	0.35	0.11	0.10	0.11
	GA	0.20	31.30	8.80	14.70	0.02	0.22	0.01	0.02	0.01
	PSO-GA	0.60	33.20	7.60	13.60	0.05	0.19	0.04	0.07	0.04

TABLE 8 The experimental results of four BN learning algorithms on CHILD network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	2.30	156.20	10.80	20.70	0.13	0.06	0.10	0.18	0.10
	NOTEARS	2.90	150.20	17.30	19.60	0.14	0.10	0.13	0.14	0.13
	GA	1.50	150.00	16.10	22.40	0.07	0.10	0.06	0.09	0.06
	PSO-GA	0.60	148.90	17.00	23.50	0.03	0.10	0.02	0.03	0.02
500	BRKGA	3.10	156.50	9.80	20.60	0.17	0.06	0.13	0.24	0.13
	NOTEARS	2.70	150.40	18.00	18.90	0.13	0.11	0.13	0.13	0.13
	GA	2.20	147.30	18.80	21.70	0.10	0.11	0.09	0.10	0.09
	PSO-GA	1.10	151.50	14.00	23.40	0.06	0.08	0.04	0.07	0.04
1000	BRKGA	3.10	156.60	9.70	20.60	0.17	0.06	0.13	0.24	0.13
	NOTEARS	1.80	148.70	19.90	19.60	0.08	0.12	0.08	0.08	0.08

(Continues)

TABLE 8 (Continued)

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
	GA	1.00	149.30	16.90	22.80	0.05	0.10	0.04	0.06	0.04
	PSO-GA	2.00	152.30	13.80	21.90	0.10	0.08	0.08	0.13	0.08
1500	BRKGA	3.20	157.10	10.00	19.70	0.18	0.06	0.14	0.24	0.14
	NOTEARS	2.20	149.30	19.00	19.50	0.10	0.11	0.10	0.10	0.10
	GA	1.90	150.00	16.10	22.00	0.09	0.10	0.08	0.11	0.08
	PSO-GA	1.10	151.40	14.40	23.10	0.06	0.09	0.05	0.07	0.05
2000	BRKGA	2.90	157.30	9.70	20.10	0.16	0.06	0.13	0.23	0.13
	NOTEARS	1.90	147.00	22.20	18.90	0.08	0.13	0.09	0.08	0.09
	GA	1.00	148.80	17.50	22.70	0.05	0.11	0.04	0.05	0.04
	PSO-GA	1.20	151.10	14.70	23.00	0.06	0.09	0.05	0.08	0.05

TABLE 9 The experimental results of four BN learning algorithms on INSURANCE network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	6.30	281.40	22.80	40.50	0.17	0.07	0.13	0.22	0.13
	NOTEARS	6.70	268.50	36.80	39.00	0.15	0.12	0.15	0.15	0.15
	GA	1.70	274.90	25.00	49.40	0.04	0.08	0.03	0.06	0.03
	PSO-GA	2.30	281.30	19.00	48.40	0.06	0.06	0.05	0.11	0.05
500	BRKGA	6.80	281.50	22.00	40.70	0.18	0.07	0.14	0.24	0.14
	NOTEARS	5.20	265.20	42.10	38.50	0.11	0.14	0.12	0.11	0.12
	GA	2.20	278.60	21.70	48.50	0.06	0.07	0.04	0.09	0.04
	PSO-GA	2.60	283.20	17.30	47.90	0.07	0.06	0.05	0.13	0.05
1000	BRKGA	6.10	280.90	22.80	41.20	0.16	0.08	0.13	0.21	0.13
	NOTEARS	7.30	265.80	39.60	38.30	0.16	0.13	0.16	0.16	0.16
	GA	2.90	277.50	22.50	48.10	0.08	0.08	0.06	0.11	0.06
	PSO-GA	2.10	281.40	18.80	48.70	0.06	0.06	0.04	0.10	0.04
1500	BRKGA	6.00	280.30	23.60	41.10	0.16	0.08	0.13	0.20	0.13
	NOTEARS	5.70	265.40	41.20	38.70	0.12	0.13	0.13	0.12	0.13
	GA	2.70	280.10	20.40	47.80	0.07	0.07	0.05	0.12	0.05
	PSO-GA	3.40	283.70	16.30	47.60	0.10	0.05	0.07	0.17	0.07
2000	BRKGA	5.90	281.60	22.40	41.10	0.16	0.07	0.13	0.21	0.13
	NOTEARS	6.00	263.00	43.20	38.80	0.13	0.14	0.13	0.12	0.13
	GA	2.90	277.70	22.50	47.90	0.08	0.07	0.06	0.11	0.06
	PSO-GA	2.10	282.20	18.00	48.70	0.06	0.06	0.04	0.10	0.04

TABLE 10 The experimental results of four BN learning algorithms on ALARM network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	4.00	600.60	22.10	39.30	0.12	0.04	0.09	0.15	0.09
	NOTEARS	2.30	584.70	38.60	40.40	0.06	0.06	0.05	0.06	0.05
	GA	0.90	588.00	32.60	44.50	0.02	0.05	0.02	0.03	0.02
	PSO-GA	1.10	595.20	25.50	44.20	0.03	0.04	0.02	0.04	0.02
500	BRKGA	3.10	599.60	23.40	39.90	0.09	0.04	0.07	0.12	0.07
	NOTEARS	1.40	581.20	41.40	42.00	0.03	0.07	0.03	0.03	0.03
	GA	1.40	589.20	32.30	43.10	0.04	0.05	0.03	0.04	0.03
	PSO-GA	0.90	594.50	26.50	44.10	0.02	0.04	0.02	0.03	0.02
1000	BRKGA	3.50	600.30	22.40	39.80	0.10	0.04	0.08	0.14	0.08
	NOTEARS	1.00	580.50	42.50	42.00	0.02	0.07	0.02	0.02	0.02
	GA	1.50	589.70	31.70	43.10	0.04	0.05	0.03	0.05	0.03
	PSO-GA	1.50	592.20	28.10	44.20	0.04	0.05	0.03	0.05	0.03
1500	BRKGA	3.50	600.80	22.70	39.00	0.10	0.04	0.08	0.13	0.08
	NOTEARS	1.30	579.70	42.50	42.50	0.03	0.07	0.03	0.03	0.03
	GA	1.20	589.40	31.60	43.80	0.03	0.05	0.03	0.04	0.03
	PSO-GA	1.00	594.00	26.50	44.50	0.03	0.04	0.02	0.04	0.02
2000	BRKGA	3.10	600.60	23.10	39.20	0.09	0.04	0.07	0.12	0.07
	NOTEARS	1.20	581.30	41.90	41.60	0.03	0.07	0.03	0.03	0.03
	GA	1.60	589.70	31.20	43.50	0.04	0.05	0.04	0.05	0.04
	PSO-GA	1.90	593.00	27.60	43.50	0.05	0.04	0.04	0.06	0.04

isolated nodes that their relationships are not reflected in the final learnt structure. According to Zhou and Wang,[41] there are two important nodes—X1, X2 also shown in our learnt network, which proves that our method can obtain a reasonable result of XSS attack detection. X1 represents the length of URL length, which is a very important feature because the length of malicious script is usually longer than the normal. Figure 5 also shows that node 2 is directly related to the final label, X31. In practice, the character "alert" is frequently used in real XSS tests,[41] which can also verify that the learnt network is effective in XSS attack detection.

5 | CONCLUSION

BN structure learning is a NP-hard problem and difficult to obtain the global exact solution. In this perspective, evolutionary algorithms provide an effective way to obtain a reliable solution and is easy to implement. However, the accuracy of evolutionary algorithms is an important factor affecting their wide applications. In this paper, we propose to use NOTEARS generating initial solutions and use BRKGA searching for more possible solutions.

TABLE 11 The experimental results of four BN learning algorithms on HAILFINDER network

Samples	Algorithm	TP	TN	FP	FN	F1	FPR	TPR	Precision	Recall
100	BRKGA	3.80	1444.80	33.70	57.70	0.08	0.02	0.06	0.10	0.06
	NOTEARS	3.40	1443.10	39.80	53.70	0.07	0.03	0.06	0.08	0.06
	GA	2.00	1424.90	49.10	64.00	0.03	0.03	0.03	0.04	0.03
	PSO-GA	1.20	1426.40	47.60	64.80	0.02	0.03	0.02	0.02	0.02
500	BRKGA	3.20	1437.30	40.20	59.30	0.06	0.03	0.05	0.07	0.05
	NOTEARS	2.80	1441.70	41.80	53.70	0.06	0.03	0.05	0.06	0.05
	GA	1.80	1424.20	49.80	64.20	0.03	0.03	0.03	0.03	0.03
	PSO-GA	1.80	1425.20	48.80	64.20	0.03	0.03	0.03	0.04	0.03
1000	BRKGA	3.10	1440.60	37.90	58.40	0.06	0.03	0.05	0.08	0.05
	NOTEARS	4.10	1441.30	40.90	53.70	0.08	0.03	0.07	0.09	0.07
	GA	2.70	1425.80	48.20	63.30	0.05	0.03	0.04	0.05	0.04
	PSO-GA	1.40	1425.80	48.20	64.60	0.02	0.03	0.02	0.03	0.02
1500	BRKGA	2.20	1439.10	39.90	58.80	0.04	0.03	0.04	0.05	0.04
	NOTEARS	2.50	1442.60	41.10	53.80	0.05	0.03	0.04	0.06	0.04
	GA	2.00	1423.70	50.30	64.00	0.03	0.03	0.03	0.04	0.03
	PSO-GA	2.10	1426.40	47.60	63.90	0.04	0.03	0.03	0.04	0.03
2000	BRKGA	5.00	1439.10	36.80	59.10	0.09	0.02	0.08	0.12	0.08
	NOTEARS	4.90	1428.60	54.30	52.20	0.08	0.04	0.09	0.08	0.09
	GA	2.70	1423.40	50.60	63.30	0.05	0.03	0.04	0.05	0.04
	PSO-GA	1.20	1427.20	46.80	64.80	0.02	0.03	0.02	0.03	0.02

TABLE 12 The features of XSS attack

Index	Feature	Index	Feature	Index	Feature
1	URL length	11	href	21	iframe
2	alert	12	javascript	22	onclick
3	script	13	window	23	the number of single quotes
4	onerror	14	fromcharcode	24	the number of double quotes
5	confirm	15	document	25	the number of left angle brackets
6	img	16	onmouseover	26	the number of right angle brackets
7	onload	17	cookie	27	the number of backslashes
8	eval	18	domain	28	the number of commas
9	prompt	19	onfocus	29	the number of pluses
10	src	20	expression	30	http://, https://, file://

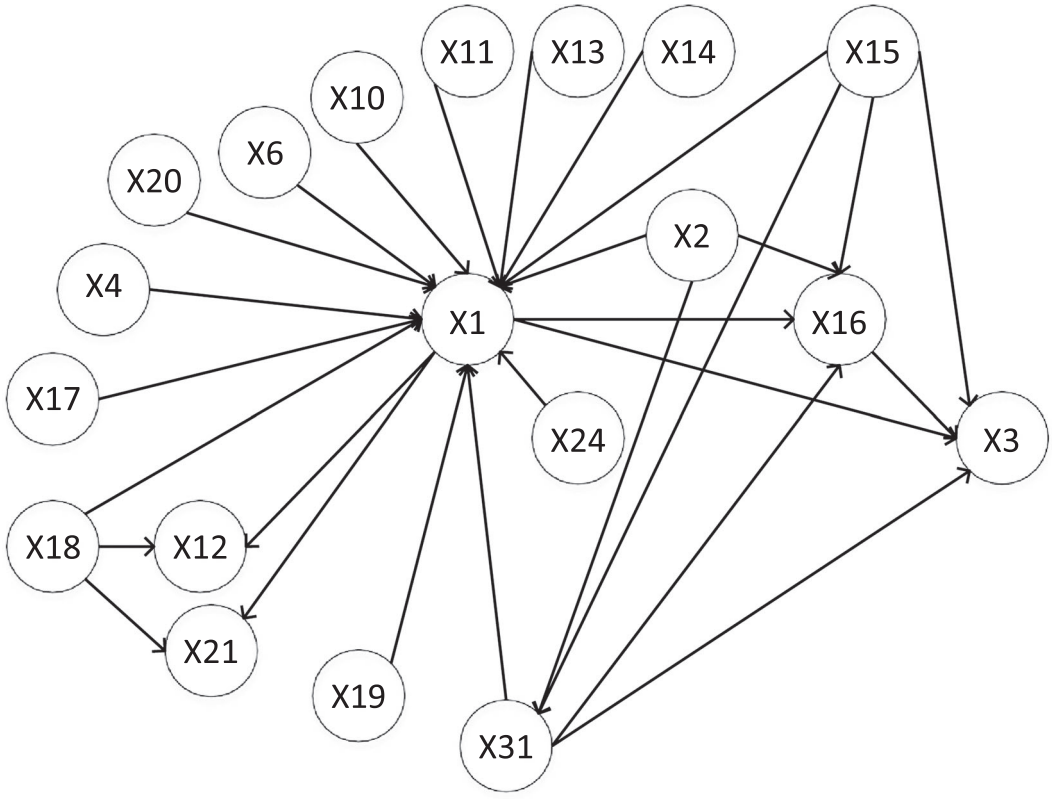


FIGURE 5 The structure of the learnt XSS attack network

To elaborate the improvements of our proposed algorithm, we firstly introduce the original BRKGA and its improvements based on RKGA. And we introduce the decoder used in this paper, which is the most important block of the algorithm. In this paper, we use NOTEARS algorithm as the decoder because this algorithm can interpret random keys into initial solutions of BNSL problem and NOTEARS algorithm can obtain a local optimal solution to improve the performance of the original BRKGA. Next, we list the crossover and mutation operators used in this paper and elaborate their principles. At the end of our proposed algorithm, we need postprocessing section to ensure the obtained solution discretized into 0-1 matrix. Finally, the acyclicity of the 0-1 matrix should be checked and if the 0-1 matrix contains cycles, we list how to deal with them. Experimental results show that our proposed algorithm can achieve a good performance on benchmark networks and the real data set.

In the future, we will continue to apply evolutionary algorithms to BN structure learning. At the same time, we will also improve the existing evolutionary algorithms based on the characteristics of this problem to obtain more accurate directed acyclic graphs.

ACKNOWLEDGMENTS

This study is supported by National Natural Science Foundation of China (No. 61703416), Huxiang Youth Talent Support Program (No. 2021RC3076) and Training Program for Excellent Young Innovators of Changsha (No. KQ2009009).

ORCID

Yun Zhou https://orcid.org/0000-0001-7328-0275

ENDNOTES

*https://www.bnlearn.com/bnrepository/.

†https://yzhou.github.io/code.

#https://www.bnlearn.com/.

REFERENCES

1. Grüttemeier N, Komusiewicz C. Learning Bayesian networks under sparsity constraints: a parameterized complexity analysis. In: *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence*, Vol. 7; 2020:4217-4223.
2. Gu J, Zhou Q. Learning big Gaussian Bayesian networks: partition, estimation and fusion. *J Mach Learn Res*. 2020;21.
3. de Campos CP, Ji Q. Properties of Bayesian dirichlet scores to learn Bayesian network structures. In: *Twenty-Fourth AAAI Conference on Artificial Intelligence 2010*. Atlanta, GA; 2010.
4. Bartlett M, Cussens J. Integer linear programming for the Bayesian network structure learning problem. *Artif Intell*. 2017;244:258-271.
5. Hongru L, Huiping G. A hybrid structure learning algorithm for Bayesian network using experts' knowledge. *Entropy*. 2018;20(8):620.
6. Oyen D, Lane T. Bayesian discovery of multiple Bayesian networks via transfer learning. In: *2013 IEEE 13th International Conference on Data Mining*; 2013:577-586.
7. Yu Y, Chen J, Gao T, Yu M. DAG-GNN: DAG structure learning with graph neural networks. In: *International Conference on Machine Learning (ICML)*; 2019.
8. Correia AHC, Cussens J, de Campos C. On pruning for score-based Bayesian network structure learning. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, Vol. 108. PMLR; Aug 26-28, 2020:2709-2718.
9. Chu VW, Wong RK, Chen F, Fong S, Hung PC. Self-regularized causal structure discovery for trajectory-based networks. *J Comput Syst Sci*. 2016;82(4):594-609.
10. Yu K, Wu X, Ding W, Wang H. Exploring causal relationships with streaming features*. *Comput J*. 2012; 55(9):1103-1117. doi:10.1093/comjnl/bxs032
11. Jin Z, Li J, Liu L, Le TD, Sun B, Wang R. Discovery of causal rules using partial association. In: *2012 IEEE 12th International Conference on Data Mining*; 2012:309-318. doi:10.1109/ICDM.2012.36
12. Hong Y, Liu Z, Mai G. An efficient algorithm for large-scale causal discovery. *Soft Comput*. 2017;21(24): 7381-7391.
13. Ma S, Li J, Liu L, Le TD. Mining combined causes in large data sets. *Knowl-Based Syst*. 2016;92:104-111.
14. Khanteymoori AR, Olyaee MH, Abbaszadeh O, Valian M. A novel method for Bayesian networks structure learning based on breeding swarm algorithm. *Soft Comput*. 22(9):3049-3060.
15. Junzhong J, Cuicui Y, Jiming L, Jinduo L, Baocai Y. A comparative study on swarm intelligence for structure learning of Bayesian networks. *Soft Comput*. 2016;21(22):1-26.
16. Ji J, Wei H, Liu C. An artificial bee colony algorithm for learning Bayesian networks. *Soft Comput*. 2013; 17(6):983-994.
17. Yang C, Ji J, Liu J, Liu J, Yin B. Structural learning of Bayesian networks by bacterial foraging optimization. *Int J Approx Reason*. 2016;69:147-167.
18. Campos L, Fernández-Luna J, Gámez J, Puerta JM. Ant colony optimization for learning Bayesian networks. *Int J Approx Reason*. 2002;31(3):291-311.
19. Zhang Q, Li Z, Zhou CJ, Wei XP. Bayesian network structure learning based on the chaotic particle swarm optimization algorithm. *Genetics Mol Res*. 2013;12(4):4468-4479.
20. Wang J, Liu S. A novel discrete particle swarm optimization algorithm for solving Bayesian network structures learning problem. *Int J Comput Math*. 2019;96:1-23.

21. Gheisari S, Meybodi MR. BNC-PSO: structure learning of Bayesian networks by Particle Swarm Optimization. *Inform Sci*. 2016;348:272-289.
22. Fukuda S, Yoshihiro T. Learning Bayesian networks using probability vectors. *Adv Intell Syst Comput*. 2014; 290:503-510.
23. Lee J, Chung W, Kim E, Kim S. A new genetic approach for structure learning of Bayesian networks: matrix genetic algorithm. *Int J Control Autom Syst*. 2010;8(2):398-407.
24. Vafaee F. Learning the structure of large-scale Bayesian networks using genetic algorithm. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO'14, Association for Computing Machinery, New York, NY; 2014:855-862.
25. Khanteymoori AR, Menhaj MB, Homayounpour MM. Structure learning in Bayesian networks using asexual reproduction optimization. *Etri J*. 2011;33(1):39-49.
26. Carvalho A. A cooperative coevolutionary genetic algorithm for learning Bayesian network structures. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. GECCO'11, Association for Computing Machinery, New York, NY; 2011:1131-1138.
27. Shetty S, Song M. Structure learning of Bayesian networks using a semantic genetic algorithm-based approach. In: *ITRE 2005. 3rd International Conference on Information Technology: Research and Education*; 2005:454-458.
28. Baioletti M, Milani A, Santucci V. Learning Bayesian networks with algebraic differential evolution. In: *Parallel Problem Solving from Nature*. Springer International Publishing; 2018:436-448.
29. Constantinou AC. Learning Bayesian networks with the Saiyan algorithm. *ACM Trans Knowl Discov Data*. 2020;14(4):44.
30. Gonçalves J, Almeida J. A hybrid genetic algorithm for assembly line balancing. *J Heuristics*. 2002;8: 629-642.
31. Sun B, Zhou Y. Biased random-key genetic algorithm for structure learning. In: *12th International Conference on Advances in Swarm Intelligence*, ICSI 2021. Vol. 12689 LNCS; 2021:399-411.
32. Goldberg DE. Genetic algorithms in search, optimization and machine learning. 1st ed. Addison-Wesley Longman Publishing Co., Inc.; 1989.
33. Kennedy J, Eberhart R. Particle swarm optimization. In: *Proceedings of ICNN'95—International Conference on Neural Networks*. Vol. 4; 1995:1942-1948.
34. David MC, Christopher M, David H. Large-sample learning of bayesian networks is NP-hard. *J Mach Learn Res*. 2004;5(1999):1287-1330.
35. Bean JC. Genetic algorithms and random keys for sequencing and optimization. *Orsa J Comput*. 1994;6: 154-160.
36. Fernando J, Mauricio G. Biased random-key genetic algorithms for combinatorial optimization. *J Heuristics*. 2011;17(5):487-525.
37. Zheng X, Aragam B, Ravikumar P, Xing EP. DAGs with NO TEARS: Continuous optimization for structure learning. In: *Advances in Neural Information Processing Systems*; 2018:9472-9483.
38. Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. *Siam J Scientific Comput*. 1995;16(5):1190-1208.
39. Wei D, Gao T, Yu Y. DAGs with no fears: a closer look at continuous optimizationfor learning Bayesian networks. In: *Advances in Neural Information Processing Systems*; 2020:3895-3906.
40. Constantinou AC. Evaluating structure learning algorithms with a balanced scoring function. *CoRR*. 2019; abs/1905.12666.
41. Zhou Y, Wang P. An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence. *Comput Security*. 2019;82(May):261-269.

How to cite this article: Sun B, Zhou Y. Bayesian network structure learning with improved genetic algorithm. *Int J Intell Syst*. 2022;37:6023-6047. doi:10.1002/int.22833