scientific reports

OPEN

Bayesian network structure learning by opposition-based learning

Baodan Sun[1,2,3], Xinyi Zhang[1,3], Junhui Jiang[1,2,3]✉, Jianguang Gong[1,2,3] & Dan Lin[1,2,3]

As a classical basic model for causal inference, Bayesian networks are of vital importance both in artificial intelligence with uncertainty and interpretability. The significant status of Bayesian networks in these research orientations depends on its topological structure, namely directed acyclic graphs. Bayesian network structure learning is a well-known NP-hard problem, and its computation accuracy is still worth being further studied. In this paper, we propose a new Bayesian network structure learning algorithm, OP-PSO-DE, which combines Particle Swarm Optimization(PSO) and Differential Evolution to search for the optimal structure. Since the computation complexity of BN structure learning increases exponentially with the number of nodes, the proposed algorithm incorporates opposition-based learning to narrow the search space of heuristic algorithms, which can effectively accelerate the searching process. Experimental results show that the proposed algorithm achieves better performances than other state-of-the-art structure learning algorithms when the sample size is 500. The source code of the paper can be found at this link: https://github.com/sunbaodan-hrbeu/paper_code.

Keywords Opposition-based learning, Differential evolution, Particle swarm optimization, Structure learning, Bayesian networks

Bayesian networks are widely used in application scenarios of artificial intelligence, and also the foundation of causal inference. A whole Bayesian network can be divided into two components: topological structure and parameters. The topological structure is represented by directed acyclic graphs(DAGs) to show the dependent relations between each two random variables directly and the parameters precisely regardless time complexity. Usually, there are three strategies used in structure learning algorithms: constraint-based approach, score-based approach and hybrid approach. Constraint-based approach employs conditional independence tests to ensure the dependencies and independences between each two random variables(represented by nodes in the graph). Score-based approach uses heuristic algorithms searching for the best structure in the solution space according to scoring functions, which is broadly utilized in structure learning. Hybrid approach combines both of them, which uses constraint-based approach to obtain the skeleton and uses score-based approach to search for the best structure. However, since Bayesian network structure learning is a non-convex combinatorial optimization problem, common numerical optimization methods are ineffective in solving it. Although branch-and-bound algorithms have been proposed for Bayesian network structure learning, their performance diminishes when handling large-scale networks with more than 100 nodes. And when we use Bayesian networks to deal with classification tasks, it is not necessary to ensure that the learnt networks are exactly the same as the original ones since classification models should have certain generalization ability. In this context, Bayesian network structure learning based on heuristic algorithms was proposed and developed in recent years.

In previous works, Villa-Blanco et al.[1] proposed a hybrid structure learning algorithm, which used PC algorithm to reconstruct the skeleton of the class subgraph and hill climbing was used to search for the directed edge. Jose et al.[2] proposed to use CI tests to construct an undirected graph, and CIGAR-based search method was used for evolving a high-quality network. WANG et al.[3] proposed ESLH algorithm, which used dynamic threshold and skeleton learning method based on triangle breaking combining with hill climbing to obtain BN structures. In these works, the authors compare the proposed hybrid BN structure learning algorithm with classical constraint-based algorithm, such as PC, and score-based algorithms, such as Tabu search, hill climbing and so on. The experiments in these papers illustrate hybrid approach for BN structure learning achieve better

[1]Harbin Engineering University, College of Computer Science and Technology, Harbin 150001, China. [2]National Key Laboratory of Smart Farm Technologies and Systems, Harbin 150001, China. [3]Baodan Sun, Xinyi Zhang, Junhui Jiang, Jianguang Gong and Dan Lin contributed equally to this work. ✉email: jhjiang@hrbeu.edu.cn

general performances, since the learnt skeleton or undirected graph can constrain the search space, and score-based can search for a relatively accurate directed acyclic graph in a restricted search space.

Also, there were many researchers applying various heuristic algorithms into Bayesian network structure learning, where traditional heuristic algorithms include Hill Climbing[4], Simulated Annealing[56], Ant Colony Optimization[7], Particle Swarm Optimization[8] and so on. Also, some relatively new proposed Bayesian network structure learning algorithms based on heuristic algorithms are worth mentioning because of their innovations and original solving strategies: Wang et al.[9] proposed a novel heuristic function and used A* searching algorithm to find the best structure of Bayesian networks. Yang et al.[10] proposed to use a well-known metaheuristic method called scatter search to solve BN structure learning problem. He et al.[11] proposed the neighboring complete node ordering search algorithm to find the node ordering of Bayesian networks and used hill climbing to find the best network structure. Haoran Liu et al.[12] proposed to use an improved Harris Hawks optimization(HHO) for Bayesian network structure learning. Kareem et al.[13] proposed to utilize Elephant Swarm Water Search Algorithm(ESWSA) for Bayesian network structure learning. Awla et al.[14] used reversing, moving, and deleting to create the Falcon Optimization Algorithm(FOA) to find the best structure of DAGs. Soloviev et al.[15] proposed to use quantum approximate optimization algorithm(QAOA) to solve Bayesian network structure learning problem by employing $3n(n-1)/2$ qubits, where n is the number of nodes of the learnt Bayesian network. Wang et al.[16] proposed a novel discrete firefly algorithm to learn Bayesian networks. These research articles prove that using heuristic algorithms to search for the best topological structure is an effective method to solve structure learning problems in finite time, but they are also suffering from totally random searching and huge solution space. Therefore, it is necessary to conduct intensive studies on these heuristic algorithms to improve their performances.

In our previous work, we proposed PC-PSO algorithm[8] for Bayesian network structure learning, which combines the well-known constraint-based approach, PC algorithm, to obtain the initial solutions and BNC-PSO[17] to search for the best network structure in the solution space. However, the convergence rate and the accuracy of PC-PSO are still worth further disscussed. To be specific, PC-PSO needs 78.3 iteration times out of 10 experiments to achieve convergence and the corresponding BIC score is -9014.95(the benchmark is -9468.28) when the sample size is 500 on INSURANCE network. So, in this paper, we propose to improve the existing PC-PSO algorithm with the opposition-based learning approach to narrow the search space of heuristic algorithms, which can effectively accelerate the searching process. Unlike recent neural-based continuous optimization methods (e.g., NOTEARS[18], DAG-GNN[19] or GraN-DAG[20]), our approach is situated in the heuristic score-based family, and is particularly suitable for discrete-variable domains and black-box scoring functions. To increase the diversity of the population in the searching process to find more feasible solutions, we employ DE algorithm instead of GA algorithm in PC-PSO. DE can make full use of individuals in the population to execute mutation and crossover operations, while GA only changes certain elements of the individuals. Meanwhile, the proposed algorithm combines Particle Swarm Optimization(PSO) with Differential Evolution(DE) to search for more accurate network structures in the solution space.

To the best of our knowledge, the proposed method firstly applies opposition-based learning into Bayesian network structure learning. Specifically, in the whole searching process, we generate regular solutions and their opposite solutions at the same time, which can obtain solutions closer to the global optimum. Also, in this paper, we utilize a new mutation operator of DE, the binary mutation with probability difference, to deal with the discrete Bayesian network structure learning algorithm, compared to the linear mutation operator used in PC-PSO.

The remainder of this paper is organized as follows. Section 2 is the preliminaries of this paper, which introduces the basic knowledge of Bayesian networks. Section 3 introduces the basic idea of Particle Swarm Optimization and its improved algorithm, BNC-PSO. Section 4 shows the details of our proposed OP-PSO-DE algorithm. Section 5 shows experiment settings and all the experiment results of the proposed algorithm. Section 6 is the conclusion of this paper.

Preliminary

Bayesian networks

As an important probabilistic graphical model, Bayesian networks originally derived from the Bayes theorem and are widely utilized because of its interpretability. Bayesian networks use directed acyclic graphs to represent the dependent relations between two nodes and use the conditional probability table to represent conditional probabilities of each two nodes. According to the implicit independence assumption in the structure, given its parent node, X_i is conditionally independent with its non-child nodes, so the joint probability distribution can be decomposed into the product of multiple conditional probability distributions:

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

where $P(X_i|Pa(X_i))$ represents the conditional probability of X_i given its parent node $Pa(X_i)$; there is a directed edge pointing from each node in $Pa(X_i)$ to X_i.

We can conclude two steps taken to represent the dependent relations and independent relations of these nodes in the above equation with directed graphs: firstly, each random variable in the equation is represented as a node in the directed graph; secondly, for each node X_i, a directed edge is drawn starting from each node in the parent node set $Pa(X_i)$ pointing to X_i.

In most application scenarios, we can only obtain the original dataset and want to obtain the directed acyclic graph according to it, which is the structure learning problem. As we mentioned above, there are three strategies can be adopted: constraint-based approach, score-based approach and hybrid approach. Constraint-based approach was firstly employed in structure learning, but it can only obtain the completed partially acyclic graph(CPDAG), which refers to the Markov equivalence class of the directed acyclic graph. Therefore, the score-based approach was widely utilized, which contains two parts: scoring functions and searching algorithms. Searching algorithms are used to search for the best network structure that maximizes scoring functions in the feasible domain. Usually, heuristic algorithms are used to search for the best solutions and different scoring functions are introduced as below, which are also used in our proposed algorithm.

Scoring function

There are three classical scoring functions widely used in many research articles: BDe[21], BIC[22] and MDL[23]. The derivation process of BDe considers prior assumptions on parameters, while no such assumptions are made in BIC and MDL. Compared to BDe, BIC and MDL are intuitive and not easy to be impacted by the errors of the dataset. To be specific, BDe score assumes the parameter distribution, and then calculates the fit degree of current network structure and data to find out the network structure that maximizes the posterior distribution. If we assume parameters are subject to Dirichlet distribution, it can be written as:

$$Dir(\theta_{ij}|\alpha_1,\alpha_2,\ldots,\alpha_r)=\frac{\Gamma\left(\sum\limits_{k=1}^{r}\alpha_k\right)}{\prod\limits_{k=1}^{r}\Gamma(\alpha_k)}\prod_{k=1}^{r}\theta_{ijk}^{\alpha_k-1}\qquad \alpha_k>0 \tag{2}$$

We can calculate BDe score according to the equation as below:

$$P(D|G)=\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha'_{ij})}{\Gamma(\alpha'_{ij}+\alpha_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha'_{ijk}+\alpha_{ijk})}{\Gamma(\alpha'_{ijk})} \tag{3}$$

where α'_{ijk} is the hyperparameter ,$\alpha'_{ij}=\sum\limits_{k=1}^{r_i}\alpha'_{ijk}$ and $\Gamma(\cdot)$ is the gamma function.

BIC score is the logarithm of BDe, and it evaluates the likelihood function of current network structure and observation data. Thus, we can write BIC score function as:

$$\log P(G|D)=\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}m_{ijk}\log\frac{m_{ijk}}{m_{ij}}-\frac{1}{2}\sum_{i=1}^{n}q_i(r_i-1)\log m \tag{4}$$

where *m* is data size. m_{ijk} is the probability that X_i takes the *k*-th value and its parent node takes the *j*-th value. $\sum\limits_{i=1}^{n}q_i(r_i-1)$ is the amount of parameters contained in the network.

Compared to BIC, MDL score adds an additional penalty term to the fitness degree of current network and observation data, which calculates the sum of the description length of network structure and sample data. The calculation equation can be written as:

$$-DL(G,D)\approx -m\sum_{i=1}^{n}H(X_i|Pa_i)-\frac{1}{2}\sum_{i=1}^{n}q_i(r_i-1)\log m \tag{5}$$

where $H(X_i|Pa_i)$ the conditional entropy of X_i relative to Pa_i.

Particle swarm optimization

The basic principle

Inspired by the regularity of bird flock foraging behavior, James Kennedy and Russell Eberhart proposed Particle Swarm Optimization[24] in 1995, which searches for the optimal solution through collaboration and sharing information among individuals in the population. PSO firstly initializes the population of particles, which has two attributes: velocity and position. To be specific, the velocity of the *i*-th particle in *d*-dimensional searching space, $V_i=(V_{i1},V_{i2},\ldots,V_{id})$, represents their searching speed and the position of the *i*-th particle, $X_i=(X_{i1},X_{i2},\ldots,X_{id})$, represents the candidate solution. Each particle searches for the optimal solution in the search space individually and represents it as the current individual extreme value, P_{best}. Meanwhile, each particle shares their individual best solution with other particles in the entire population, and finds out the best individual extreme value, G_{best}, as the current global optimal solution of the population. All the particles in the population adjust their speed and position according to the current individual best solution and the global optimal solution. The velocity and position of particles can be updated according to the following equations:

$$v_i(t+1)=v_i(t)+c_1r_1(P_{best}(t)-x_i(t))+c_2r_2(G_{best}-x_i(t)) \tag{6}$$

$$x_i(t+1)=x_i(t)+v_i(t+1) \tag{7}$$

where *t* is the number of iterations. *w* is the inertia weight. c_1, c_2 are acceleration constants. r_1, r_2 are random numbers, usually $r_1, r_2 \in [0, 1]$. The original PSO algorithm was proposed to solve continuous optimization problems, while BN structure learning is a discrete optimization problem in general. Therefore, the modified PSO algorithms used to solve discrete BN structure learning problem were proposed and BNC-PSO is one of the state-of-the-art.

BNC-PSO

BNC-PSO[17] was proposed by Gheisari and Meybodi in 2016 and achieves good performances while solving BN structure learning problems. BNC-PSO combines the original PSO algorithm with Genetic Algorithm(GA), which utilizes PSO algorithm to search in the candidate solution space and utilizes GA to generate more discrete solutions. To improve the accuracy, BNC-PSO executes two crossover operations compared to the original GA: the first crossover operation is executed with individual best solutions; the second crossover operation is executed with the global optimal solution. According to BNC-PSO, the updated equations of particles can be written as:

$$
\begin{aligned}
X_i(t) &= N_3(N_2(N_1(X_i(t-1), w), c_1), c_2) \\
V_i(t) = W_i(t) &= N_1(X_i(t-1), w) = \begin{cases} M(X_i(t-1)), & r_1 < w \\ X_i(t-1), & others \end{cases} \\
S_i(t) &= N_2(W_i(t), c_1) = \begin{cases} C_p(W_i(t)), & r_2 < c_1 \\ W_i(t), & others \end{cases} \\
X_i(t) &= N_3(S_i(t), c_2) = \begin{cases} C_g(S_i(t)), & r_3 < c_2 \\ S_i(t), & others \end{cases}
\end{aligned}
\tag{8}
$$

where *M* denotes the mutation operation. C_p denotes the crossover of each individual and its personal best solution. C_g denotes the crossover of each individual and global best solution. N_1, N_2 and N_3 represent the results of these three operators. *w* is the mutation probability and c_1, c_2 are crossover probabilities. $r_1, r_2, r_3 \in [0, 1]$ are random numbers.

OP-PSO-DE: BN structure learning based on opposition-based learning

Opposition-based learning

Opposition-based learning was proposed by Hamid R. Tizhoosh[25], which is an effective scheme that can improve the convergence speedup of machine intelligence algorithms. The basic principle of opposition-based learning is to generate opposite solutions of feasible solutions and evaluate both of them to choose better solutions as the next generation. To explain the theoretical reasoning behind why opposition-based learning enhances the convergence of Bayesian network (BN) structure learning, we must first recognize that BN structure learning is a combinatorial optimization problem, which can be addressed through heuristic algorithms, as demonstrated in previous researches. BN structure learning based on heuristic algorithms often begin with a random initial guess, which is typically far from the optimal solution. In the worst case, the initial guess may be at the opposite end of the solution space, causing the search algorithm to spend significantly more time finding the correct answer. Without sufficient prior knowledge, it is difficult to make a perfect initial solution. Logically, one should explore in all directions at once, or more specifically, in the opposite direction. By comparing the estimate and its counter-estimate, the search space can be progressively halved, and the algorithm will continue to narrow the search until one of these estimates is sufficiently close to the optimal solution. In this way, the searching efficiency and global optimization capability of searching algorithms can be improved.

Let $X = (x_1, x_2, \ldots, x_d)$ be a feasible solution in *d*-dimensional space, where $x_i \in [a_i, b_i]$ are real numbers. Then, its corresponding opposite solution can be written as:

$$
\begin{aligned}
\tilde{X} &= (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_d) \\
\tilde{x}_i &= a_i + b_i - x_i
\end{aligned}
\tag{9}
$$

where $i = 1, 2, \ldots, d$.

The original opposition-based learning was used to improve the searching process of continuous optimization problem, but it can also be applied into solving binary domain. In this paper, we focus on the discrete Bayesian network structure learning problem, where its solution space is binary. Thus, we apply the binary opposition-based learning into the searching space of our proposed algorithm. For the opposition solutions in binary domain, (9) can be written as:

$$\tilde{x}_i = 1 - x_i \tag{10}$$

The binary opposite-learning[26] was proven mathematically, where also proved that opposite solutions can find candidate solutions more closer to the global optimum. In this paper, we choose certain particles according to their fitness values as elite solutions to execute opposition-based learning, which can increase the diversity of the population. Also, selecting elite solutions from the current candidate solutions and the opposite solutions as offspring individuals can improve the convergence speedup of the algorithm. The executing process of opposition-based learning is shown in Fig.1.

Differential evolution

As we mentioned above, PSO is designed to solve continuous optimization problem and we focus on discrete BN structure learning problem in this paper because discrete BNs are more common in practical applications. To discretize the original PSO algorithm, we vary candidate solutions with the mutation operator and crossover operator of Differential Evolution(DE) algorithm[27].

DE algorithm is an evolutionary algorithm that simulates the process of cooperation and competition among individuals in the population. In DE algorithm, each candidate solution is represented by each individual in the population. In the initialization phase, the initial population consisting of *N* individuals is generated and each individual, $X_i(t)$, is a *d*-dimensional vector, where $i = 1, 2, \ldots, N$ and *t* is the number of iterations. $X_i(t)$ is randomly initialized within the searching space defined by the optimization problem. Then, new individuals are obtained by mutation operator and crossover operator, and individuals with higher fitness values are selected to generate the next population in this paper. The details of these operations are introduced below.

The binary mutation with probability difference

In the mutation operation, the standard mutation operation first randomly selects three different individuals in the population, then multiplies the difference of these three individuals by a factor (difference weight) and adds it to another target individual to generate a new individual. Mutation operator used in DE algorithm is to randomly select three individuals, $X_{r1},\ X_{r2},\ X_{r3}$ in the population and the new offspring can be obtained:

$$V_i(t) = X_{r1}(t) + F \cdot (X_{r2}(t) - X_{r3}(t)) \tag{11}$$

where $X_{r2}(t) - X_{r3}(t)$ is the differential variation.

To tackle discrete BN strucure learning problem, we adopt the binary mutation with probability difference[28] to improve the convergence of the algorithm and ensure its global searching capability. The mutation operator selects the best vector among three randomly chosen vectors as the base vector, and employs the difference between the remaining two vectors as a mutation probability to be used on the base vector to generate a mutation vector for the next crossover operator. For the *i*-th vector in the population $X_i(t)$, the offspring generated by the binary mutation with probability difference can be calculated as below[28]:

$$\begin{cases} V_i(t) = (v_{i,1}(t), v_{i,2}(t), \ldots, v_{i,d}(t)) \\ v_{i,j}(t) = \begin{cases} x_{best,j}(t), & c_{i,j} < rand \\ 1 - x_{best,j}(t), & otherwise \end{cases} \\ C_i = \begin{cases} \sigma, \quad if \quad X_{best}(t) \prec X_i(t) \\ min\{1,\ F \cdot (X_{r1}(t) \oplus X_{r2}(t)) + \sigma\}, \quad otherwise \end{cases} \end{cases} \tag{12}$$

where $V_i(t)$ denotes the target vector generated by the mutation operation, and $X_{best}(t)$ is the best vector among the three randomly chosen vectors and $X_{r1}(t),\ X_{r2}(t)$ are the remaining two vectors. If $X_{best}(t)$ is superior to $X_i(t)$(denoted by $X_{best}(t) \prec X_i(t)$ in the equation), the probability vector $C_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,d})$ equals to σ; otherwise, the probability difference $min\{1,\ F \cdot (X_{r1}(t) \oplus X_{r2}(t)) + \sigma\}$ is taken and $\oplus$ represents XOR operation. σ is a turbulence coefficient, and experiments in [28] show that its value should be taken in [0.001, 0.01]. $F \in [0, 1]$ is a scale parameter to control the learning rate. In this paper, we set *rand* in the equation equals to the mutation probability *w*.

Crossover operator

Crossover operator in DE algorithm is designed to be a discrete method to obtain new offspring with recombining the elements of vectors. To search for optimal solutions in discrete space, the current candidate solutions should vary in different iterations, thus the crossover operator is necessary in the algorithm. In crossover operation, the algorithm exchanges parts of the elements of the solution vectors with a certain probability to produce a new individual. It is a little bit different from crossover operator in Genetic Algorithms(GA) and the trial vectors $S_i(t)$ are obtained by recombining $V_i(t)$ with $X_i(t)$:

$$\begin{gathered} S_i(t) = (s_{i,1}(t), s_{i,2}(t), \ldots, s_{i,d}(t)) \\ s_{i,j}(t) = \begin{cases} v_{i,j}(t), & rand < CR \ or \ j = h \\ x_{i,j}(t), & otherwise \end{cases} \end{gathered} \tag{13}$$

where $rand \in [0, 1]$ is a random number and *CR* is the crossover probability. *h* is a random number in [1, *d*] to ensure that there is at least one element chosen from $V_i(t)$.

For our proposed algorithm, we combine PSO with DE and adopt the crossover strategy of BNC-PSO, which executes two crossover operations: the crossover with particle best solution and the crossover with global best solution. In this way, we reserve the effective elements in the vectors to speed up the convergence of our proposed algorithm. As we mentioned above, BN structure learning is a NP-hard combinatorial optimization problem, which means that the searching process is quite complex. So we adopt two times crossover operations to reserve effective information contained in these particles.

OP-PSO-DE

To illustrate our proposed algorithm, we list all the necessary details of critical process in this section. In this paper, we use the adjacent matrix of DAGs to represent the dependent and independent relations between nodes.

To be specific, if there is one directed edge from node X_1 pointing to node X_2, the corresponding element in the adjacent matrix is set to 1, namely $a_{X_1X_2} = 1$; if there are no edges between these two nodes, $a_{X_1X_2}$ is set to 0.

In the initialization stage, the proposed algorithm randomly generates initial feasible solutions of BN structure learning problem, which means that OP-PSO-DE randomly generates directed acyclic graphs and represents them as their adjacent matrices. In our previous work, the PC-PSO algorithm achieves good performances because it introduces PC algorithm to obtain structure priors to improve the initial solutions. Therefore, we still adopt this strategy in this paper to obtain initial solutions of OP-PSO-DE algorithm. PC algorithm selects an empty graph as the initial network and adds edges to the network structure through conditional independence tests. The details of PC algorithm[29] can be seen in Algorithm 1:

```
Require: undirected graph, $G$ and node set, $V$
Ensure: partial directed graph
 1: for each adjacent (a, b) in $G$ do
 2:     if the cardinal number of $A_Gab \bigcap U_Gab$ less than $n$ then
 3:         turn to the next pair of adjacent nodes in graph $G$
 4:     end if
 5:     if the cardinal number of $A_Gab$ less than $n$ then
 6:         if (a, b) is conditionally independent with the cardinal number of the subset of $A_Gab \bigcap U_Gab$ equal to $n+1$ then
 7:             delete the undirected edge of $a-b$
 8:         end if
 9:     end if
10: end for
11: output the undirected graph, $F$
12: for each triple (a, b, c) satisfying (a, b), (b, c) adjacent, but (a, c) not adjacent in graph $F$ do
13:     set the direction of $a-b-c$ as $a \rightarrow b \leftarrow c$
14: end for
15: return partial directed graph
```

Algorithm 1. PC algorithm

Where, A_Gab represents the node set (except *a* and *b*) adjacent to *a* or *b*. U_Gab represents the node set(except *a* and *b*) consisting of nodes on the acyclic undirected path between *a* and *b*.

After that, as Fig.1 shows, the proposed algorithm sorts these solutions according to their BIC scores and obtain the elite individuals. The opposite solutions of elite individuals are generated and we merge the original initial solutions with opposite solutions to evaluate them together. We determine the top *N* individuals in the merged population to be our final initial solutions.

In the iteration process, the algorithm should determine the mutation probability and crossover probabilities first to vary the initial solutions and generate offsprings. In this paper, we utilize the $\mathrm{SSRDIW_{PSO}}$[30] inertia weight strategies to update the corresponding mutation rate of our proposed algorithm. $\mathrm{SSRDIW_{PSO}}$ defines the swarm success rate(ssr) to measure the evolution extent of the population. For the *i*-th particle in the population in the *t*-th iteration, its success rate can be defined as:

$$succ_i(t) = \begin{cases} 1, & if\ BIC(P_{best,i}(t)) > BIC(P_{best,i}(t-1)) \\ 0, & otherwise \end{cases} \tag{14}$$

where $P_{best,i}(t)$ is the individual best solution of particle *i* in the *t*-th iteration. Thus, for the *t*-th iteration, the success rate of the whole population can be calculated by:

$$ssr(t) = \frac{\sum_i^N succ_i(t)}{N} \tag{15}$$

where *N* is the population size. Next, the inertia weight *w* of PSO in the *t*-th iteration, which is also referred to the mutation probability of our proposed algorithm, can be updated according to $\mathrm{SSRDIW_{PSO}}$:

$$w = (w_{start} - w_{end})\frac{MaxIt - t}{MaxIt} + w_{end} \times ssr(t-1) \tag{16}$$

$\mathrm{SSRDIW_{PSO}}$ considers the feedback parameter $ssr(t-1)$ to reserve the population information during the iteration process and adjust the inertia weight according to it.

For acceleration coefficients of PSO, which is also referred to our crossover probabilities of our proposed algorithm, we utilize the same Self-Tuned (ST) method[31] as BNC-PSO to update:

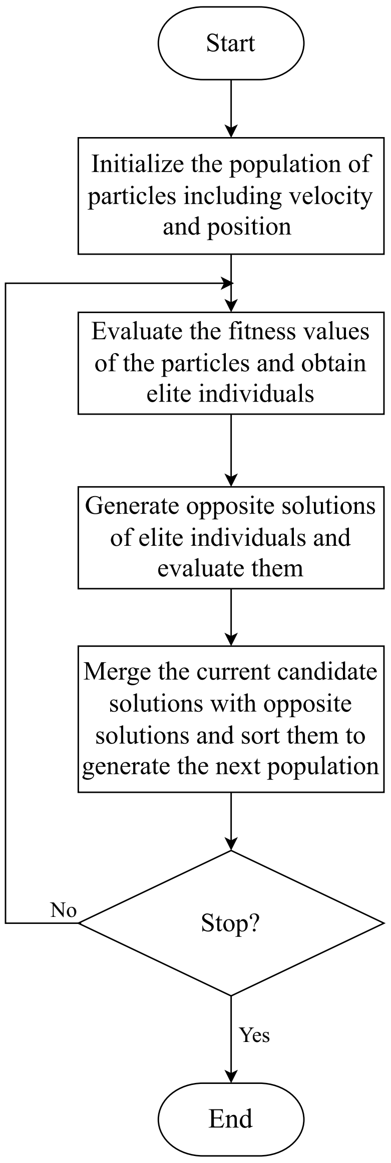
Start
Initialize the population of particles including velocity and position
Evaluate the fitness values of the particles and obtain elite individuals
Generate opposite solutions of elite individuals and evaluate them
Merge the current candidate solutions with opposite solutions and sort them to generate the next population
Stop?
No
Yes
End

Fig. 1. The executing process of opposition-based learning in our proposed algorithm. It should be noticed that we only list steps containing opposition-based learning and the whole details of our proposed algorithm can be seen below.

$$\begin{aligned} c_1 &= c_{10} - \frac{c_{10} - c_{11}}{MaxIt} * t \\ c_2 &= c_{20} - \frac{c_{20} - c_{21}}{MaxIt} * t \end{aligned} \tag{17}$$

where c_{10}, c_{20} are starting values of c_1 and c_2 in the iteration process; c_{11}, c_{21} are ending values of c_1 and c_2.

After determining the values of these parameters, the proposed algorithm executes the mutation operation and two crossover operations and generates the new generation of particles. Then, we obtain the new generated population and the elite opposite solutions. Next, the proposed algorithm merges the new generation with opposite solutions and selects top *N* particles as the next generation. Finally, the proposed algorithm updates the individual best solutions, positions and velocities of these particles. If the algorithm does not achieve the maximum of iterations, it continues to execute; otherwise, the algorithm ends and outputs the final result. All the steps and important details of the proposed algorithm mentioned above can be seen in Algorithm 3 $\sim$ 5 in the Appendix.

It should be noticed that the mutation operation and two crossover operations might introduce cycles into the DAGs, thus we adopt the property of DAGs to check cycles and remove them. In graph theory, there is an important proposerty of adjacent matrix: the elements of *q*-th power of adjacent matrix, A_{ij}^q, represent the number of paths of length n from vertex i to vertex j. In order to ensure the acyclicity of the directed acyclic graph, the trace of the q-th power of the adjacency matrix must be equal to 0. The details of this method can also be seen in Algorithm 2:

```
Require: adjacent matrix, $A$
Ensure: adjacent matrix of DAG
 1: initialize $C=A$
 2: if the diagonal elements of $C$ do not contain non-zero elements then
 3:     $C = C * A$
 4:     $i = i + 1$
 5: else
 6:     turn to line 11
 7: end if
 8: if $i < q$ then
 9:     turn to line 2
10: end if
11: if $i = q$ and non-zero elements are not contained in $C$ then
12: return $A$
13: end if
14: randomly select the corresponding nodes $V_x$ and $V_y$ of non-zero elements in $C$, delete or reverse the corresponding edges in
    $A$, and recursively execute this algorithm
15: $C=A$
16: return $A$
```

Algorithm 2. Cycle removal procedure

The sparsity of DAGs

Opposition-based learning is an effective method to achieve fast convergence of heuristic algorithms. However, it also consumes a lot of computing resources, especially for structure learning problems in BNs. So in this paper, we reduce the computation complexity of OP-PSO-DE algorithm by restricting the in-degree and out-degree of DAGs. Also, it can be found that DAGs in practical applications are sparse and BIC score is tended to choose DAGs with lower structure complexity. By this way, we can guarantee the sparsity of DAGs returned by our proposed algorithm. For different structure learning problems in BNs, the options of in-degree and out-degree can be determined according to specific application scenarios.

The time complexity of OP-PSO-DE

Since the calculation in our proposed algorithm is based on matrix, the complexity of our algorithm is $O(MaxIt * nPop * nVar^4)$ where *MaxIt* is the iteration time, *nPop* is the population size, and *nVar* is the number of random variables. In the following works, we will try our best to reduce algorithm complexity to apply our algorithm into more application scenarios. To be specific, the time complexity of cycle checks is $O(nVar^4)$, thus the time complexity of generating opposite solutions is $O(nVar^2 + L(nVar)) = O(nVar^2 + nVar^4) = O(nVar^4)$, where $L(nVar)$ is the complexity of cycle checks. Also, the time complexity of elite selection is $O(eli\_num \cdot (nVar^4 + nVar \cdot n)) = O(eli\_num \cdot nVar^4)$, where eli_num is the number of elite individuals and n is the number of samples.

Experimental results and analysis

Experimental parameters

To verify the performances of our proposed OP-PSO-DE algorithm, we conduct experiments on different discrete networks, including CANCER[32], ASIA[33] and INSURANCE[34]. To be specific, CANCER network contains 5 nodes and 4 arcs, which describes a lung cancer diagnosis application. There are two nodes, "Pollution" and "Smoker", representing the factors that affect the chance of a patient having cancer. ASIA network contains 8nodes, 8 arcs and 18 parameters, which describes the factors affect the chances of a patient getting diseases. There are two input nodes, "Asia" and "Smoke" representing the patient whether visited to Asia and smokes. And there are two output nodes representing the result of getting diseases, "Xray" and "Dyspnoea", which means that it will result in a positive X-ray and dyspnoea. The INSURANCE network contains 27 nodes, 52 arcs and 984 parameters, which describes car insurance risk estimation. It is a network for estimating the expected claim costs for a car insurance policyholder. There is one input node in this network, "Age" and three output nodes, "MedCost", "ILiCost" and "PropCost". Since Bayesian network structure learning algorithms have high requirements for data quality, this study uses benchmark networks commonly used in other papers within this research field for the experiments. This also allows for a comparison of our proposed algorithm with those in other papers.

Parameters	Connotation	Value range
F	scale number	[0,1]
σ	turbulence coefficient	[0.001,0.01]
rand	random number	[0,1]
h	random number	[1,d]

Table 1. Hyperparameters of OP-PSO-DE algorithm.

Dataset	500	1000	1500	2000
CANCER	-1052.90 ± 0	– 2129.68 ± 0	– 3173.84 ± 0.11	– 4233.08 ± 0.43
	(– **1059.01**)	(– **2131.10**)	(-**3178.94**)	(-**4233.38**)
ASIA	-1126.89 ± 3.21	-2369.95 ± 4.27	-3400.48±6.82	– 4527.01 ± 5.01
	(-**1126.75**)	(-**2366.21**)	(-**3396.32**)	(– **4522.48**)
INSURANCE	– 9461.35 ± 43.49	– 16747.60 ± 107.04	– 24715.27 ± 186.70	– 32370.27 ± 375.99
	(-**9468.28**)	(– **16340.47**)	(– **23155.05**)	(– **30024.96**)

Table 2. Experimental results of OP-PSO-DE algorithm.

All the algorithms in this section are implemented in R and our datasets are randomly sampled from these networks downloaded from Bayesian Network Repository(https://www.bnlearn.com/bnrepository/) of R package *bnlearn*. the sample sizes of these networks are 500, 1000, 1500, 2000. The swarm population of all the heuristic algorithms *nPop* are set to 50 and the iteration *MaxIt* is set to 100. The size of elite individuals in OP-PSO-DE algorithm is set to 25. Other parameters are set to $w_{start} = 0.9, w_{end} = 0.35, c_{10} = 0.84, c_{11} = 0.52, c_{20} = 0.38$ and $c_{21} = 0.81$. The specific calculation of these parameters are shown in equation (16) and (17). The sensitivity analysis of c_1, c_2 and *w* was conducted in the previous works[35,36] and it can also be seen in the following section. We also list the hyperparameters and their rationale in Table 1:

Experimental results

In this section, we show the experimental results of our proposed OP-PSO-DE algorithm with other state-of-the-art algorithms, including PC-PSO[8], BNC-PSO[17], Hybrid HPC[37], and Max-Min Hill-Climbing(MMHC)[38]. PC-PSO uses PC algorithm to generate initial solutions and improves PSO to search for the best global DAG. BNC-PSO combines PSO with Genetic Algorithm to discretize PSO to search for the best network structure. Hybrid HPC firstly reconstructs the skeleton of the network and then uses greedy hill-climbing to search for the best sturcture. MMHC firstly identifies all the potential parent nodes or child nodes of the nodes as the candidate node set, and then uses score-based algorithm to find the network structure with the highest score.

The sample sizes of these algorithms are 500, 1000, 1500 and 2000, respectively. All the results are averages of 10 times experiments. The crossover probabilities and mutation probability of PC-PSO and BNC-PSO are the same as our proposed algorithm to compare their performances. BIC score is utilized to evaluate the searching ability of these algorithms. Table 2 shows the final convergence experimental results of OP-PSO-DE on and the bold BIC scores in brackets are benchmarks. Fig. 2, Fig. 3, and Fig. 4 show the iteration process of OP-PSO-DE on CANCER, ASIA and INSURANCE networks.

To verify the convergence capacity of OP-PSO-DE algorithm, we count the iteration time required to converge and compare it with PC-PSO and BNC-PSO. Since the principles of Hybrid HPC and MMHC are different from the other three algorithms, the implementation process can not count iteration time. Therefore, Table 3 does not contain the corresponding results.

Sensitivity analysis

In this section, we conduct experiments to show the sensitivity analysis of key parameters of our proposed algorithm. We change the value of mutation probability w_{start} from 0.9 to 0.5 with the interval 0.2 and w_{end} from 0.75 to 0.35 with the interval 0.2. We change the value of crossover probability c_{10} from 1.0 to 0.68 with the interval 0.16 and c_{11} from 0.68 to 0.36 with the interval 0.16. We change the value of crossover probability c_{20} from 0.57 to 0.19 with the interval 0.19 and c_{21} from 1.0 to 0.62 with the interval 0.19.

As Tables 7, 8 and 9 show, on the CANCER network, the impacts of the mutation probability and crossover probabilities on BIC scores are very limited. In small-sample regimes, its approximation accuracy deteriorates, which can result in under-penalizing complex models and the risk of overfitting. The size of the CANCER network is very small, only containing 5 nodes and 4 edges. The algorithm is more likely to converge to a good BIC score, and it is difficult to further explore the solution space by changing the current solutions through crossover operators and the mutation operator. On the ASIA network and INSURANCE network, with the increases of w_{start} and the decreases of w_{end}, OP-PSO-DE is tended to converge to a better BIC score. Also, a lager crossover probability can usually achieve a better BIC score than the small ones.

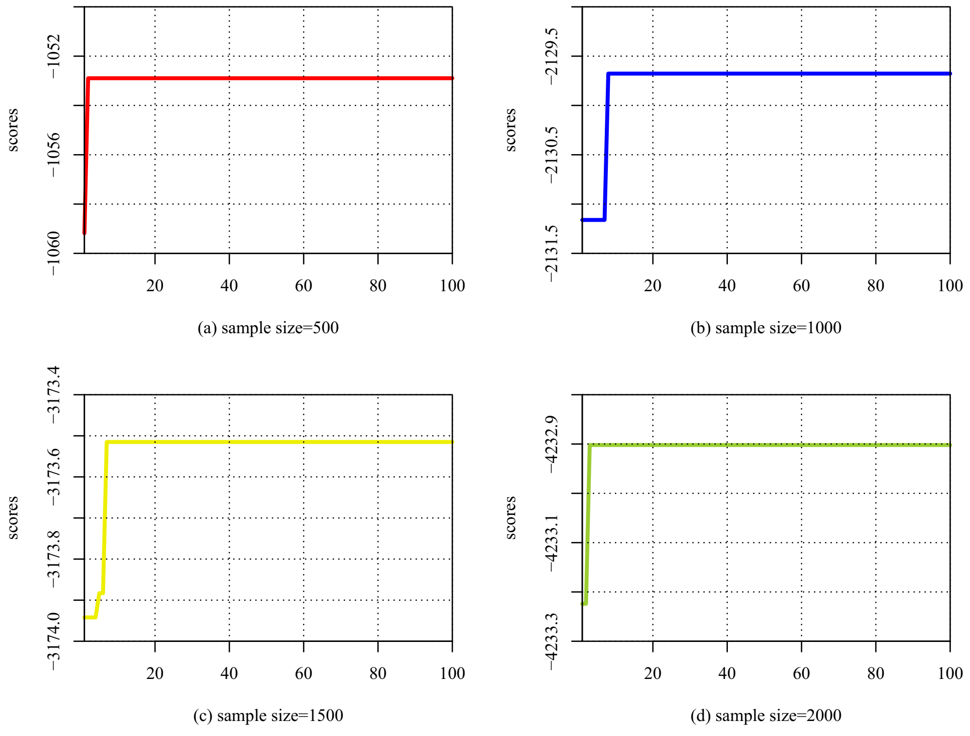

Fig. 2. The iteration process of OP-PSO-DE algorithm executes on CANCER network.

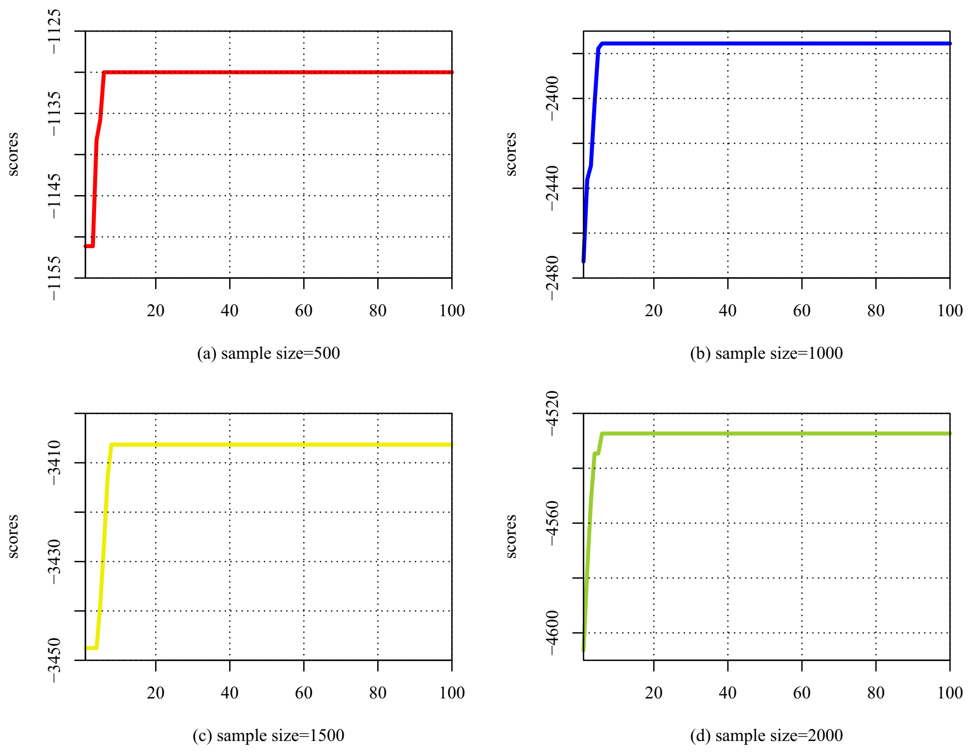

Fig. 3. The iteration process of OP-PSO-DE algorithm executes on ASIA network.

Comparison of OP-PSO-DE with other structure learning algorithms

According to Table 2, we can see that our proposed algorithm can achieve BIC scores very close to the benchmarks in the brackets. BIC score of our proposed algorithm on CANCER network is higher than the standard score, because BIC is derived under asymptotic assumptions and performs reliably when the sample size is large. With

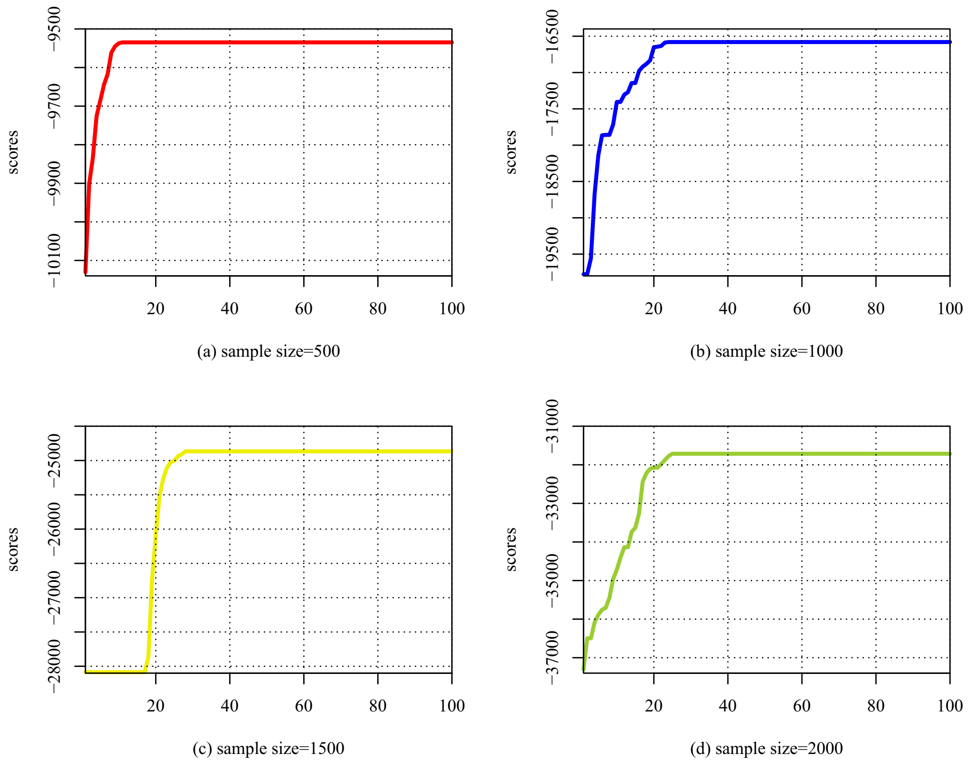

Fig. 4. The iteration process of OP-PSO-DE algorithm executes on INSURANCE network.

	CANCER				ASIA				INSURANCE			
Algorithm	**500**	**1000**	**1500**	**2000**	**500**	**1000**	**1500**	**2000**	**500**	**1000**	**1500**	**2000**
OP-PSO-DE	2.1	1.8	2.7	3.9	5.3	5.8	5.6	5.7	9.7	28.4	25.5	20.1
PC-PSO	5.7	3.6	11.2	12.3	22.6	29.3	33.1	33.1	78.3	86.3	89.8	93.3
BNC-PSO	3.3	3.3	3.0	4.9	17.4	16.9	23.4	18.5	94.9	96.7	97.9	97.5

Table 3. The number of iterations required for heuristic algorithms to converge.

	CANCER			
Algorithm	**500**	**1000**	**1500**	**2000**
OP-PSO-DE	**− 1052.90**	− 2129.68	**− 3173.84**	− 4233.08
PC-PSO	**− 1052.90**	− 2129.68	− 3173.52	− 4232.79
BNC-PSO	− 1053.01	**− 2130.13**	− 3173.74	− 4232.63
Hybrid HPC	**− 1052.90**	− 2129.68	− 3173.52	**− 4233.22**
MMHC	− 1053.45	− 2129.68	− 3173.52	**− 4233.22**

Table 4. Experimental results of different algorithms on CANCER network.

the increases of sample sizes, the standard deviations of our proposed algorithm present the trend of growth. As Table 3 shows that our proposed algorithm can converge rapidly than the other heuristic algorithms on CANCER, ASIA and INSURANCE networks, which verifies the effectiveness of opposition-based learning used in OP-PSO-DE algorithm. Fig. 2, Fig. 3, and Fig. 4 indicate that the iterations of our proposed algorithm increase with the network sizes. With the increase of the sample size on INSURANCE network, the performance of our proposed algorithm decreases. Although opposition-based learning can effectively reduce the iteration times of heuristic algorithms, premature convergence arises when sample sizes become larger.

Compared to other state-of-the-art structure learning algorithms, OP-PSO-DE algorithm achieves best BIC scores on CANCER network when sample sizes are 500 and 1500, which can be seen in Table 4. When the sample size of CANCER network is 2000, our proposed algorithm achieves BIC scores better than PC-PSO and BNC-PSO.As we can see in Table 5, our proposed algorithm achieves better BIC scores than PC-PSO, BNC-

Algorithm	ASIA			
	500	1000	1500	2000
OP-PSO-DE	**− 1126.89**	**− 2369.95**	**− 3400.48**	**− 4527.01**
PC-PSO	− 1125.24	− 2370.53	− 3401.96	− 4527.45
BNC-PSO	− 1128.86	− 2378.81	− 3409.90	− 4541.84
Hybrid HPC	− 1219.64	− 2574.37	− 3702.83	− 4901.70
MMHC	− 1208.89	− 2549.88	− 3676.24	− 4864.38

Table 5. Experimental results of different algorithms on ASIA network.

Algorithm	INSURANCE			
	500	1000	1500	2000
OP-PSO-DE	**− 9461.35**	− 16747.60	− 24715.27	− 32370.27
PC-PSO	− 9014.95	− 16823.61	− 24563.79	− 33194.194
BNC-PSO	− 8383.71	**− 16093.07**	− 23667.46	− 31127.48
Hybrid HPC	− 8324.36	− 15388.29	− 22418.60	− 29676.46
MMHC	− 8747.69	− 16006.97	**− 23248.39**	**− 30789.33**

Table 6. Experimental results of different algorithms on INSURANCE network.

Parameter		CANCER	ASIA	INSURANCE
w_{start}	0.90	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49
	0.70	− 1052.90 ± 0	− 1131.05 ± 3.71	− 9426.30 ± 255.72
	0.50	− 1052.90 ± 0	− 1132.21± 6.25	− 9266.34 ± 435.33
w_{end}	0.75	− 1052.90 ± 0	− 1135.27 ± 8.11	− 9391.06 ± 378.04
	0.55	− 1052.90± 0	− 1134.77± 9.20	− 9597.34± 252.32
	0.35	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49

Table 7. BIC scores of OP-PSO-DE for different values of w_{start} and w_{end} when sample size is 500.

Parameter		CANCER	ASIA	INSURANCE
c_{10}	1.00	− 1052.90 ± 0	− 1133.23 ± 7.15	− 9555.52 ± 276.63
	0.84	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49
	0.68	− 1052.90 ± 0	− 1134.41 ± 7.01	− 9341.43 ± 227.24
c_{11}	0.68	− 1052.90 ± 0	− 1134.24 ± 10.70	− 9426.60 ± 203.74
	0.52	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49
	0.36	− 1052.90 ± 0	− 1133.95 ± 9.14	− 9607.03 ± 288.22

Table 8. BIC scores of OP-PSO-DE for different values of c_{10} and c_{11} when sample size is 500.

Parameter		CANCER	ASIA	INSURANCE
c_{20}	0.57	− 1052.90 ± 0	− 1133.39 ± 8.33	− 9268.55 ± 271.76
	0.38	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49
	0.19	− 1052.90 ± 0	− 1136.62 ± 7.35	− 9813.12 ± 233.23
c_{21}	1.00	− 1052.90 ± 0	− 1132.17 ± 9.07	− 9424.01 ± 191.43
	0.81	− 1052.90 ± 0	− 1126.89 ± 3.21	− 9461.35 ± 43.49
	0.62	− 1052.90 ± 0	− 1135.12 ± 7.90	− 9514.82 ± 343.88

Table 9. BIC scores of OP-PSO-DE for different values of c_{20} and c_{21} when sample size is 500.

PSO, Hybrid HPC and MMHC when sample sizes are 500, 1000, 1500 and 2000. Also, in Table 6, OP-PSO-DE achieves the best BIC scores when sample size is 500 on INSURANCE network and the difference between OP-PSO-DE and the standard value is much smaller than the other four algorithms.

To verify the effect opposition-based learning, we tried to remove the module from OP-PSO-DE. On the INSURANCE network, the proposed algorithm achieved -17926.70 (true score is -16340.47) when the sample size is 1000 after 8 iterations. Although the iteration time is much smaller than OP-PSO-DE, the resulting solutions are significantly worse in terms of final score and network quality compared to other state-of-the-art algorithms. This suggests that opposition-based learning plays a crucial role in maintaining population diversity and avoiding poor local optima, especially in the early search stages.

It is worth noting that wall-clock time is not reported in this study due to variability across implementations, which may render direct comparisons unreliable. Instead, we focus on the number of iterations and score function evaluations, which typically dominate the computational cost in structure learning tasks. Our method demonstrates substantial improvements in both metrics, suggesting that the observed convergence speed is not only theoretical but also likely translates to practical time efficiency.

As experimental results show, our proposed OP-PSO-DE algorithm can achieve good performances with a rapid convergence rate. Also, since our proposed algorithm is based on heuristic algorithms, it is very easy to implement and understand.

Conclusion

In this paper, we propose a hybrid structure learning algorithm, OP-PSO-DE, which utilizes opposition-based learning to accelerate its convergence rate and combines PSO with Differential Evolution algorithm searching for the best BIC scores. For the mutation operator, we utilize the binary mutation with probability difference and $SSRDIW_{PSO}$ method to update the crossover probability. We introduce the execution process of our proposed algorithm specifically in the above sections and conduct experiments based on different networks. Experimental results show that our proposed algorithm can achieve better results compared to other state-of-the-art structure learning algorithms.

The contribution of this paper is that we introduce opposition-based learning method into the structure learning algorithm in Bayesian networks, which is verified to be an effective way to accelerate convergence process in our experiments. While tackling the structure learning problem of large networks, our proposed algorithm arises premature convergence. In particular, its performance degrades as the network scale increases, which becomes especially evident in the case of large networks. In the following researches, we plan to improve the current algorithm to avoid premature convergence and conduct our experiments on more practical application scenarios to measure the performances of OP-PSO-DE.

Data availibility

The datasets generated and analyzed during the current study are available in the Bayesian Network Repository, https://www.bnlearn.com/bnrepository/.

Received: 21 January 2025; Accepted: 19 May 2025
Published online: 27 May 2025

References

1. Villa-Blanco, C., Bregoli, A., Bielza, C., Larrañaga, P. & Stella, F. Constraint-based and hybrid structure learning of multidimensional continuous-time Bayesian network classifiers. *Int. J. Approx. Reason.* **159**, 108945. https://doi.org/10.1016/j.ijar.2023.108945 (2023).
2. Jose, S., Louis, S., Dascalu, S. & Liu, S. Transfer learning-based hybrid approach for Bayesian network structure learning. *Int. J. Artif. Intell. Tools* **31**, 2260003. https://doi.org/10.1142/S021821302260003X (2022).
3. Wang, N., Liu, H., Zhang, L., Cai, Y. & Shi, Q. An efficient skeleton learning approach-based hybrid algorithm for identifying Bayesian network structure. *Eng. Appl. Artif. Intell.* **133**, 108105. https://doi.org/10.1016/j.engappai.2024.108105 (2024).
4. Russell, S. J. & Norvig, P. *Artificial Intelligence: A Modern Approach* (Prentice-Hall Inc, 1995).
5. Lee, S. & Kim, S. B. Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning. *IEEE Trans. Knowl. Data Eng.* **32**, 1157–1166 (2020).
6. Ye, Q., Amini, A. A. & Zhou, Q. Optimizing regularized Cholesky score for order-based learning of Bayesian networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 3555–3572 (2021).
7. Alonso-Barba, J. I. *et al.* Ant colony and surrogate tree-structured models for orderings-based bayesian network learning. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15* 543–550 (Association for Computing Machinery, 2015).
8. Sun, B., Zhou, Y., Wang, J. & Zhang, W. A new pc-pso algorithm for Bayesian network structure learning with structure priors. *Expert Syst. Appl.* **184**, 5237. https://doi.org/10.1016/j.eswa.2021.115237 (2021).
9. Wang, C. et al. Finding community structure in Bayesian networks by heuristic k-standard deviation method. *Future Gener. Comput. Syst.* **158**, 556–568 (2024).
10. Yang, W.-T., Tamssaouet, K. & Dauzere-Peres, S. Bayesian network structure learning using scatter search. *Knowl.-Based Syst.* **300**, 112149 (2024).
11. He, C. et al. A novel structure learning method of Bayesian networks based on the neighboring complete node ordering search. *Neurocomputing* **585**, 127620 (2024).
12. Liu, H. et al. An improved Harris hawks optimization for Bayesian network structure learning via genetic operators. *Soft Comput.* **27**, 14659–14672 (2023).
13. Kareem, S. W. & Okur, M. C. Structure learning of Bayesian networks using elephant swarm water search algorithm. *Int. J. Swarm Intell. Res.* **11**, 19–30 (2020).
14. Awla, H. Q., Kareem, S. W. & Mohammed, A. S. A comparative evaluation of Bayesian networks structure learning using falcon optimization algorithm. *Int. J. Interact. Multimed. Artif. Intell.* **8**, 81–87 (2023).
15. Soloviev, V. P., Bielza, C. & Larranaga, P. Quantum approximate optimization algorithm for Bayesian network structure learning. *Quantum Inf. Process.* https://doi.org/10.1007/s11128-022-03769-2 (2022).

16. Wang, X., Ren, H. & Guo, X. A novel discrete firefly algorithm for Bayesian network structure learning. *Knowl.-Based Syst.* **242**, 108426. https://doi.org/10.1016/j.knosys.2022.108426 (2022).
17. Gheisari, S. & Meybodi, M. Bnc-pso: structure learning of Bayesian networks by particle swarm optimization. *Inf. Sci.* **348**, 272–289 (2016).
18. Zheng, X., Aragam, B., Ravikumar, P. K. & Xing, E. P. Dags with no tears: Continuous optimization for structure learning. In (eds. Bengio, S. *et al.*) *Advances in Neural Information Processing Systems* Vol. 31 (2018).
19. Yu, Y., Chen, J., Gao, T. & Yu, M. Dag-gnn: Dag structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (2019).
20. Lachapelle, S., Brouillard, P., Deleu, T. & Lacoste-Julien, S. Gradient-based neural dag learning. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)* (2020).
21. Heckerman, D., Geiger, D. & Chickering, D. M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **20**, 197–243 (1995).
22. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **6**, 461–464 (1978).
23. Lam, W. & Bacchus, F. Learning Bayesian belief networks: An approach based on the mdl principle. *Comput. Intell.* **10**, 269–293 (1994).
24. Kennedy, J. & Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* Vol. 4, 1942–1948 (IEEE, 1995).
25. Tizhoosh, H. R. Opposition-based learning: a new scheme for machine intelligence. In *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)* vol. 1, 695–701 (IEEE, 2005).
26. Seif, Z. & Ahmadi, M. B. Opposition versus randomness in binary spaces. *Appl. Soft Comput.* **27**, 28–37 (2015).
27. Storn, R. & Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **11**, 341–359 (1997).
28. Zhang, Y., Gong, D.-W., Gao, X.-Z., Tian, T. & Sun, X.-Y. Binary differential evolution with self-learning for multi-objective feature selection. *Inf. Sci.* **507**, 67–85 (2020).
29. Spirtes, P. & Glymour, C. An algorithm for fast recovery of sparse causal graphs. *Soc. Sci. Comput. Rev.* **9**, 62–72 (1991).
30. Adewumi, A. O. & Arasomwan, A. M. An improved particle swarm optimiser based on swarm success rate for global optimisation problems. *J. Exp. Theor. Artif. Intell.* **28**, 441–483 (2016).
31. Ghosh, S., Nath, S. & Sarkar, S. Pso algorithm with self tuned parameter for efficient routing in vlsi design. In *International Conference on Futuristic Trends in Computing and Communication* 60–63 (International Journal of Electronics and Communication Engineering, 2015).
32. Korb, K. B. & Nicholson, A. E. *Bayesian Artificial Intelligence* (CRC Press, 2010).
33. Lauritzen, S. L. & Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Stat. Soc. Ser. B* **50**, 157–194 (2018).
34. Binder, J., Koller, D., Russell, S. & Kanazawa, K. Adaptive probabilistic networks with hidden variables. *Mach. Learn.* **29**, 213–244 (1997).
35. Ratnaweera, A., Halgamuge, S. K. & Watson, H. C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evol. Comput.* **8**, 240–255 (2004).
36. Shi, Y. & Eberhart, R. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)* 69–73 (IEEE, 1998).
37. Gasse, M., Aussem, A. & Elghazel, H. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Syst. Appl.* **41**, 6755–6772 (2014).
38. Tsamardinos, I., Brown, L. E. & Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.* **65**, 31–78 (2006).

Acknowledgements
This work was supported by the Key R&D Program of Heilongjiang Province under Grant No.2022ZX01A23.

Author contributions
B.S. wrote the first draft of the manuscript and conducted the experiments. X.Z. and J.G. edited and revised the manuscript. J.J. and D.L. analyzed the results. All authors reviewed the manuscript.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-03267-2.

Correspondence and requests for materials should be addressed to J.J.