Hindawi
Scientific Programming
Volume 2022, Article ID 4391071, 14 pages
https://doi.org/10.1155/2022/4391071

Research Article

Research on Dynamic Programming Strategy of Bayesian Network Structure Learning

Ruohai Di,[1] **Ye Li**,[1] **Tingpeng Li**,[2] **Peng Wang**,[1] **and Chuchao He**[1]

[1]*School of Electronics and Information Engineering, Xi'an Technological University, Xi'an 710021, China*
[2]*State Key Laboratory of Complex Electromagnetic Environment Effects on Electronic and Information System (CEMEE), Luoyang 471003, China*

Correspondence should be addressed to Chuchao He; 1543687340@qq.com

Received 18 July 2021; Revised 14 March 2022; Accepted 23 April 2022; Published 20 May 2022

Academic Editor: Shujuan Jiang

Bayesian network structure learning based on dynamic programming strategy can be used to find the optimal graph structure compared with approximate search methods. The traditional dynamic programming method for Bayesian network structure learning is a depth-first-based strategy, which is inefficient. We proposed two methods to solve this problem. First, the dependency constraints were used to prune the process of calculating redundancy scores. The constraints were obtained by the conditional independence test from the observed data sets. However, it was difficult to guarantee the accuracy of the constraints, which may have led to a decrease in the accuracy of the method. Second, we proposed a breadth-first-based strategy, which enhanced efficiency greatly while also ensuring global optimality. Experimental results showed that on the standard network data sets, compared with the dynamic programming based on depth-first search (DFSDP) algorithm, dynamic programming based on constraints (CBDP) could reduce the average running time by 57.10% and that dynamic programming based on breadth-first search (BFSDP) could reduce the average running time by 50.02%. On the UCI data sets, compared with DFSDP, CBDP reduced the average running time by 40.71%, and BFSDP reduced the average running time by 81.78%.

1. Introduction

As a graphical modeling tool, Bayesian networks (BNs) [1] provide a method for expressing the causal relationships between variables, which can be used to obtain knowledge concealed in the data. A Bayesian network is a directed acyclic graph (DAG), in which the nodes correspond to the variables in the domain and the edges correspond to direct probabilistic dependencies. A key feature of Bayesian network research is structure learning, which aims to construct network structures automatically using the observed data sets and prior knowledge. Formally, the structure of the network represents a set of conditional independence assertions: each variable is conditionally independent of its non-descendants given its parents [2].

Learning a BN from observational data is an important problem that has been studied extensively over the past decade [3]. It can be used to automatically construct decision support systems and is used for inferring possible causal relations under certain conditions [4]; the edges in the BN graph have causal semantics. BN has been used widely in reliability analysis [5, 6], medical diagnosis [7, 8], gene analysis [9], fault diagnosis [10], language recognition [11], and index sensitivity analysis problems [12].

Structure learning methods of BN can be categorized into two main types according to the learning accuracy: (1) approximation methods and (2) exact methods. Approximation methods are easy to sink into the local optimal, while exact methods can find the optimal graph structure in the whole solution space. However, the latter method may be limited by the network size and is suitable for occasions with high accuracy requirements. In this study, we focused mainly on the exact methods. The main contributions of this paper follow.

(1) We proposed a dynamic programming algorithm based on dependency constraints. We used a priori

constraints to guide the DFSDP algorithm to calculate the node family score and find the optimal parent node set, reduce the number of the calculations of the family score to improve the operating efficiency of the algorithm, and effectively reduce the time and space costs.

(2) We proposed a dynamic programming strategy based on breadth-first search (BFSDP). This method avoided the backtracking operation that needs to be performed in the iterative process of the DFSDP algorithm and thus improved the efficiency of the algorithm.

This paper is organized as follows. In Section 2, we discuss the existing literature on structure learning according to two types of search methods. The basic knowledge of the BN structure learning and dynamic programming strategy is introduced in Section 3. The structure learning method based on the depth-first strategy is introduced in Section 4.1, and the two proposed algorithms are introduced in Section 4.2. The performance of the proposed methods is shown in Section 5. Finally, we conclude and outline our future work in Section 6.

2. Related Work

In this section, we review the algorithms for learning BN structures according to the different kinds of methods that have been proposed to date.

(1) Conditional independence (CI) test methods—such as the SGS method [13]; the PC algorithm [14]; and the drafting, thickening, and thinning three-step method [15]—are representative methods. It is difficult for this kind of method, which treats the BN as a graph structure that encodes the independent relationship between variables, to decide whether two variables are independent or conditionally independent, and the time to execute CI tests grows exponentially with the number of the variables.

(2) Scoring and searching methods include the following two steps: model selection and model optimization. Model selection requires choosing a criterion, which is named as the scoring function. Currently, the following are some of the frequently used scoring functions: Bayesian information criterion (BIC) score [16], minimum description length (MDL) score [17], Bayesian Dirichlet (BD) score [18], implicit score (IS) [18], and mutual information test (MIT) score [19]. Modeling optimization aims to find the network structure that obtains the highest score according to the selection criterion. Usually heuristic searching algorithms are used, such as the bee colony algorithm [20], the genetic algorithm [21], the fish swarm algorithm [22], and the particle swarm optimization [23, 24]. The methods based on scoring and searching are intended to balance the accuracy, robustness, and sparsity. Nevertheless, these methods are easy to sink into the local optimal, which is considered to be their congenital weakness.

(3) Mixed search methods use CI tests to reduce the graph search space and then obtain the optimal network structure through scoring and searching. Tsamardinos et al. [25] proposed the max-min hill-climbing (MMHC) algorithm, which is a typical mixture method. It sets up parent and children sets for each node to build the network skeleton using the MMPC algorithm [26] and then finds the optimal structure using hill-climbing algorithm based on the frame.

(4) Among the few articles on optimal search, Ott and Miyano [27] proposed the first exact algorithm. While investigating the problem of exact model averaging, Koivisto and Sood [28] proposed another algorithm that also learned optimal graphs in a similar way. Singh and Moore [29] proposed a recursive implementation method that is less efficient in terms of calculation but has the advantage that potential branch-pruning rules can be applied. Silander and Myllymaki [30] provided a practically efficient implementation of the search and empirically demonstrated that optimal graphs could be learned up to $n = 29$. Yuan et al. [31] proposed an A^* search algorithm for learning optimal BN. Malone et al. [32] proposed a memory-efficient implementation of the dynamic programming algorithm, which leveraged the layered structure of the dynamic programming graphs representing the recursive decomposition of the problem to reduce the memory requirements of the algorithm from $O(n2^n)$ to $O(C(n, n/2))$, where $C(n, n/2)$ is the binomial coefficient. This kind of method can obtain the optimal network structure but has the following two limitations: (1) the scale of the learned network is limited to 30; (2) the efficiency of the algorithm is insufficient as most of the algorithms are based on depth-first search strategy.

In this study, we focused on how to enhance the efficiency of the exact structure learning algorithm.

3. Theoretical Basis of Bayesian Network

In this section, we briefly introduce the basics of BN and the concepts that are used to learn the structure of these networks.

3.1. Bayesian Network

Definition 1. (Bayesian network). A Bayesian network $\mathbf{B}(\mathbf{G}, \boldsymbol{\theta})$ is composed of a network structure $\mathbf{G}(\mathbf{V}, \mathbf{E})$ and a set of network parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\}$, in which the parameters $\boldsymbol{\theta}$ are the set of conditional probabilities of joint distribution of network nodes based on structure $\mathbf{G}$ decomposition.

Definition 2. (structure of Bayesian network). The structure of BN is a DAG, whose nodes are variables $X_1, X_2, \ldots, X_n \in \mathbf{V}$, and for any $X_i \in \mathbf{V}$, after given the set of all its parent nodes $\prod_{X_{t_i}} \subseteq \{X_{t_1}, \ldots, X_{t_{i-1}}\}$ (or Π_i), X_i is independent of all of its non-descendant nodes.

From this definition, we can see that the structure of BN is a DAG satisfying Markov condition, from which we can directly obtain a series of local independence relations in the node set $\mathbf{V}$. According to the topological order $[X_{t_1}, X_{t_2}, \ldots, X_{t_n}]$ and Markov condition, the joint distribution of Bayesian network nodes can be decomposed into the following:

$$P(X_1, X_2, \ldots X_n) = \prod_{i=1}^{n} P\left(X_{t_i} | X_{t_1}, \ldots X_{t_{i-1}}\right) = \prod_{i=1}^{n} P(X_i | \Pi_i), \tag{1}$$

where $P(X_i|\Pi_i)$ denotes the conditional probability that quantifies the parent-child relationship in BN structure. Set r_i to be the number of states of node X_i; then $q_i = \prod_{X_t \in \prod_i} r_t$ denotes the number of state combinations of the parent node set Π_i; and $P(X_i|\Pi_i)$ forms a matrix with size $r_i \times q_i$, which is called the conditional probability table (CPT) and is recorded as $\boldsymbol{\theta}_i$. The set of CPT of all nodes $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is called the parameters of BN. For brevity, we use the symbol θ_{ijk} to represent conditional probability $P(X_i = k/\Pi_i = j)$, where $k \in \{1,2, \ldots, r_i\}$ is the state of X_i, and $j \in \{1,2, \ldots, q_i\}$ denotes the combined state of the parent node set Π_i.

Definition 3. (topological order). Let $\mathbf{G}(\mathbf{V}, \mathbf{E})$ be a BN structure and $[X_{t_1}, X_{t_2}, \ldots, X_{t_n}]$ be a complete permutation of node set $\mathbf{V}$. If $t_i < t_j$ holds for any edge (parent-child relationship) $X_{t_i} \longrightarrow X_{t_j} \in E$, we call $[X_{t_1}, X_{t_2}, \ldots, X_{t_n}]$ a topological ordering (TO) of the Bayesian network.

Definition 4. (valid path). Let $\mathbf{G}(\mathbf{V}, \mathbf{E})$ be a BN structure and $X_i \rightleftarrows \cdots \rightleftarrows X_j$ be a path in $\mathbf{G}$. Given the condition set $\mathbf{Z} \in \mathbf{V}$, the path $X_i \rightleftarrows \cdots \rightleftarrows X_j$ is valid when the following conditions are satisfied:

(i) For any V-structure $X_{t-1} \longrightarrow X_t \leftarrow X_{t+1}$ on the path $X_i \rightleftarrows \ldots \rightleftarrows X_j$, a child of X_t or X_t belongs to $\mathbf{Z}$.

(ii) Other nodes on the path (including X_i and X_j) do not belong to $\mathbf{Z}$.

Definition 5. (d-separation). Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be three disjoint subsets of the node set $\mathbf{V}$ of Bayesian network $\mathbf{B}(\mathbf{G}, \boldsymbol{\theta})$. If there is no effective path between any $x \in \mathbf{X}$ and $y \in \mathbf{Y}$ given $\mathbf{Z}$, then $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-partitioned, denoted as $\mathrm{Dsep}_{\mathrm{G}}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$.

Theorem 1. *(global Markov independence). Let* $\mathbf{X}$*,* $\mathbf{Y}$*, and* $\mathbf{Z}$ *be three disjoint subsets of node set* $\mathbf{V}$ *of Bayesian network* $\mathbf{B}(\mathbf{G}, \boldsymbol{\theta})$*. If* $\mathbf{X}$ *and* Y *are d-separated by* $\mathbf{Z}$*, then given* $\mathbf{Z}Z$*,* $\mathbf{X}$ *and* $\mathbf{Y}$ *are conditionally independent (i.e.,* $\mathbf{X}, \mathbf{Y}|\mathbf{Z}$*).*

Corollary 1. *In a BN, given the Markov boundary of any node X, that is, the set of parent node, child node, and spouse node (other parent nodes of child node), X is independent of all of the other nodes in the network.*

From Theorem 1, it can be seen that the independence relation contained in the BN structure is a subset of the independence relation contained in the joint distribution which can be decomposed according to the BN structure.

3.2. Scoring Function. The scoring function is used to measure the fitting degree of the BN structure and data. The following is an introduction to commonly used scoring functions.

The CH score is the first Bayesian score function. For BN $\mathbf{B}$ and data sets $\mathbf{D}$, the CH score is as follows:

$$s_{\mathrm{CH}}(B|D) = P(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{\left(N_{ij} + r_i - 1\right)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{2}$$

where N_{ijk} denotes the frequency of the family state combination corresponding to the network parameter θ_{ijk} in the data set $\mathbf{D}$, and $N_{ij} = \sum_k N_{ijk}$; $P(\mathbf{B})$ is the prior distribution of $\mathbf{B}$; and the rest of (2) is the likelihood function.

The BD score, which introduces a reliable theoretical basis into the CH score, is a more commonly used:

$$s_{\mathrm{BD}}(B|D) = P(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\alpha_{ij}\right)}{\Gamma\left(N_{ij} + \alpha_{ij}\right)} \prod_{k=1}^{r_i} \frac{\Gamma\left(N_{ijk} + \alpha_{ijk}\right)}{\Gamma\left(\alpha_{ijk}\right)}, \tag{3}$$

where $\Gamma(x)$ is a gamma function over a real field and satisfies the property $\Gamma(x+1) = x\Gamma(x)\Gamma(x+1) = x\Gamma(x)$, and α_{ijk} is the prior knowledge of the family state combination corresponding to the network parameter θ_{ijk}. The larger the value, the more likely the family state combination, and $\alpha_{ijk} = \sum_k \alpha_{ijk}$.

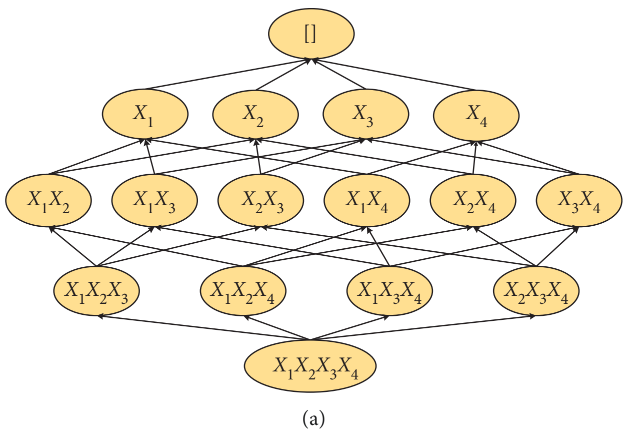
When all $\alpha_{ijk} = 1$, BD score degenerates to CH score. If the likelihood equivalence constraint is given, the likelihood equivalent BD score (BDe) is obtained. If we assume that the probability of all family state combinations is the same (i.e., $\alpha_{ijk} = \alpha/r_i q_i$), we get the uniform distribution likelihood equivalent BD score (BDeu), but its structure learning result is very sensitive to the equivalent sample size α.

The BIC score is another common scoring function, which can be regarded as the maximum likelihood function with penalty term; that is,

$$s_{\mathrm{BIC}}(B|D) = P(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{\log N}{2} \sum_{i=1}^{n} q_i (r_i - 1), \tag{4}$$

where N denotes the sample size. The first term of the BIC score is the kernel of the maximum likelihood function, and the second term is the model complexity penalty.

3.3. Exact Search Based on Dynamic Programming. Exact search methods find the optimal solution of the problem in



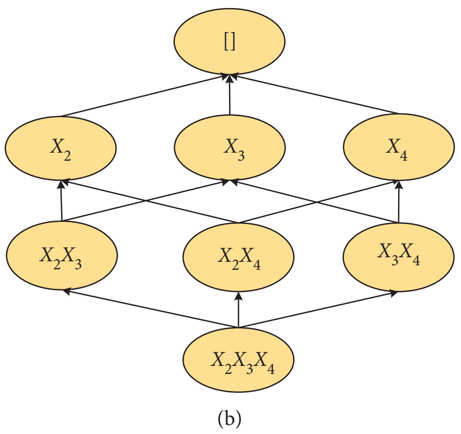


FIGURE 1: Order graph and parent graph. (a) Order graph of four nodes. (b) Parent graph of node X_1.

the global space. Thus, they can obtain the optimal network structure, which belongs to the equivalent class of the true network model. As this study mainly researched exact search methods based on the dynamic programming, we have introduced the basic theory of the dynamic programming in detail.

Dynamic programming methods traverse all the node orderings and obtain the global optimal solution. When learning BN structure by dynamic programming, each Bayesian network structure has at least one leaf node. In addition, the score criteria used are decomposable. Supposing the set of variables contained in the problem domain is **V** and the optimal BN structure has a leaf node X, the state transition equation of dynamic programming is as follows:

$$\max \text{Score}(V) = \max_{X \in V} \left\{ \max \text{Score}\left(\frac{V}{X}\right) + \max \text{Score}\left(\frac{X, V}{X}\right) \right\}, \tag{5}$$

$$\text{Bestscore}\left(\frac{X, V}{X}\right) = \max \text{Score}\left(\frac{X, V}{X}\right) = \max_{Pa(X) \subseteq V \setminus \{X\}} \text{Score}(X, Pa(X)). \tag{6}$$

Formula (5) together with (6) linked a structure to its substructures. The optimal structure based on the remaining node set $\mathbf{V} \setminus X$ was recursively constructed through the previous process, until only one node was left. All of the structures in **V** construct a hash diagram, which showed the whole process of the dynamic programming. Because the hash diagram contained node ordering information of the network, the diagram was named as an order graph in the literature [28]. Another similar graph was named the parent graph, which contained a candidate parent set for each node. Figure 1 shows the presentation of the order graph and the parent graph.

Figure 1(a) is the order graph of four nodes. The order graph starts from the full set **V**, and each layer in the graph represents a state of dynamic programming. The transition from one state to another is a programming element, and one variable is excluded in each programming element until

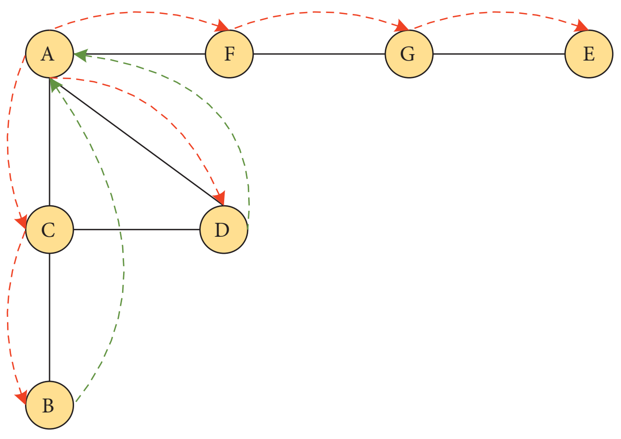


FIGURE 2: A strategy of depth-first search.

all of the variables are eliminated. Each node ordering of the BN network corresponds to the opposite direction of the path in the graph. Figure 1(b) is the parent graph of node X_1, and each node in the graph is a candidate parent set of node X_1, which stores the corresponding optimal parent set and the family score, that is, Bestparents$(X_1, .)$ and Bestscore$(X_1, .)$.

4. Dynamic Programming Algorithm Based on Depth-First Search Strategy

4.1. Search Strategy. The DFSDP strategy started from the bottom of the order graph and searched the paths that connect a full node set and the empty node set from bottom to top. The transition from state $\boldsymbol{S}$ to other states corresponded to the selection of the leaf nodes for node set **S** in the network. Thus, a path from $\{X_1, X_2, \ldots, X_n\}$ to $\{\}$ corresponded to the removal sequence of the leaf nodes, which was a reverse network node ordering. If the value of score (**S**) could be obtained in a certain path, it was stored. In this way, duplicate computation was avoided as the subsequent paths also may have arrived at the node set **S**.

The core idea of the depth-first search is as follow: Consider that all of the vertices in the graph were not accessed in the initial state. We started from a certain vertex

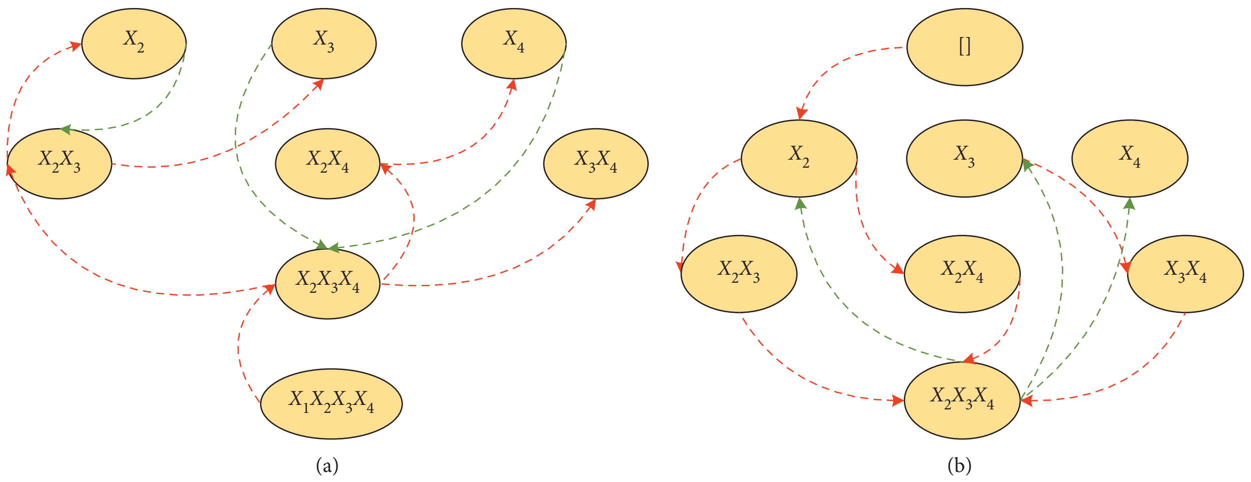


FIGURE 3: The depth-first search strategy for four nodes. (a) Order graph of four n. (b) Parent graph of node X_1.

v and accessed the node v first. The graph was searched from each of its unaccounted adjacent points in turn, until all of the vertices in the graph that had paths to v were accessed. If other vertices were not visited, then we selected another unvisited vertex as the starting point and repeated this process, until all of the vertices in the graph were accessed. Based on the depth-first search strategy, the dotted line in Figure 2 shows an example of one of the search strategies. The red lines in Figure 2 represent the algorithm's forward search, and the green lines represent the backtracking operation. When the access order of the node is $A \succ C \succ B \succ (A) \succ D \succ (A) \succ F \succ G \succ E$, the search strategy is as follows.

Figure 3, from left to right, shows the search strategy for the order graph of four nodes when taking X_1 as the leaf node under the depth-first search principle. The red lines in Figure 3 represent the algorithm's forward search, and the green lines represent the backtracking operation. For other nodes as leaf nodes or other parent graphs, the search order is similar. The order of access to nodes by this search strategy is as follows.

4.2. Problems and Solutions. The DFSDP algorithm can find the optimal network structure globally. Because the algorithm adopts the depth-first strategy, the backtracking operation was executed repeatedly during the process of iteration, which made the efficiency low. There are two ways to solve this problem: (1) restricting the size of the candidate parent set of each node by constraints; (2) using the breadth-first search strategy instead of the depth-first search strategy. These two improvements are described in the following sections.

4.2.1. Research on Dynamic Programming Algorithms Based on Dependency Constraints

(1) Dependency constraints. Computing node family scores and finding the optimal parent set under the guidance of prior constraints can reduce temporal and spatial cost effectively. We named the method CBDP (dynamic programming based on constraints).

Theorem 2. *Given sample data set* $\mathbf{D}$ *on a set of variables* $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$*, if conditional independence* $Ind(X_i, X_j | X_k)$ *holds, the statistic* $U^2_{ij|k}$ *approximately obeys the* χ^2 *distribution of degree of freedom* $(r_i - 1)(r_j - 1)r_k$.

In Theorem 2, $U^2_{ij|k} = 2\sum_{a,b,c} N^{abc}_{ijk} \log N^{abc}_{ijk} N^c_k / N^{ac}_{ik} N^{bc}_{jk}$, N^{abc}_{ijk} denotes the number of samples in $\mathbf{D}$, that is, $X_i = a, X_j = b, X_k = c$, and r_i denotes the number of values of variable X_i.

Definition 6. (Dependency coefficient). Given sample data set $\mathbf{D}$ on a set of variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, $c_{ij\alpha} = \min_{k \neq i,j} \{U^2_{ij|k} - \chi^2_{ij|k,\alpha}\}$ is defined as the dependency coefficient between variables X_i and X_j. $U^2_{ij|k}$ denotes the U^2 statistic of variable X_i and X_j under the condition of a given variable X_k. $\chi^2_{ij|k,\alpha}$ denotes the value of χ^2 distribution with significance level α and degree of freedom $(r_i - 1)(r_j - 1)r_k$.

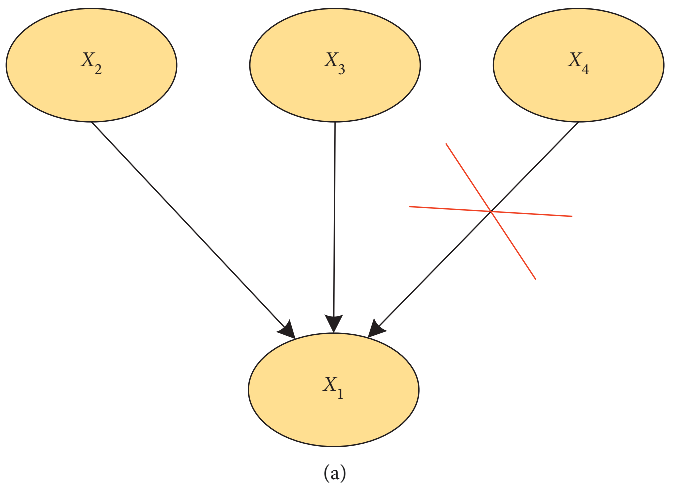
According to the χ^2 hypothesis test, if $c_{ij\alpha} > 0$, then X_i and X_j are interdependent whether or not additional nodes are added. If $c_{ij\alpha} < 0$, then X_i and X_j may be independent given some variable $X_k \in \mathbf{X}$.

Lemma 1. *Given sample data set* $\mathbf{D}$ *on a set of variables* $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$*, variables* X_i *and* X_j *are locally conditionally independent at the significance level* α *if and only if there exists a variable* $X_k \in \mathbf{X}$ *where* $U^2_{ij|k} < \chi^2_{ij|k,\alpha}$ *holds.*

Lemma 1 can be obtained directly by Definition 6 and by the χ^2 hypothesis test. According to Lemma 1, variables X_i and X_j are locally conditionally independent at the significance level α if and only if $c_{ij\alpha} < 0$. Variables X_i and X_j are globally conditionally independent at the significance level α if and only if $\forall k \neq i, j$, $U^2_{ij|k} < \chi^2_{ij|k,\alpha}$ holds.

Matrix $\mathbf{C} = [c_{ij}]$ is defined as the dependent coefficient matrix of variable set $\mathbf{X}$, where

$$c_{ij} = \begin{cases} c_{ij\alpha}, & \text{when } X_i \prec X_j, \\ 0, & \text{else}. \end{cases} \tag{7}$$



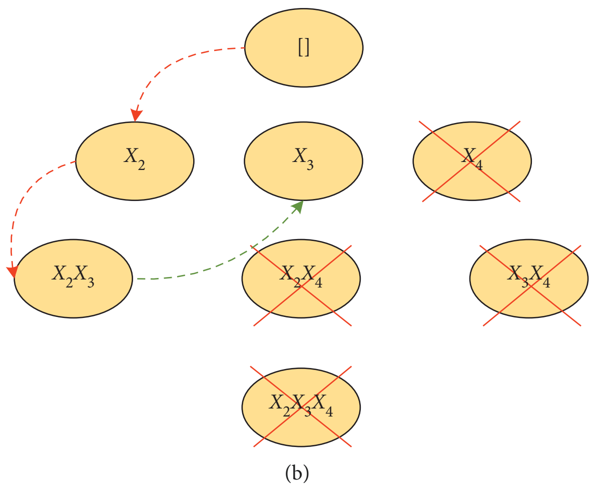


Figure 4: Pruning of parent graph of X_1 by dependency constraints. (a) Dependency constraint graph of node X_1. (b) The search strategy for parent graph based on dependency constraints.

We took the dependent coefficient matrix **C** as the prior constraints and restricted the size of the candidate parent set of each node in the network. Then, the number of family score calculations could be reduced effectively, and the efficiency of the algorithm was improved as a result.

(2) Search strategy.

Theorem 3. *Assume that* **C** *is the dependent coefficient matrix of the network.* ***CPa****(X) refers to the set where variables are interdependent with X in C.* ***CNPa****(X) infers the set where variables are independent with X in C. U refers to the candidate parent set of X. PSbest (X) refers to the optimal parents set of X. Then, the following relationships hold:*

(1) $\forall i, j$, *if* $c_{ij} \neq 0$, *then* $X_i \in \mathbf{CPa}(X_j)$; *if* $c_{ij} = 0$, *then* $X_i \in \mathbf{CNPa}(X_j)$.

(2) $CNPa(X_j) \subseteq U$, $CPa(X_j) \cap CNPa(X_j) = \varnothing$, $PS_{best}(X_j) \subseteq CPa(X_j) \subseteq U$.

Theorem 3 can be obtained by the definition of the dependency coefficient. According to Theorem 3, the formula for calculating family scores can be rewritten from

$$\text{BestScore}(\mathbf{V}, X) = \max_{\mathbf{PS} \subseteq \mathbf{U}} \text{NodeScore}(X|\mathbf{PS}) \quad (8)$$

to

$$\text{BestScore}(V, X) = \max_{PS \subseteq CPa(X)} \text{NodeScore}(X|\text{PS}). \quad (9)$$

This transformation can reduce the total score calculation times from $n2^{n-1}$ to $\sum_{X\in V} 2^{|CPa(X)|}$.

Supposing that $c_{2,1} > 0$, $c_{3,1} > 0$, and $c_{4,1} = 0$ hold in the dependent coefficient matrix—that is, $X_2 \in \mathbf{CPa}(X_1)$, $X_3 \in \mathbf{CPa}(X_1)$, and $X_4 \in \mathbf{CPa}(X_1)$, which are shown in Figure 4(a)—the search strategy of parent graph of X_1 is shown in Figure 4(b).

In general, the total number of family scores needed to be calculated in standard DP is $n2^{n-1}$ and 2^{n-1} for each node. When considering the dependency constraints, the total number of family scores needed to be calculated in DP was $\sum_{X\in V} 2^{|CPa(X)|}$ and $2^{|CPa(X)|}$ for each node. $|\mathbf{CPa}(X)|$ was far less than 2^{n-1} in general. Thus, the efficiency of the algorithm after adding dependency was enhanced greatly, as calculating the score was one of the most time-consuming parts. We next compared the candidate parent set (CPS) and number of family scores (NFS) of the Asia network with those needed to be considered with and without the dependency constraints in DP.

From Table 1, we also can find that the candidate parent set obtained by the dependency constraints was not exactly the same as that of the standard BN, which meant that the dependency constraints mined from the sample data were not always accurate.

4.2.2. Research on Dynamic Programming Algorithms Based on Breadth-First Search Strategy

(1) Search strategy. As mentioned earlier, the CBDP method restricts the size of candidate parent sets of each node by constraints, which then reduces the time for calculating the family score and improves the efficiency of the algorithm. However, the dependency constraint is mined from sample data, whose accuracy cannot be guaranteed, which may reduce the accuracy of the algorithm to a certain extent. As a result, the optimal network structure cannot be obtained. To address this problem, we proposed a breadth-first search strategy to replace the depth-first search strategy to avoid the backtracking operation. In this way, the search efficiency could be improved while guaranteeing the global optimum.

The core idea of the breadth-first search is as follows: We started from a vertex v in the graph and visited the adjacent points of v once after visiting v. Then, we accessed the adjacent points of these points in turn and followed the principle that “the adjacent points of the first visited vertex take precedence over those of the second visited vertex,” until all of the adjacent vertices of the visited vertices in the graph were accessed. If other vertices were not visited, we

Table 1: The CPS and NFS of the Asia network and those needed to be considered with and without priors in DP.

Node	Standard		Without priors		With priors	
	CPS	NFS	CPS	NFS	CPS	NFS
1	$\{X_2, X_3\}$	4	$\{\mathbf{X} \backslash X_1\}$	128	$\{X_2, X_3\}$	4
2	$\{X_1, X_7\}$	4	$\{\mathbf{X} \backslash X_2\}$	128	$\{X_1, X_7\}$	4
3	$\{X_1, X_6\}$	4	$\{\mathbf{X} \backslash X_3\}$	128	$\{X_1, X_6, X_8\}$	8
4	$\{X_5\}$	2	$\{\mathbf{X} \backslash X_4\}$	128	$\{X_5\}$	2
5	$\{X_4, X_6\}$	4	$\{\mathbf{X} \backslash X_5\}$	128	$\{X_4, X_6, X_8\}$	8
6	$\{X_3, X_5, X_7, X_8\}$	16	$\{\mathbf{X} \backslash X_6\}$	128	$\{X_3, X_5, X_8\}$	8
7	$\{X_2, X_6\}$	4	$\{\mathbf{X} \backslash X_7\}$	128	$\{X_2\}$	2
8	$\{X_6\}$	2	$\{\mathbf{X} \backslash X_8\}$	128	$\{X_3, X_5, X_6\}$	8

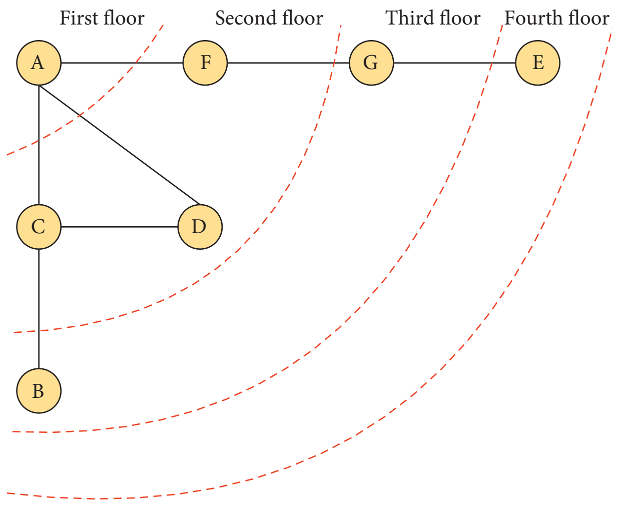


Figure 5: The breadth-first search case diagram.

selected another unvisited vertex as the starting point and repeated this process, until all of the vertices in the graph were accessed. Figure 5 shows an example search strategy when the access order of the node is $A \succ C \succ D \succ F \succ B \succ G \succ E$.

Figure 6, from left to right, shows the search strategy for the order graph of four nodes when taking X_1 as the leaf node under the breadth-first search principle. The red lines in Figure 6 represent the algorithm's forward search. For other nodes as leaf nodes or other parent graphs, the search order is similar. The order of access to the nodes by this search strategy is as follows.

(2) Search method. From the analysis in the preceding section, we found that, compared with the depth-first search strategy, the breadth-first search strategy was more efficient because we did not have to execute the backtracking operation. Therefore, we proposed the BFSDP algorithm. The execution steps of the BFSDP algorithm are shown in Algorithm 1.

The algorithm may repeatedly query the family score and the optimal network structure score of each combination during the execution process. Therefore, we constructed a hash table corresponding to the node set and its label to improve the query efficiency. Representing node sets in binary encoding, for network with n nodes, a binary array b with n digits was set. For a set $\boldsymbol{U}$ in the node sequence diagram, if $X_i \in \mathbf{U}$, then position i of b was set to 1; otherwise, it was set to 0. We then converted it to decimal labels by hash function $\sum_i b_i \cdot 2^{i-1} + 1\,(i \in \{1, 2, \ldots, n-1\})$. In scoring lookup, we used the decimal label instead of the node set. Taking a network of four nodes as an example, the hash table is shown in Table 2.

Step 1: Calculate the whole family score of each node in the network and store them in the hash table, as follows:

$$\begin{aligned} &\forall X \in V,\ \text{Nodescore}_{PS \subseteq V/\{X\}}(X|PS), \\ &\forall X \in V,\ \left\{\text{Nodescore}_{PS \subseteq V/\{X\}}(X|PS), PS\right\} \longrightarrow \text{HashTable}. \end{aligned} \tag{10}$$

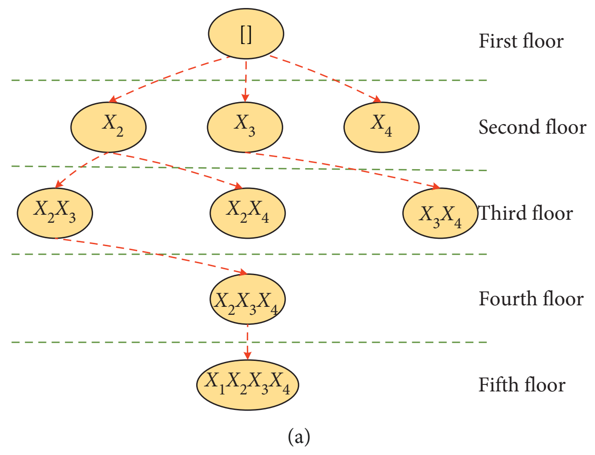
Step 2: For each node in the network, obtain the best parent set and the corresponding score and store them in the hash table according to the breadth-first search strategy in their parent graph, as follows:

$$\begin{aligned} &\forall X \in V,\ \text{HashTable} \longrightarrow \text{Nodescore}_{PS \subseteq V/\{X\}}(X|PS), \\ &\text{BestScore}(V, X) = \max_{PS \subseteq V-\{X\}} \text{NodeScore}(X|PS), \\ &\forall X \in V,\ PS_{\text{best}}(V, X) = \arg\max_{PS \subseteq V-\{X\}} \text{NodeScore}(X|PS), \\ &\{\text{BestScore}(V, X), PS_{\text{best}}(V, X)\} \longrightarrow \text{HashTable}. \end{aligned} \tag{11}$$

Step 3: Obtain the optimal network structure score and the optimal leaf node of each node combination and store them in the hash table according to the breadth-first search strategy in order graph, as follows:

$$\begin{aligned} &\forall S \subseteq V,\ \text{HashTable} \longrightarrow \text{BestScore}(S, X), \\ &\text{Score}(S) = \max_{X \in S}\left[\text{Score}\left(\frac{S}{X}\right) + \text{BestScore}(S, X)\right], \\ &\{\text{Score}(S), S\} \longrightarrow \text{HashTable}, \\ &\forall S \subseteq V,\ \text{Leaf}(S) = \arg\max_{X \in S}\left[\text{Score}\left(\frac{S}{X}\right) + \text{BestScore}(S, X)\right]. \end{aligned} \tag{12}$$

Step 4: Starting from the full node combination, extract the optimal leaf node and the optimal parent node set of the corresponding leaf node to construct part of the network structure. Update the current node set and repeat the process until the node set is empty. The



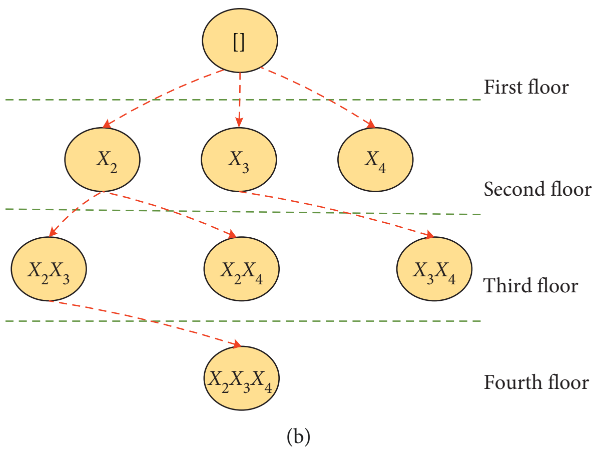


FIGURE 6: The breadth-first search strategy for four nodes. (a) Order graph of four n. (b) Parent graph of node X_1.

(1) Obtain family scores of all nodes and store them.
(2) Obtain the best parent set and the corresponding score of each and store them.
(3) Obtain the optimal network structure score and the optimal leaf nodes of each node combination and store them.
(4) Construct the network structure.

ALGORITHM 1: Execution steps of the BFSDP algorithm.

TABLE 2: Hash table of four node sets.

Node	Binary coding							
	000	001	010	011	100	101	110	111
1	[]	4	3	34	2	24	23	234
2	[]	4	3	34	1	14	13	134
3	[]	4	2	24	1	14	12	124
4	[]	3	2	23	1	13	12	123
Label $\sum_{i=1}^{n-1} b_i \cdot 2^{i-1} + 1$	1	2	3	4	5	6	7	8

```
Input: The best parent sets of all nodes ($\forall X \in V, \mathrm{PS}_{\mathrm{best}}(V, X)$), number of nodes ($n$), and best leaf node of each combination of nodes ($\forall S \subseteq V, \mathrm{Leaf}(S)$).
Output: The optimal network structure ($G$).
Set $G = \varnothing$, nodes = $[1, \ldots, 1]$, order = $\varnothing$, raw = $\varnothing$;
**For** $m = n$ **to** 1 in steps of $-1$ **do**
    $X$ = index of **nodes** in Hash Table, calculated by $\sum_i b_i \cdot 2^{i-1} + 1\ (i \in \{1, 2, \ldots, n-1\})$;
    $Y = Leaf(X)$;
    The $m$-th position in **order** is set to be $Y$;
    The $Y$-th position in **nodes** is set to be 0;
    Set **raw** to be the best parent sets of $Y$, which is $\mathrm{PS}_{\mathrm{best}}(\mathrm{nodes} \cup Y, Y)$;
    **G** (**raw**, $Y$) $\leftarrow 1$;
End for
Return $G$
```

ALGORITHM 2: The pseudocode of Step 4 in the BFSDP algorithm.

Table 3: The setting of experimental parameters.

Number	Data sets	Evaluation criteria	Number of data sets	Analysis interval	Execution times	Compared algorithms
1	Sampled from networks (https://www.bnlearn.com/bnrepository)	Time consumption (s)	8	1000, 2000, 5000, 10000	1	DFSDP, CBDP, BFSDP
2	Sampled from networks	BIC score	8	1000, 2000, 5000, 10000	1	DFSDP, CBDP, BFSDP
3	UCI database (https://archive.ics.uci.edu/ml/data%20sets.html)	Time consumption (s)	14	Fixed sample size	1	DFSDP, CBDP, BFSDP
4	UCI database	BIC score	14	Fixed sample size	1	DFSDP, CBDP, BFSDP

Table 4: Time-consuming comparison between three algorithms under standard network data sets (s).

BNs	Nodes	Edges	Tns	Sample sizes	DFSDP	CBDP	BFSDP
Weather	4	4	8	1000	0.1935	0.0590	0.1742
				2000	0.2016	0.0899	0.1800
				5000	0.2120	0.1593	0.1865
				10000	0.2326	0.2240	0.1948
Cancer	5	4	10	1000	0.3348	0.0511	0.2391
				2000	0.3620	0.1077	0.2487
				5000	0.3768	0.1854	0.2693
				10000	0.4308	0.3494	0.2905
Earthquake	5	4	10	1000	0.3331	0.0849	0.2470
				2000	0.3493	0.1122	0.2487
				5000	0.3749	0.1615	0.2673
				10000	0.4234	0.2642	0.2981
Survey	6	6	14	1000	0.7828	0.1668	0.3391
				2000	0.7871	0.2790	0.3882
				5000	0.8565	0.5817	0.4355
				10000	1.0047	1.0419	0.4961
Asia	8	8	16	1000	7.8807	1.6342	1.3848
				2000	8.4305	1.7328	1.5450
				5000	8.5538	2.1903	1.8037
				10000	9.2984	2.9253	2.1439
Sachs	11	17	33	1000	$1.02e+003$	464.0321	14.55
				2000	$1.03e+003$	289.0895	20.38
				5000	$1.05e+003$	238.4937	27.91
				10000	$1.07e+003$	230.4499	38.11
Child	20	25	60	1000	OT	OT	$2.1348e+005$
				2000			$2.5035e+005$
				5000			OT
				10000			OT
Insurance	27	52	89	1000	OM	OM	OM
				2000			
				5000			
				10000			

pseudocode of Step 4 in the BFSDP algorithm is given in Algorithm 2.

5. Experiments

5.1. Experimental Setup. When sample data are generated by sampling from benchmark networks, each network generated 10 sets of sample data with fixed sample size and executed the algorithm once based on each set of sample data. Therefore, the experimental results listed in this section were the average of 10 experiments. All of the experimental environments in this section were Windows 10, Inter® Core$^{\mathrm{TM}}$ i5-6500 CPU @3.20 GHz, RAM 4.00 GB, using the MATLAB R2014a software platform. The setting of experimental parameters is given in Table 3.

5.2. Experimental Results. The experimental results of the time-consuming comparison of the three algorithms under standard network data sets are shown in Table 4 and

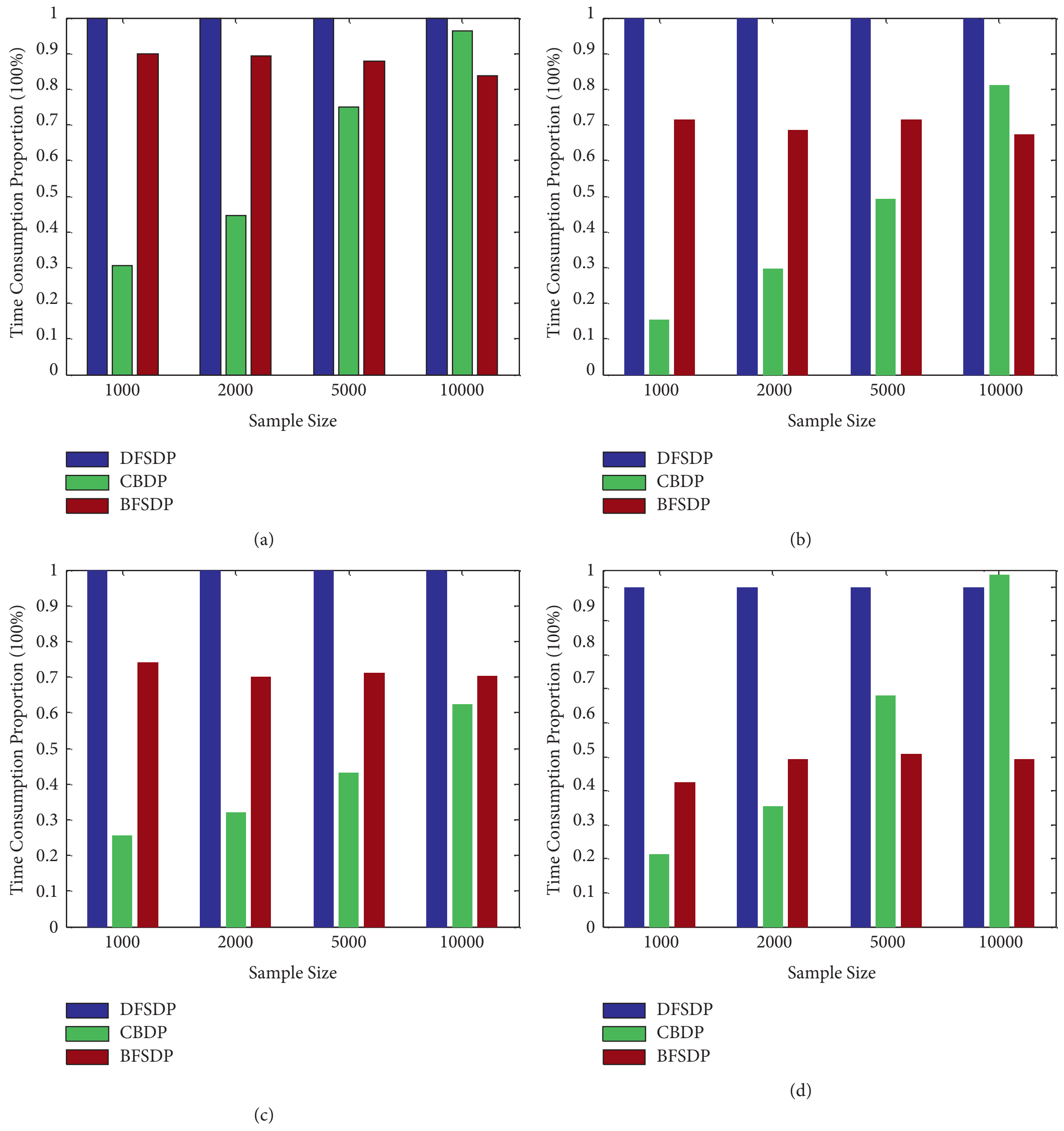
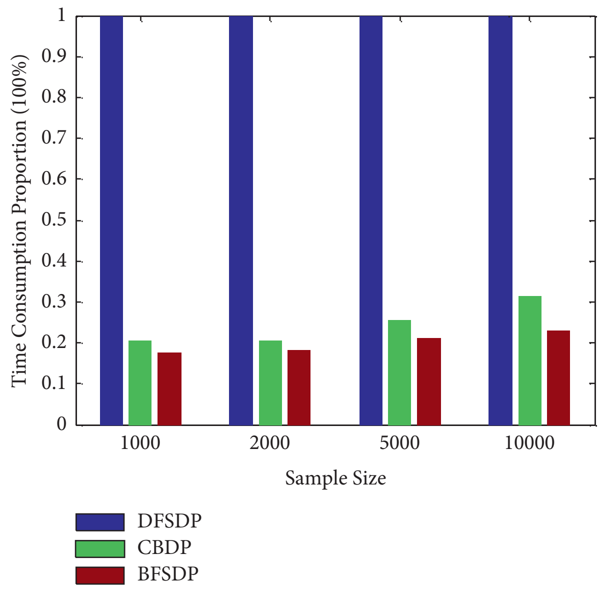
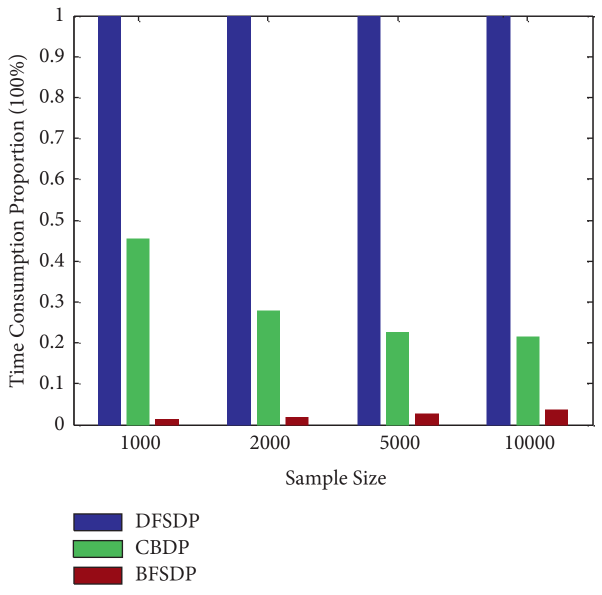


Figure 7: Continued.



(e)



(f)

FIGURE 7: Time-consuming comparison of the three algorithms under standard network data sets. (a) Weather. (b) Cancer. (c) Earthquake. (d) Survey. (e) Asia. (f) Sachs.

TABLE 5: Accuracy comparison between three algorithms under standard network data sets.

BNs	Nodes	Edges	Tns	Sample sizes	DFSDP	CBDP	BFSDP
Weather	4	4	8	1000	−984.7736	−984.7736	−984.7736
				2000	$-3.79e+003$	$-3.79e+003$	$-3.79e+003$
				5000	$-9.57e+003$	$-9.57e+003$	$-9.57e+003$
				10000	$-1.91e+004$	$-1.91e+004$	$-1.91e+004$
Cancer	5	4	10	1000	$-1.04e+003$	$-1.04e+003$	$-1.04e+003$
				2000	$-4.18e+003$	$-4.18e+003$	$-4.18e+003$
				5000	$-1.04e+004$	$-1.04e+004$	$-1.04e+004$
				10000	$-2.10e+004$	$-2.10e+004$	$-2.10e+004$
Earthquake	5	4	10	1000	−191.4748	−199.6867	−191.4748
				2000	−905.1324	−905.1324	−905.1324
				5000	$-2.23e+003$	$-2.23e+003$	$-2.23e+003$
				10000	$-4.42e+003$	$-4.42e+003$	$-4.42e+003$
Survey	6	6	14	1000	$-2.00e+003$	$-2.00e+003$	$-2.00e+003$
				2000	$-8.05e+003$	$-8.05e+003$	$-8.05e+003$
				5000	$-2.00e+004$	$-2.00e+004$	$-2.00e+004$
				10000	$-3.95e+004$	$-3.95e+004$	$-3.95e+004$
Asia	8	8	16	1000	$-1.10e+003$	$-1.11e+003$	$-1.10e+003$
				2000	$-4.55e+003$	$-4.55e+003$	$-4.55e+003$
				5000	$-1.11e+004$	$-1.11e+004$	$-1.11e+004$
				10000	$-2.25e+004$	$-2.25e+004$	$-2.25e+004$
Sachs	11	17	33	1000	$-3.87e+003$	$-5.68e+003$	$-3.87e+003$
				2000	$-1.46e+004$	$-1.83e+004$	$-1.46e+004$
				5000	$-3.60e+004$	$4.12e+004$	$-3.60e+004$
				10000	$-7.20e+004$	$8.00e+004$	$-7.20e+004$

Figure 7. The corresponding experimental results of the accuracy comparison are shown in Table 5. The experimental results of the time-consuming comparison of the three algorithms under UCI standard data sets are shown in Table 6. The corresponding experimental results of the accuracy comparison are shown in Table 7. “OT” in Tables 4, 6, and 7 indicates that the execution time of the algorithm exceeded the upper limit. We set the time limit in this study to 3 days. “OM” indicates that the storage space required by the algorithm exceeded the memory of the computer. “Nodes”

Table 6: Time-consuming comparison between three algorithms under UCI data sets (s).

Data sets	Nodes	Sample sizes	Tns	DFSDP	CBDP	BFSDP
monks_T	7	432	19	2.4193	0.4570	0.7090
Car	7	1728	27	2.6733	1.5168	0.8333
Diabetes	9	768	113	731.78	746.5755	334.7031
Abalone	9	4177	245	OM	OM	OM
Nursery	9	12960	35	51.01	25.680	8.1019
Contracep	10	1473	46	227.25	122.8150	12.0520
Wine	14	178	95	$2.45e+005$	$2.068e+005$	169.1900
Heart	14	270	82	$2.45e+005$	$1.190e+005$	225.0510
Australian	15	690	108	OT	OT	$1.06e+003$
Zoo	17	101	42	OT	OT	$8.2036e+003$
pen_A	17	7494	170	OM	OM	OM
letter_A	17	15000	282	OM	OM	OM
Segment	20	2310	427	OM	OM	OM
German	25	1000	148	OM	OM	OM

Table 7: Accuracy comparison between three algorithms under UCI data sets.

Data sets	Nodes	Sample sizes	Tns	DFSDP	CBDP	BFSDP
monks_T	7	432	19	$-2.68e+003$	$-2.88e+003$	$-2.68e+003$
Car	7	1728	27	$-1.36e+004$	$-1.36e+004$	$-1.36e+004$
Diabetes	9	768	113	$-1.22e+004$	$1.79\ e+004$	$-1.22e+004$
Abalone	9	4177	245	OM	OM	OM
Nursery	9	12960	35	$-1.26e+005$	$-1.26e+005$	$-1.26e+005$
Contracep	10	1473	46	$-1.53e+004$	$-2.47e+004$	$-1.53e+004$
Wine	14	178	95	$-2.88e+003$	$-5.81e+003$	$-2.88e+003$
Heart	14	270	82	$-4.29e+003$	$-7.96e+003$	$-4.29e+003$
Australian	15	690	108	OT	OT	$-1.11e+004$
Zoo	17	101	42	OT	OT	−638.3446

denotes the number of nodes in the network, "Edges" denotes the number of edges of the network, and "Tns" denotes the total number of states for all of the nodes. The confidence level of the CBDP algorithm was set to be 0.05 when computing dependency constraints [33]. The DFSDP algorithm and the CBDP algorithm both exceeded the time limit on all of the sample data sets when learning the Child network. All of three algorithms exceeded the memory limit on all of the sample data sets when learning the Insurance network. Thus, we did not compare the learning accuracy of these two networks in Table 5.

Based on the execution time overhead of DFSDP algorithm, the ratio of execution time of the CBDP algorithm and the BFSDP algorithm overhead to baseline is shown in Figure 7.

According to the time-consuming comparison results, because the DFSDP algorithm adopted the depth-first search strategy, which required a repetitive backtracking operation, the execution efficiency of this algorithm was the slowest among all of the data sets. Compared with the DFSDP algorithm, the CBDP algorithm constructed a parent graph according to dependency constraints, but the algorithm was still executed under the depth-first strategy, and the efficiency was enhanced significantly. The CBDP algorithm could reduce the running time by −3.70% to 84.74% (57.10% in average) compared with the DFSDP algorithm. The BFSDP algorithm adopted a breadth-first search strategy, and no backtracking operation was required. Its efficiency was higher than that of the DFSDP algorithm on all of the data sets. The BFSDP can reduce the running time by 9.97% to 98.57% (50.02% in average) compared with DFSDP. The average efficiency improvement percentage of the CBDP algorithm was higher than that of the BFSDP algorithm because in small-scale network learning, the efficiency improvement of the CBDP algorithm was more significant. When the scale of the network was small (nodes less than 6), the efficiency of the CBDP algorithm was higher than that of the BFSDP algorithm. If the scale of the network becomes larger, the BFSDP algorithm will be more efficient.

When the scale of the network was small, the difference in efficiency between the depth-first search strategy and breadth-first search strategy was not obvious, and the effect of constraints on the efficiency of the algorithm was more significant. As the scale of the network grew, improvements in the efficiency of the algorithm made by the search strategy were more significant than those made by the constraints. When the scale of the network structure was further enlarged (larger than 20), only the BFSDP algorithm could find the optimal network within the time limit, whereas the DFSDP algorithm and CBDP algorithm both exceeded the time limit defined in this study. When the number of nodes in the network was greater than 26, the storage space required by all three algorithms exceeded the computer memory.

According to the accuracy comparison results, the DFSDP algorithm and the BFSDP algorithm always found the optimal network structure on all of the data sets and under different sample sizes. When the scale of the network was small (nodes less than 6) and the sample size was sufficient (size more than 1000), the CBDP algorithm also found the optimal network structure. When the scale of the network grew larger, the CBDP algorithm was less accurate than the other two algorithms. The score of the structure obtained by CBDP algorithm reduced from 0.91% to 46.77% (17.14% in average). As the CBDP algorithm pruned the search process by dependency constraints, even though the accuracy of the dependency constraints may not have been guaranteed, the larger the scale of the network, the higher the accuracy of constraints required (which was also true when a larger sample size was required). Thus, when learning large-scale networks, the accuracy of the CBDP algorithm decreased.

The conclusions of the analysis based on the time-consuming comparison and the accuracy comparison results under the UCI data sets were similar to those obtained under standard network data sets. In terms of efficiency, the CBDP algorithm reduced the running time by -2.02% to 81.11% (40.71% in average) compared with the DFSDP algorithm. The BFSDP algorithm reduced the running time by 54.26% to 99.93% (81.78% in average) compared with the DFSDP algorithm. In terms of accuracy, the score of the structure obtained by the CBDP algorithm was reduced by 7.46% to 101.74% (60.58% in average). Note that when the sample size was small relative to the scale of the network structure, the accuracy of the CBDP algorithm was usually poor. The main reason was that the sample size was insufficient to support the accuracy of the dependency constraints on the scale of the network under this circumstance. In addition, although some data sets had fewer than 27 nodes, as the number of value states of each node was large, this led to an increase in the number of family scores needed to be stored, and the memory still exceeded the limit.

6. Conclusions and Future Work

To address the problem that the traditional dynamic programming methods based on depth-first search strategy are inefficient, we proposed to prune the process of calculating redundancy scores by dependency constraints. Because it was difficult to guarantee the accuracy of the constraints, this led to a decrease in the accuracy of the method. Then, we proposed a breadth-first based strategy, which enhanced the efficiency significantly while also ensuring the global optimality. The experiments comparing the three algorithms verified the validity of the proposed CBDP algorithm and the BFSDP algorithm.

In this study, priors were integrated into the construction of parent graph, and the parent graph was pruned with priors, which enhanced the search efficiency. The reduction in the space complexity of the algorithm, however, was insufficient. Even with the addition of priors, it was still impossible to learn the large-scale network structures. Although the space complexity of the algorithm could be effectively reduced by pruning the order graph with priors, it would be ideal to learn the network structure with more nodes. Future work will focus on extending the learning scale based on prior constraints.

Data Availability

The data used in the experiments are 14 datasets in the UCI database and sampled from the 8 standard networks.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are grateful to Prof. Wang and Dr. Guo for helpful discussions. This work was supported by the National Key Laboratory Fund (CEMEE2020Z0202B) and the Shaanxi Science Foundation (2020JQ-816 and 20JK0680).

References

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Series in Representation and Reasoning*, Morgan Kaufmann Press, Burlington, Massachusetts, USA, 1988.

[2] N. Friedman and D. Koller, *Being Bayesian about Network Structure," Uncertainty In Artificial Intelligence*, pp. 201–210, Elsevier Science, Amsterdam, Netherlands, 2000.

[3] C. L. Hu, "A review of Bayesian networks," *Journal of Hefei University*, vol. 23, no. 1, pp. 33–40, 2013.

[4] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, Springer, New York, NY, USA, 1993.

[5] X. Pan, D. Zuo, and W. Zhang, "Research on human error risk evaluation using extended Bayesian Networks with hybrid data," *Reliability Engineering & System Safety*, vol. 209, Article ID 107336, 2021.

[6] B. Sun, Y. Li, and Z. Wang, "A combined physics of failure and bayesian network reliability analysis method for complex electronic systems," *Process Safety and Environmental Protection*, vol. 148, no. 3, pp. 698–710, 2021.

[7] K. Yu, L. Liu, and J. Li, "Multi-Source causal feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 9, pp. 2240–2256, 2020.

[8] S. McLachlan, K. Dube, and G. A. Hitman, "Bayesian Networks in Healthcare: Distribution by Medical Condition," *Artificial Intelligence in Medicine*, vol. 107, Article ID 101912, 2020.

[9] S. Lax, N. Sangwan, D. N. Smith et al., "Bacterial colonization and succession in a newly opened hospital," *Science Translational Medicine*, vol. 9, no. 391, pp. 6500–6513, 2017.

[10] Z. Wang, Z. Wang, and S. He, "Fault detection and diagnosis of chillers using Bayesian network merged distance rejection and multi-source non-sensor information," *Applied Energy*, vol. 188, pp. 200–214, 2017.

[11] Q. Xiao, M. Qin, and P. Guo, "Multimodal fusion based on LSTM and a couple conditional hidden markov model for Chinese sign language recognition," *IEEE Access*, vol. 7, Article ID 112268, 2019.

[12] C. C. He and X. G. Guo, "BNSobol method for accuracy sensitivity analysis of helicopter fire control system," *Journal of Aeronautics*, vol. 37, no. 10, pp. 3110–3120, 2016.

[13] P. Spirtes, C. Glymour, and R. Scheines, “Causality from Probability,” *Evolving Knowledge In Natural And Artificial Intelligence*, 1989.

[14] P. Spirtes and C. Glymour, “An Algorithm for Fast Recovery of Sparse Causal Graphs,” *Social Science Computer Review*, vol. 9, pp. 62–72, 1990.

[15] J. Cheng, D. A. Bell, and W. Liu, “Learning Belief Networks from Data: An Information Theory Based Approach,” in *Proceedings of the International Conference on Information and Knowledge Management*, pp. 325–331, Las Vegas, Nevada, USA, November 1997.

[16] J. Suzuki, “Approximating Discrete Probability Distributions with Causal Dependence Trees,” in *Proceedings of the 2010 International Symposium On Information Theory And its Applications*, pp. 100–105, IEEE, Taichung, Taiwan, Octomber 2010.

[17] F. Gregory, G. F. Cooper, and E. Herskovits, “A Bayesian method for the induction of probabilistic networks from data,” *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.

[18] L. Bouchaala, A. Masmoudi, and F. Gargouri, “Improving algorithms for structure learning in Bayesian Networks using a new implicit score,” *Expert Systems with Applications*, vol. 37, no. 7, pp. 5470–5475, 2010.

[19] J. Ji, H. Wei, and C. Liu, “An artificial bee colony algorithm for learning Bayesian networks,” *Soft Computing*, vol. 17, no. 6, pp. 983–994, 2013.

[20] L. M. D. Campos, “A scoring function for learning bayesian networks based on mutual information and conditional independence tests,” *Journal of Machine Learning Research*, vol. 7, no. 7, pp. 2149–2187, 2006.

[21] C. Contaldi, F. Vafaee, and P. C. Nelson, “Bayesian network hybrid learning using an elite-guided genetic algorithm,” *Artificial Intelligence Review*, vol. 293, pp. 1–28, 2018.

[22] T. Guo and F. Lin, “Bayesian network structure learning based on hybrid genetic fish swarm algorithm,” *Journal of Zhejiang University*, vol. 48, no. 1, pp. 130–135, 2014.

[23] G. Li, L. Xing, and Z. Zhang, “A new bayesian network structure learning algorithm mechanism based on the decomposability of scoring functions,” *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 7, pp. 1541–1551, 2017.

[24] S. Gheisari and M. R. Meybodi, “BNC-PSO: structure learning of Bayesian networks by Particle Swarm Optimization,” *Information Sciences*, vol. 348, pp. 272–289, 2016.

[25] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, “The max-min hill-climbing Bayesian network structure learning algorithm,” *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.

[26] I. Tsamardinos, S. Aliferis, and A. Statnikov, “Time and sample efficient discovery of Markov blankets and direct causal relations,” in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 673–678, New York, NY, USA, August 2003.

[27] S. Ott and S. Miyano, “Finding optimal gene networks using biological constraints,” *Genome informatics. International Conference on Genome Informatics*, vol. 14, pp. 124–133, 2003.

[28] M. Koivisto and K. Sood, “Exact bayesian structure discovery in bayesian networks,” *Journal of Machine Learning Research*, vol. 5, no. 5, pp. 549–573, 2004.

[29] A. P. Singh and A. W. Moore, *Finding Optimal Bayesian Networks by Dynamic Programming*, Carnegie University, Pittsburgh, PA, USA, 2005.

[30] T. Silander and P. Myllymaki, “A Simple Approach for Finding the Globally Optimal Bayesian Network structure,” in *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pp. 445–452, AUAI Press, Cambridge, MA, July 2006.

[31] C. Yuan, B. Malone, and X. Wu, “Learning optimal Bayesian networks using A^* search,” in *Proceedings of the international joint conference on artificial intelligence*, pp. 2186–2191, AAAI Press, Barcelona, Spain, July 2011.

[32] B. Malone, C. H. Yuan, and E. A. Hansen, “Memory-efficient dynamic programming for learning optimal bayesian networks,” in *Proceedings of the Twenty- Fifth AAAI Conference on Artificial Intelligence*, pp. 1057–1062, AAAI Press, San Francisco, California, August 2011.

[33] S. Behjati and H. Beigy, “Improved K2 algorithm for Bayesian network structure learning,” *Engineering Applications of Artificial Intelligence*, vol. 91, no. 3, Article ID 103617, 2020.