scientific reports

Check for updates

OPEN

An improved greedy equivalent search method based on relative entropy

Xiaohan Liu[1], Qi Feng[1], Ziyi Yang[1,5], Shuying Wu[1,5], Xiaoguang Gao[1]✉, Yuqing Yang[2], Chuchao He[3] & Jia Ren[4]

As a well-known Bayesian network structure learning algorithm in equivalence class space (E-space), Greedy equivalence search (GES) is used in many fields. However, it encounters high complexity when searching for information from an empty graph. If the initial graph of GES is an equivalence class containing the strongest dependencies instead of an empty graph, its performance will be significantly improved. In this study, we propose a three-phase algorithm to establish the initial graph. First, we design a measure based on relative entropy to evaluate the relation between variables. Then, the variables are connected based on the previously designed metrics and the resulting graph is transformed into E-space. Finally, the resulting graph is used as the initial graph of GES for E-space optimization. We compare the proposed algorithm with GES in efficiency and accuracy, and the results show that our algorithm improves the efficiency and accuracy of GES. Furthermore, extensive comparisons are designed to compare our method with other state-of-the-art methods on benchmarks and real data about COVID-19 pandemic in the UK.

Keywords Bayesian networks, Structure learning, Greedy equivalent search, Relative entropy

Discovering the causal mechanisms behind data is a fundamental challenge in the field of artificial intelligence. Bayesian networks (BNs)[1], as a type of probabilistic graphical model, provide an intuitive representation of the causal relationships among random variables. Hence, BNs have been applied in a wide variety of domains in the past decades, such as prediction of protein-protein[2], explainability of chemical sciences[3], time series forecasting[4], Earth systems[5], and telecom-fraud risk warnings[6].

BNs encode the variables and independent relationships between variables as nodes and edges between nodes, respectively, which forms a directed acyclic graph (DAG) that becomes the graphical structure of a BN. The process of recovering the DAG corresponding to a BN is called BN structure learning (BNSL). Existing BNSL methods can be divided into function-based, constraint-based (CB), score-and-search (S&S), and hybrid algorithms. The function-based methods describe the relationship between variables using the given function, such as the linear non-Gaussian acyclic model (LiNGAM)[7], post-nonlinear (PNL)[8], or additive noise models[9], among others. These methods usually perform well in the case of continuous variables that satisfy their given assumptions. However, when the precise function does not apply in the case of discrete variables, identifying the direction of causality becomes intractable[10]. CB approaches generally use a series of hypothesis tests to determine the conditional independence relationships between variables. The most popular CB methods include PC[11], IAMB[12], and their improved versions or variants[13–15]. S&S approaches rely on well-defined scoring functions such as BIC/MDL[16,17], BDeu[18], and so on, to transform the BNSL problem into a task of searching for the DAG with the highest score. S&S methods can be further divided into two classes of algorithms, according to their different search strategies: exact search algorithms and local search algorithms. Although the former can find globally optimal solutions, such as GOBNILP[19,20], A*[21], and their variants[22,23], they hardly scale beyond 60 nodes; thus, local search algorithms are becoming increasingly popular. BNSL methods using local search typically search for optimal structures in three spaces: DAG space (D-space), equivalent class (EC, introduced in "Equivalence class" section) space (E-space), and ordering space (O-space). D-space methods operate on individual edges in the DAG until a locally optimal score is reached, such as Hill-climbing (HC)[24] and its more complex version[25,26]. However, the number of candidate structures in D-space grows super-exponentially as the

[1]School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China. [2]School of Automation, Northwestern Polytechnical University, Xi'an 710129, China. [3]School of Electronics and Information Engineering, Xi'an Technological University, Xi'an 710021, China. [4]School of Information and Communication Engineering, Hainan University, Haikou 570228, China. [5]Ziyi Yang and Shuying Wu contributed equally to this work. ✉email: cxg2012@nwpu.edu.cn

number of variables increases. The number of ECs is generally less than the number of DAGs, and some of the inefficient operations of D-space can be avoided in E-space, so E-space is much simpler than D-space (For more details, refer to Section 2.3 in[27].)[27,28]. The most representative algorithm in E-space is greedy equivalence search (GES)[29], which updates the current EC to another EC with a higher score. Order-based search (OBS)[30] provides the third search space for BNSL, and it uses a topological order to denote a DAG. Although some excellent O-space methods have been proposed[31,32], O-space is rooted in consistency rules for node order, which provides easier optimization but also limits its performance[31,32]. Naturally, there are approaches to combine CB and S&S algorithms, and the most representative form is Max-min HC (MMHC)[33]. However, these algorithms do not perform better than CB and S&S methods[34]. Therefore, this study focuses on E-space.

Since GES was proposed, researchers have studied how to improve its performance in the past two decades. Chickering designed two clever operators and refined the structure of GES to make it a baseline algorithm for E-space[27,35], upon which subsequent E-space algorithms have been proposed. Nielsen et al. slightly improved the performance of GES by making a tradeoff between randomness and greediness[36]. Chen et al. combined ordering and EC to apply the genetic algorithm to E-space[37]. Zhang et al. used the maximal information coefficient to determine the draft and modified the draft to obtain a higher score[38]. Alonso et al. restricted the search space of the first phase of GES to improve efficiency[39]. Ramsey et al. refined the score cache and parallelized the search phases, significantly speeding up GES[40]. Alonso et al. modified the changes in scoring at each step to incorporate some excellent metaheuristic algorithms, improving accuracy[41]. Nandy et al. proposed a hybrid E-space algorithm by combining GES and CB methods for high-dimensional continuous data[42]. Liu et al. recently designed two frameworks to transfer the local optimum of E-space to other spaces, which improves the accuracy of GES[43]. Liu et al. proposed a hybrid algorithm based on mutual information for discrete data[44]. Laborda et al. used GES as a local solver for distributed parallel optimization[45]. Chen et al. developed an algorithm to find the k-best EC using dynamic programming and model averaging[46]. Chen et al. used A* to search for the globally optimal EC on networks with fewer than 30 variables[47].

Zheng et al. recently transformed the BNSL problem into a continuous optimization process[48], which serves as a novel way to discover the DAG. Many works have used different deep learning methods to improve performance from Zheng's perspective[49–52]. However, some comparison results show that these methods generally perform poorly on discrete data[25,49].

In this study, we focus on improving the performance of GES in terms of both efficiency and accuracy on discrete data. When GES starts searching from an empty graph, the search space is very complex. Naturally, a high-level intuition is that the efficiency will be improved significantly if GES searches for the optimal EC from an excellent initial graph. Therefore, we first design a metric to measure the degree of dependence between variables based on relative entropy. Then, a framework containing connection rules according to the quantified dependency strengths is proposed. Finally, following these linking rules, the variables are connected and converted into EC as the initial graph of GES for optimization. Our framework can also simultaneously improve the local optimum of the forward phase of GES. We conduct extensive comparisons using the proposed method, GES, and other state-of-the-art algorithms to reveal the performance of the proposed method.

The remainder of this paper is organized as follows: "Preliminaries" section provides the information about graph theory, BN, and EC. "Methods" section describes how GES works, and proposes our novel algorithm with a new metric and introduces it in detail. "Results" section compares the proposed method with GES and other state-of-the-art algorithms on well-known benchmarks and a real dataset. "Discussion" section summarizes the paper and gives future directions for our research.

Preliminaries

Graph theory

A graph $G = (\mathrm{V}, \mathrm{E})$ is composed of sets of edges E and nodes V. If two arbitrary nodes $X, Y \in \mathrm{V}$ are connected by an undirected edge $X - Y$, they are *neighbors*, and the set of the neighbors of X is Ne_X. We say that Y is a *parent* of X if there is a directed edge pointing from Y to X, such as $Y \to X$. All parents of X form a parent set, expressed as Pa_X, and the maximum value of $|\mathrm{Pa}_X|$ in G is called the *maximum in-degree* d_G. Regardless of whether the edges between X and Y are directed or undirected, we consider X and Y to be *adjacent*, and all nodes adjacent to X are denoted as Ad_X. If some nodes are pairwise adjacent, they form a *clique*. A *V-structure* is a subgraph formed by three nodes, where two non-adjacent nodes are parents of the third node, that is, three nodes are connected as $Y \to X \leftarrow Z$.

$X \mapsto Y$ is a *path* from X along some directed or undirected edges that end up at Y. If the edges of the path are all undirected edges, then $X \mapsto Y$ is an *undirected path*. Similarly, $X \mapsto Y$ is a *directed path* if these edges are directed from the previous node to the next node in the path. Otherwise, $X \mapsto Y$ is considered a *semi-directed path* if there is at least one directed edge and one undirected edge in the path. In a directed path $X \mapsto Y$, all nodes following X are *descendants* of X. A graph G containing a cycle means that there is a path $X \mapsto Y$ in G and X that is adjacent to Y. G is a DAG if there is no cycle in G and every edge in G is directed. Similarly, if there is at least one semi-directed path in G that does not contain any cycle, G is a partially DAG (PDAG).

Bayesian networks

A BN is a probabilistic graphical model that can be expressed as $BN = (G, \mathbb{P})$, where G is a DAG and $\mathbb{P}$ is its joint probability distribution. Let Nd_X denote all descendants of X. G encodes the conditional independence assumptions as follows:

$$\forall X \in \mathrm{V}, X \perp \mathrm{Nd}_X | \mathrm{Pa}_X. \tag{1}$$

Based on the conditional independences and chain rule of probability, the joint probability P over V can be factorized as:

$$\mathbb{P}(\mathrm{V}) = \prod_{X \in \mathrm{V}}^{|\mathrm{V}|} P(X|\mathrm{Pa}_X), \tag{2}$$

where $|\mathrm{V}|$ is the number of nodes, and Eq. (2) obviously simplifies the complexity of computing joint probability.

S&S methods allow the BNSL problem to be represented as follows:

$$G^* = \arg\max_{G \in \mathrm{G}} score(G, \mathrm{D}) \tag{3}$$

where G is all candidate structures, G^* is the optimal structure, and D is the data to be studied. In this paper, we use BIC, one of the most used scores, to estimate the learned structure. BIC has a fundamental property: it is *decomposable*, which means the BIC of a BN can be decomposed into the sum of the scores of each node with its parents, that is,

$$BIC(G, \mathrm{D}) = \sum_{X \in \mathrm{V}}^{|\mathrm{V}|} BIC(X, \mathrm{Pa}_X, \mathrm{D}), \tag{4}$$

where

$$BIC(X, \mathrm{Pa}_X, \mathrm{D}) = LL(X|\mathrm{Pa}_X) - Pen(X|\mathrm{Pa}_X). \tag{5}$$

For discrete observational data:

$$LL(X|\mathrm{Pa}_X) = \sum_{pa \in \Omega_{\mathrm{Pa}_X}} \sum_{x \in \Omega_X} N_{x,pa} \log \tilde{\theta}_{x|pa}, \tag{6}$$

$$Pen(X|\mathrm{Pa}_X) = \frac{\log |\mathrm{D}|}{2} |\Omega_{\mathrm{Pa}_X}| (|\Omega_X| - 1). \tag{7}$$

In this equation, Ω_X and Ω_{Pa_X} represent the state spaces of X and Pa_X, respectively; $N_{x,pa}$ denotes is the number of instances with $X = x \wedge \mathrm{Pa}_X = pa$ in D, so we omit D in the expansion of the equation; $\log \tilde{\theta}_{x|pa}$ is the log-likelihood estimation of the conditional probability $P(X = x|\mathrm{Pa}_X = pa)$.

Then, we introduce the relationship between the graph G and distribution $\mathbb{P}$. Let $I(G)$ and $I(\mathbb{P})$ denote the sets of conditional independence assertions entailed by G and $\mathbb{P}$, respectively. We say that G is a perfect map (p-map) of $\mathbb{P}$ if and only if $I(G) = I(\mathbb{P})$. Let $\varepsilon(G)$ denote the equivalence class (CPDAG) of a DAG G.

Equivalence class

If the independence relationships and distributions of G and G' from two DAGs are identical, they are *equivalent* to each other. The *skeleton* of a DAG is an undirected graph that transforms the directed edges of the DAG into undirected edges. Based on skeleton and v-structures, Verma and Pearl prove the following theorem:

Theorem 1 [53] *Two DAGs are equivalent iff they share the same skeleton and v-structures.*

If a PDAG G_P has the same skeleton and v-structures as a DAG G, and all directed edges in G_P also exist in G, we say that the PDAG admits a *consistent extension* and the DAG is a consistent extension of the PDAG.

All equivalent DAGs form their equivalent classes (EC), denoted by ε. Obviously, PDAG is not a precise representation of EC because we cannot ensure that every DAG in ε is a consistent extension of the PDAG corresponding to one DAG in ε. To get a suitable representation of ε shown in Definition 2, we distinguish the edges of the DAGs in ε as in Definition 1.

Definition 1 If a directed edge exists in every DAG in ε, the edge is compelled. Otherwise, the edge is reversible, that is, this edge exists in at least two of the DAGs of ε in the opposite direction.

Definition 2 A PDAG is a completed PDAG (CPDAG) if every directed edge of the PDAG is compelled and all undirected edges are reversible.

Based on Definition 2, any DAG in ε is a consistent extension of the CPDAG corresponding to ε, so we also use ε to denote the CPDAG corresponding to ε in the rest of this paper. Figure 1 shows a simple DAG, its skeleton, the CPDAG corresponding to the EC to which it belongs, and another consistent extension of the CPDAG.

With these definitions, we introduce the notion of a consistent score function. Let G^* denote a p-map of $\mathbb{P}$, and let G be an arbitrary DAG that is not a p-map. A score function is *consistent* if, in the limit as $|\mathrm{D}| \to \infty$, the following hold:

- G^* attains the highest score;

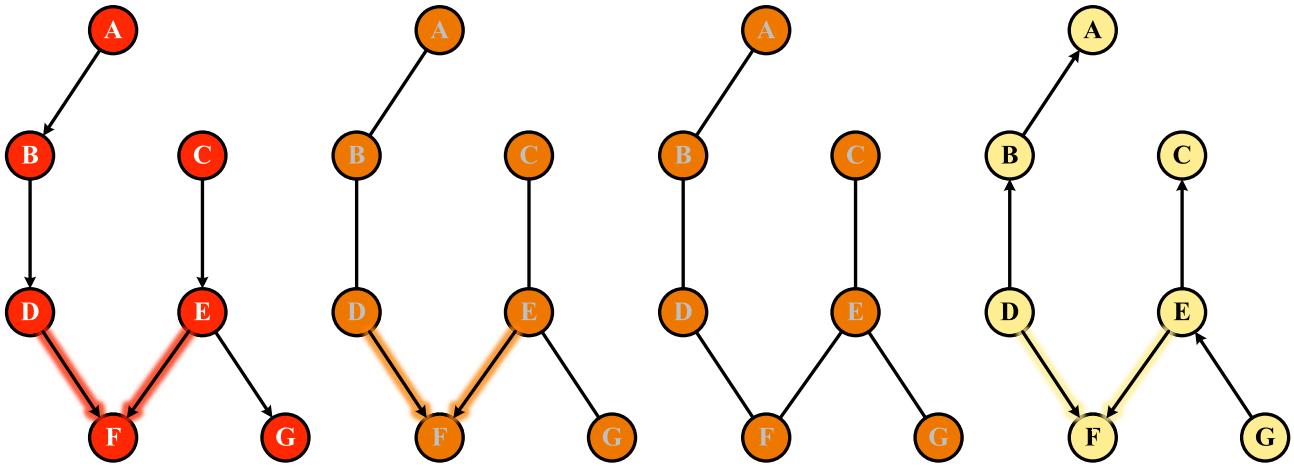

Fig. 1. Examples of (**a**) A DAG. (**b**) The CPDAG corresponds to the EC to which (**a**) belongs. (**c**) The skeleton of (**a**). (**d**) another consistent extension of (**c**).

- The score of any non-p-map DAG G is strictly less than that of any DAG in $\varepsilon(G^*)$, i.e.,

$$score(G, \mathrm{D}) < score(H, \mathrm{D}), \quad \forall H \in \varepsilon(G^*).$$

Theorem 2 [35] *BIC is a consistent score function.*

Methods

The original greedy equivalent search

GES searches for the EC with the highest score in two phases (The consistency of GES relates to the nature of the perfect map and the consistency of the score (see[54] for a detailed introduction), for detailed proof, see Sections 4.2 and 4.3 of[35].) In the first phase, called forward equivalent search (FES), iterative updates to the current MEC occur by adding one edge using a well-designed operator $Insert(X, Y, \mathrm{T})$. The second phase is called Backward Equivalent Search (BES), and it deletes one edge in one operation via the operator $Delete(X, Y, \mathrm{H})$. The operations $Insert(X, Y, \mathrm{T})$ and $Delete(X, Y, \mathrm{H})$ are shown in Definition 3 and Definition 4.

Definition 3 For any non-adjacent variables X and Y, and any subset T of the set $\mathrm{NnA}_{Y,X}$ ($\mathrm{NnA}_{Y,X} = \mathrm{Ne}_Y \bigcap \{\mathrm{V} \backslash \mathrm{Ad}_X\}$), $Insert(X, Y, \mathrm{T})$ updates the current ε in the following ways:

- Adding the edge $X \to Y$;
- Changing the undirected edges between Y and any $T \in \mathrm{T}$ into directed edges $T \to Y$.

Definition 4 For any adjacent variables X and Y, and any subset H of the set $\mathrm{NA}_{Y,X}$ ($\mathrm{NA}_{Y,X} = \mathrm{Ne}_Y \bigcap \mathrm{Ad}_X$), $Delete(X, Y, \mathrm{H})$ operates the current ε in the following ways:

- Deleting the edge $X \to Y$ or $X - Y$;
- Changing the original undirected edges $Y - H$ and $X - H$ for any $H \in \mathrm{H}$ into the directed edges $Y \to H$ and $X \to H$.

Algorithm 1 shows the outline of FES. Since the results of $Insert(X, Y, \mathrm{T})$ are not necessarily CPDAG, the function $PDAG\_To\_CPDAG$ in Line 7 transforms the PDAG into the corresponding CPDAG. It is combined by two functions, $PDAG\_To\_DAG$[55] and DAG_To_CPDAG[56]. The former extracts a consistent extension from the PDAG, and the latter transforms a DAG into the CPDAG to which it belongs. The operation of BES is similar to FES. We can obtain BES by replacing the validity tests (in Line 15), local score calculation criterion (in Line 16), and the operation (in Line 18) of $Insert(X, Y, \mathrm{T})$ in Algorithm 1 with those of $Delete(X, Y, \mathrm{H})$. The validity tests of $Insert(X, Y, \mathrm{T})$ and $Delete(X, Y, \mathrm{H})$ test how to determine whether or not an $Insert(X, Y, \mathrm{T})$ or $Delete(X, Y, \mathrm{H})$ operator is valid (more details and proof refer to Theorems 15 and 17 in[35].) The local score changes of two operators are introduced in Theorems 3 and 4.

Require: Data **D**, the initial CPDAG ε.
Ensure: A local optimal CPDAG $\varepsilon*$

```
done ← True; ε* ← ε
while done do
    bestop ← localop(ε*)
    if bestop ≠ ∅ then
        Apply bestop to ε*
    else
        done ← False
        ε* ← PDAG_To_CPDAG(ε*)
    end if
end while
procedure localop(ε*)
    bestop ← ∅; bestdelta ← 0
    for each X ∈ V do
        for each T ⊆ NnA_{Y,X} do
            if Test(X → Y, T) == True then
                Compute Δ_BIC as shown in Theorem 3.
                if Δ_BIC ≥ bestdelta then
                    bestdelta ← Δ_BIC; bestop ← Insert(X, Y, T)
                end if
            end if
        end for
    end for return bestop
end procedure
```

Algorithm 1. FES.

Theorem 3 *The changes in BIC that result from applying a valid operator* $Insert(X, Y, \mathrm{T})$ *to a CPDAG* ε *is*

$$\Delta_{BIC} = BIC(Y, \mathrm{NA}_{Y,X} \bigcup \mathrm{T} \bigcup \{\mathrm{Pa}_Y{}^{+X}\}, \mathrm{D}) - BIC(Y, \mathrm{NA}_{Y,X} \bigcup \mathrm{T} \bigcup \mathrm{Pa}_Y, \mathrm{D}). \tag{8}$$

Theorem 4 *The increments in BIC that result from applying a valid operator* $Delete(X, Y, \mathrm{H})$ *to a CPDAG* ε *is*

$$\Delta_{BIC} = BIC(Y, \{\mathrm{NA}_{Y,X} \backslash \mathrm{H}\} \bigcup \{\mathrm{Pa}_Y{}^{-X}\}, \mathrm{D}) - BIC(Y, \{\mathrm{NA}_{Y,X} \backslash \mathrm{H}\} \bigcup \mathrm{Pa}_Y, \mathrm{D}). \tag{9}$$

Based on the above, GES can be simply expressed as $GES : \varepsilon* \leftarrow BES(FES(\mathrm{D}, \varepsilon_{empty}))$, where ε_{empty} is an empty graph because the CPDAG of the EC to which the empty graph belongs is still an empty graph based on Definition 2.

Theorem 5 [35] *If the distribution* $\mathbb{P}$ *admits a p-map and* ε^* *denotes the CPDAG found by GES with a consistent score function, then in the limit of large* $|\mathrm{D}|$, ε^* is a p-map of $\mathbb{P}$.

The proposed greedy equivalent search

When GES starts from an empty graph, it is inefficient as the search space is very complex. Naturally, if GES searches for the optimal EC from an excellent seed EC with some strong dependency edges, the search space will be restricted, and the efficiency will be improved. Therefore, for the proposed method, a metric to evaluate the relationships between variables is first designed. Then, we connect the most correlated variables to form a preliminary undirected graph. After determining the v-structure, we obtain an excellent seed graph of GES.

Considering the prior probability $P(X)$ of a variable X and the posterior probability $P(X|Y)$ of that variable given another variable Y, we can use $P(X)$ and $P(X|Y)$ to measure the relationship between variables X and Y. Recall the meaning of relative entropy (RE): the difference between two probability distributions R and Q, which is formulated as follows:

$$RE(R||Q) = \sum R \log \frac{R}{Q}. \tag{10}$$

Based on RE, we design a metric, prior-posterior relative entropy (PPRE), to quantify the strength of dependencies between X and Y, as follows:

$$
\begin{aligned}
PPRE(X,Y) &= \frac{1}{2}\left(RE(P(X)||P(X|Y)) + RE(P(X||Y)||P(X))\right) \\
&= \frac{1}{2}\left(\sum_{X,Y} P(X)\log\frac{P(X)}{P(X|Y)} + \sum_{X,Y} P(X|Y)\log\frac{P(X|Y)}{P(X)}\right).
\end{aligned} \tag{11}
$$

We analyze why PPRE can determine variable dependencies in terms of both meaning and formula. On one hand, PPRE reflects the difference between $P(X)$ and $P(X|Y)$. That is, the greater the PPRE, the greater the effect of Y on X, indicating a stronger relationship between the variables Y and X. On the other hand, the posterior probability is substituted into Eq. (11) in the form of joint and marginal distribution, that is,

$$
\begin{aligned}
PPRE(X,Y) &= \frac{1}{2}\left(\sum_{X,Y} P(X)\log\frac{P(X)P(Y)}{P(X,Y)} + \sum_{X,Y}\frac{P(X,Y)}{P(Y)}\log\frac{P(X,Y)}{P(X)P(Y)}\right) \\
&= \frac{1}{2}\sum_{X,Y}\frac{1}{P(Y)}\left(P(X)P(Y)\log\frac{P(X)P(Y)}{P(X,Y)} + P(X,Y)\log\frac{P(X,Y)}{P(X)P(Y)}\right) \\
&= \frac{1}{2}\sum_{X,Y}\frac{1}{P(Y)}\left((P(X)P(Y)-P(X,Y))\log\left(P(X)P(Y)-P(X,Y)\right)\right).
\end{aligned} \tag{12}
$$

Equation (12) represents that $PPRE(X, Y)$ is mainly affected by the product of the marginal distributions $P(X)P(Y)$ and their joint distribution $P(X, Y)$, which indicates the extent to which the uncertainty about X is reduced given that Y is known. To clarify, the proposed PPRE metric is defined as RE between the prior distribution $P(X)$ and the posterior distribution $P(X|Y)$. This direct comparison of prior and posterior distributions naturally captures the effect of Y on X, reflecting the conditional dependency between variables. Unlike general-purpose divergence measures that compare arbitrary distributions, PPRE is specifically designed to quantify variable dependencies in Bayesian network structure learning. PPRE is derived from RE, so it has three properties similar to RE.

Property 1 *If X and Y are independent, then* $PPRE(X,Y) = 0$.

Proof If X and Y are independent, then $P(X|Y) = P(X)$. Therefore, $PPRE(X,Y) = 0$ based on Eq. (11). □

Let Id_X denote the set of variables independent of X, and then we get Property 2.

Property 2 *For* $\forall Y \in \mathrm{Id}_X$, *iff the distributions of X and Y are identical,* $PPRE(X,Y) = PPRE(Y,X)$.

Proof According to Eq. (12),

$$
PPRE(X,Y) - PPRE(Y,X) = \frac{1}{2}\sum_{X,Y}\left(\frac{1}{P(Y)} - \frac{1}{P(X)}\right)\cdot Q(X,Y), \tag{13}
$$

where $Q(X,Y) = (P(X)P(Y) - P(X,Y))\log(P(X)P(Y) - P(X,Y))$. Since Y is independent of X, $Q(X, Y)$ must be nonzero. Therefore, iff the distributions of X and Y are identical, Eq. (13) is zero, and the property is proven. □

Property 3 $PPRE(X,Y) \geq 0$

Proof The property naturally follows from the nonnegativity of the RE. □

PPRE compares $P(X)$ and $P(X \mid Y)$ via a symmetrized RE for a fixed target X. Hence $PPRE(X,Y) \geq 0$ and $PPRE(X,Y) = 0$ if $P(X \mid Y) = P(X)$ (e.g., when $X \perp Y$). Note that $PPRE(X, Y)$ and $PPRE(Y, X)$ are generally different, since they quantify how Y affects X and how X affects Y, respectively. This directional asymmetry is used in the proposed algorithm through the strongest dependent sets to determine edge orientation.

For the evaluation criteria for dependencies, we designed a PPRE-based GES algorithm, called PPGES, the pseudo-code of which is given in Algorithm 2. The PPGES will be introduced in detail in stages. Before describing the pseudo-code, we clarify several notations used in Algorithm 2. $\varepsilon*$ denotes the current partially directed acyclic graph (PDAG) constructed during the algorithm. It is initialized as the empty graph on the variable set V, and will be updated iteratively until it becomes a local optimal CPDAG. SD_X denotes the strongest dependent set of variable X, which is a subset of candidate parent variables for X. It is obtained by ranking all variables $Y \neq X$ according to $PPRE(X, Y)$. *cache* is a temporary data structure storing the pairwise $PPRE(X, Y)$ values between variable X and the remaining variables. For each X, *cache* is sorted in descending order of PPRE values to facilitate the selection of SD_X.

Require: Data $\mathbf{D}$, the maximum in-degree d_G, threshold t
Ensure: A local optimal CPDAG $\varepsilon*$
1: $\mathbf{V} \leftarrow$ get the variables of $\mathbf{D}$; $\varepsilon* \leftarrow emptygraph(\mathbf{V})$
2: **for** each $X \in \mathbf{V}$ **do**
3: $\quad \mathbf{SD}_X \leftarrow \varnothing$
4: $\quad$ **for** each $Y \in \mathbf{V} \backslash \{X\}$ **do**
5: $\quad\quad cache \leftarrow$ compute $PPRE(X, Y)$ in $\mathbf{D}$
6: $\quad$ **end for**
7: $\quad cache \leftarrow sort(cache)$
8: $\quad \mathbf{SD}_X \leftarrow select(cache, d_G)$
9: $\quad \mathbf{SD}_X \leftarrow thresh(\mathbf{SD}_X, t)$
10: **end for**
11: **for** each $X \in \mathbf{V}$ **do**
12: $\quad$ **if** $\mathbf{SD}_X ! = \varnothing$ **then**
13: $\quad\quad$ **for** each $Y \in \mathbf{SD}_X$ **do**
14: $\quad\quad\quad$ **if** $X \in \mathbf{SD}_Y$ **then**
15: $\quad\quad\quad\quad \varepsilon* \leftarrow AddEdge(\varepsilon*, X - Y, check.cycles = T)$
16: $\quad\quad\quad\quad \mathbf{SD}_Y = \mathbf{SD}_Y \backslash \{X\}$
17: $\quad\quad\quad$ **else**
18: $\quad\quad\quad\quad \varepsilon* \leftarrow AddEdge(\varepsilon*, X \leftarrow Y, check.cycles = T))$
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21: $\quad$ **end if**
22: **end for**
23: $\varepsilon* \leftarrow$ PDAG_to_CPDAG($\varepsilon*$)
24: $\varepsilon* \leftarrow GES(\mathbf{D}, \varepsilon = \varepsilon*)$

Algorithm 2. PPGES.

Obviously, PPGES is a three-phase algorithm. The inputs of PPGES include the observation data D, maximum in-degree of G d_G, and the threshold t. In the first phase (Lines 2-10), we compute the PPRE to determine the strongest dependent sets (expressed as SD_X) for each variable. Since subsequent operations may query the PPRE multiple times, the PPRE of each variable and the remaining variables is calculated and stored in the cache for efficiency, which has a complexity of $O(|\mathrm{D}| \cdot (|\mathrm{V}| - 1))$. In order to facilitate subsequent operation, PPGES sorts $PPRE(X, \mathbb{V}\backslash X), X \in \mathrm{V}$ in the cache, the worst complexity of which is $O((|\mathrm{V}| - 1) \cdot \log(|\mathrm{V}| - 1))$. After sorting, we select the first d_G variables from the cache as the SD_X of the variable X. Finally, the variables with a PPRE greater than the threshold t were filtered out as the final SD_X. As for the threshold t, we will obtain it statistically based on the PPRE of the standard network, and the determination of it will be discussed in "Results" section. The complexity of sorting and filtering is $O(2d_G)$. Therefore, for each variable $X \in \mathrm{V}$, the worst-case complexity is $O(|\mathrm{D}| \cdot |\mathrm{V}| + |\mathrm{V}| \cdot \log(|\mathrm{V}| - 1) - \log(|\mathrm{V}| - 1))$. Naturally, the worst complexity of the first phase is $O\left(|\mathrm{V}|^2\right)$.

In the second phase (Lines 11-23), PPGES obtains a CPDAG with the strongest dependencies. Based on the set SD_X, we conduct connect rules for the variables:

- Rule 1. If $X \in \mathrm{SD}_Y \wedge Y \notin \mathrm{SD}_X$, the algorithm applies $AddEdge(X \rightarrow Y)$;
- Rule 2. If $X \notin \mathrm{SD}_Y \wedge Y \in \mathrm{SD}_X$, the algorithm applies $AddEdge(X \leftarrow Y)$;
- Rule 3. If $X \in \mathrm{SD}_Y \wedge Y \in \mathrm{SD}_X$, the algorithm applies $AddEdge(X - Y)$.

These connection rules are based on the conditional dependence information captured by PPRE. For Rules 1 and 2, if X strongly depends on Y but not vice versa, we direct the edge as $X \leftarrow Y$, reflecting that knowing Y reduces the uncertainty of X more than the reverse. Rule 3 handles the case where both variables are strongly dependent on each other; since the direction cannot be reliably determined from PPRE alone, we add an undirected edge $X - Y$, consistent with the PDAG representation in Bayesian networks. Overall, these rules ensure the resulting PDAG respects acyclicity and can serve as a sound seed graph for the subsequent GES optimization. While heuristic in implementation, they are theoretically motivated by conditional dependence and BN structure constraints. In order to prevent the repeated addition of edges, X is removed from SD_Y after connecting them. Since the operation of the first phase, for $\forall X \in \mathrm{V}, |\mathrm{SD}_X| \leq d_G$, the worst case of the above operations' complexity is $O\left(d_G^2 \cdot |\mathrm{V}|\right)$. The function $AddEdge(\cdot)$ in Lines 15 and 18 denotes the operation of adding an edge into the current graph. In our implementation, it is realized by functions such as *set.edge()* and *set.arc()* in the *bnlearn* package. After connecting their strongest dependent variables for each variable, the resulting graph is a PDAG. However, the required input for GES is CPDAG. Thus, the final step of the second

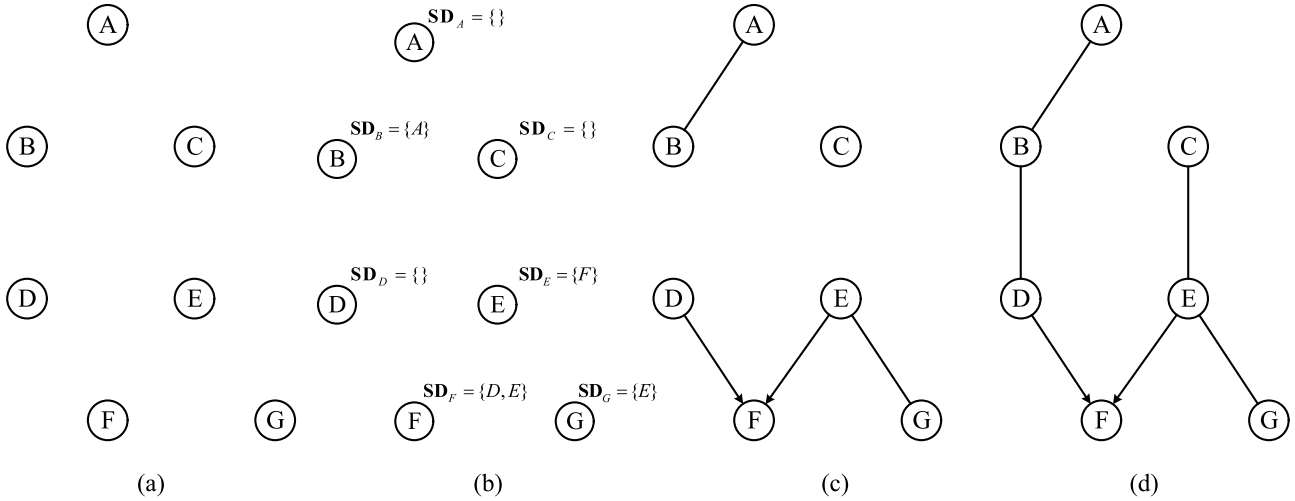

Fig. 2. Examples of PPGES. (**a**) An empty graph. (**b**) The resulting graph after the first phase of PPGES. (**c**) The resulting graph after the second phase of PPGES. (**d**) The resulting graph after the third phase of PPGES, that is, the final CPDAG.

Networks	Nodes	Edges	Parameters	Maximum indegree
Asia	8	8	18	2
Sachs	11	17	178	3
Alarm	37	46	509	4
Child	20	25	230	2
Insurance	27	52	1008	3
Water	32	66	10,083	5
Hailfinder	56	66	2656	4
Hepar2	70	123	1453	6
Win95pts	76	112	574	7

Table 1. The characteristics of benchmarks.

phase transforms the resulting graph into a CPDAG using $PDAG\_to\_CPDAG$, whose worst complexity is $O\left(d_G^2 \cdot |\mathrm{V}| + d_G^2 \cdot |\mathrm{E}| + d_G^2\right)$, where |E| represents the number of edges in G. Therefore, the upper bound of the second phase's complexity is $O\left(|\mathrm{V}|\right)$.

The final phase (Line 24) of PPGES uses the CPDAG obtained in the second phase as input to optimize the final CPDAG further using GES. Figure 2 shows an example of PPGES.

Results

In this section, we first introduce the backgrounds of method comparisons in "Backgrounds of comparisons" section, such as benchmarks, comparison algorithms, and hardware and software conditions. Then, "Comparisons on GES and PPGES" section compares the performance differences between GES and PPGES with different parameters. "Comparisons on PPGES and other algorithms" section shows the accuracy of the results of PPGES and other competitive algorithms. Finally, "Comparisons on real data" section applies the proposed algorithm and the state-of-the-art solver to a real dataset.

Backgrounds of comparisons

All benchmarks in "Comparisons on GES and PPGES" and "Comparisons on PPGES and other algorithms" sections are sampled from the ground-truth models, which can be downloaded from the *Bayesian Network Repository* (https://www.bnlearn.com/bnrepository/). We select two small networks (*Asia*, *Sachs*), four medium networks (*Alarm*, *Child*, *Insurance*, *Water*), and three large networks (*Hailfinder*, *Hepar2*, *Win95pts*), as well as the classification criteria are also from the Bayesian Network Repository. The characteristics of these real networks are given in Table 1.

All the state-of-the-art or well-established algorithms used for the comparison are briefly described as follows:

- PC.stable[13]: The competitive CB baseline algorithm;
- GOBNILP[20]: The state-of-the-art exact search solver;
- GES[35]: The most representative E-space local search algorithm;

- ILGES[41]: The state-of-the-art E-space local search algorithm;
- ARGES[42]: The hybrid algorithm in E-space;
- MMHC: The hybrid baseline algorithm in D-space;
- MMPC+GES: The initial graph for GES is provided by MMPC;
- MAHC[26]: A D-space local search algorithm using model averaging;
- DAG-GNN[49]: The competitive continuous optimization baseline algorithm.
- GOLEM[51]: A continuous optimization baseline algorithm.
- DAGMA[52]: The competitive continuous optimization baseline algorithm.

PC.stable and MMHC are programmed using the R package bnlearn[57] (https://www.bnlearn.com/). GOBNILP runs in C based on the source code (https://www.cs.york.ac.uk/aig/sw/gobnilp/). GES, ILGES, and the proposed PPGES are implemented in R based on bnlearn and pcalg. ARGES runs based on the R package pcalg[58]. MAHC runs in Java using the source code (https://bayesian-ai.eecs.qmul.ac.uk/bayesys/). ESTOBS runs in C++, leveraging the source code from its authors. DAG-GNN (https://github.comfi/shmoon1234/DAG-GNN), and DAGMA (https://github.com/kevinsbello/dagma) run in Python based on their source code. GOLEM is provided by the Python toolbox gCastle[59]. It is noted that GOBNILP plays a role in searching for the global optimum in the comparisons, which provides a reference for other algorithms. MMPC+GES combines the MMPC phase of MMHC with GES according to the operations of the authors of ARGES. The parameters of all the above algorithms use the default values from their papers. All comparisons are made on a computer with an Intel Core i7-10700f (2.9GHz) processor and 32GB of RAM.

In "Comparisons on GES and PPGES" section, the criteria of comparisons are BIC and run time. In "Comparisons on PPGES and other algorithms" section, the criteria are BIC and structural hamming distances (SHD). SHD is defined by the authors of MMHC to compare the differences between the learned CPDAG $\varepsilon_{learned}$ and the true CPDAG of BN ε_{true}, which can be computed as follows:

$$SHD(\varepsilon_{learned}, \varepsilon_{true}) = \#\text{missing edges} + \#\text{extra edges} + \#\text{reversal edges}. \tag{14}$$

That is, SHD is the sum of the number of missing edges, extra edges, and edges with the opposite direction in the learned CPDAG compared to the true CPDAG.

Sensitivity analysis of PPGES parameters: comparisons with GES

To investigate how the parameters of PPGES affect its performance, we conduct a sensitivity analysis by comparing PPGES with GES under different parameter settings. Specifically, for the maximum in-degree, we consider three alternatives $d_G = \{1, 3, 6\}$. For the threshold parameter, we select the mean, median, and upper-hinge (the latter two are values in Tukey's five-number summary[60]) of PPRE, which corresponds to the strongest variable dependence among all ground-truth models. Note that these threshold values are computed globally across all variables rather than individually. This yields nine combinations of parameters for PPGES, which are evaluated in this section.

We sample 10,000 instances for each benchmark network as this section's data sets. On the one hand, the differences between GES and PPGES run time, $\Delta_{time} = T_{PPGES} - T_{GES}$, are used to show that PPGES improves the efficiency of GES with different parameters. On the other hand, $\Delta_{BIC} = BIC_{PPGES} - BIC_{GES}$ shows the effect of PPGES on the scores of GES under different parameters. The results are given in Figs. 3 and 4. The combination of parameters is represented as "in-degree=X,threshold=X". For example, "d=3,t=upper" indicates the result when the maximum degree is set to 3, and the threshold is set to upper-hinge. The lighter bars in Figs. 3 and 4 indicate that PPGES improves the performance of GES, while the darker bars imply that PPGES does not perform as well as GES.

In each graph of Fig. 3, the results of PPGES with different parameters are lighter bars, which means that PPGES with different parameters achieve better efficiency. To the contrary, some combinations of parameters obtain worse scores than GES. In summary, the parameters we choose for PPGES are $d_G = 3$ and $t = upper\text{ - }hinge$. PPGES with the combination outperformed GES in 7/9 networks, and of those seven networks, four were the most accurate results. Meanwhile, PPGES with the combination performs well in terms of efficiency as shown in Fig. 3.

With the parameters, we compared the ratio in which PPGES improved efficiency with different amounts of data. Meanwhile, in this subsection, we compare the efficiency of MMPC+GES and PPGES, and their accuracy will be compared in "Comparisons on PPGES and other algorithms" section. In Fig. 5, the x-axis represents the amount of data, and the y-axis represents $\frac{T_{GES} - T_{algorithm}}{T_{GES}}$.

Obviously, both MMPC+GES and PPGES can significantly improve the efficiency of GES. As a well-established CB method, MMPC is more efficient than the proposed algorithm. MMPC+GES runs faster than PPGES in 29/36 cases, and the differences between their results are less than 0.25 in 16/29 cases.

Comparisons on PPGES and other algorithms

In this section, we sample 1000, 5000, 10,000, and 100,000 instances for each ground-truth model. The BIC scores obtained by all the comparison algorithms are listed separately in Tables 2, 3, 4, and 5. The results of SHD are shown in Tables 6, 7, 8, and 9. The best score for each of the benchmarks in the table is written in bold, and two decimal places are reserved for BIC. If the algorithm goes wrong for some reason, the results will be represented as "–".

In Table 2, although GOBNILP finds a global optimum on Asia, Sachs, and Child, it fails when the number of nodes increases. MAHC, DAG-GNN, GOLEM and DAGMA can stably output results, but their scores are far from those of other algorithms, especially in the Hailfinder model. PC.stable wins 1 of 9 models, but its

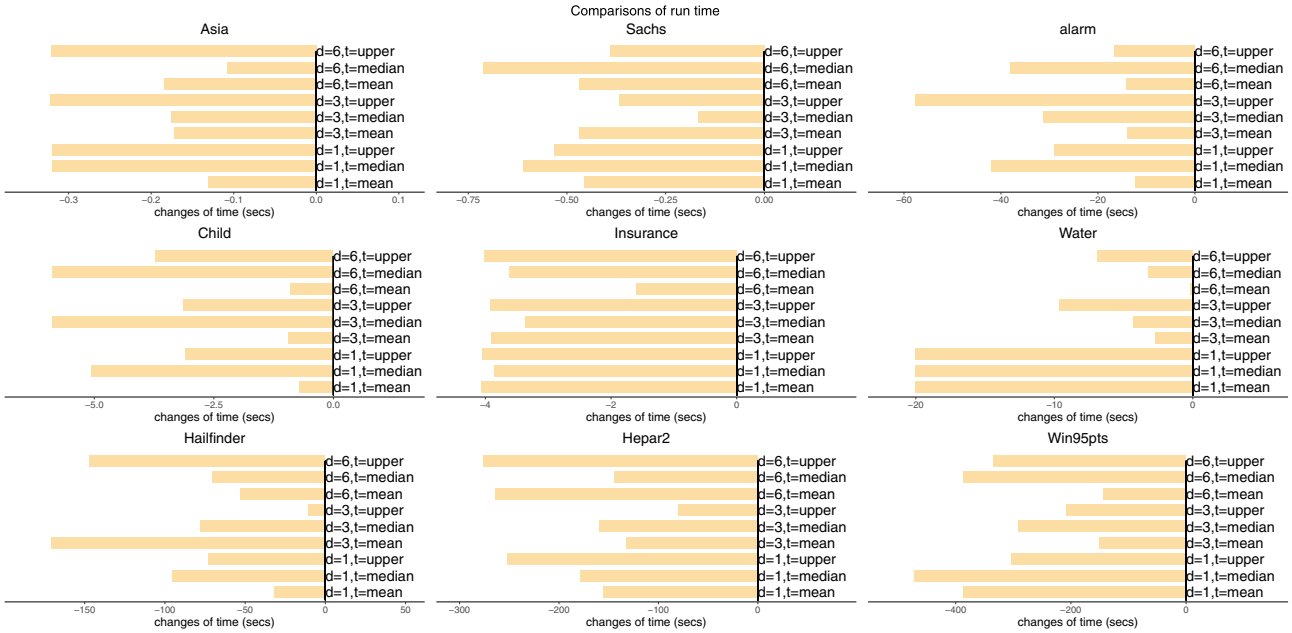

Fig. 3. Comparisons of PPGES and GES in terms of running time.

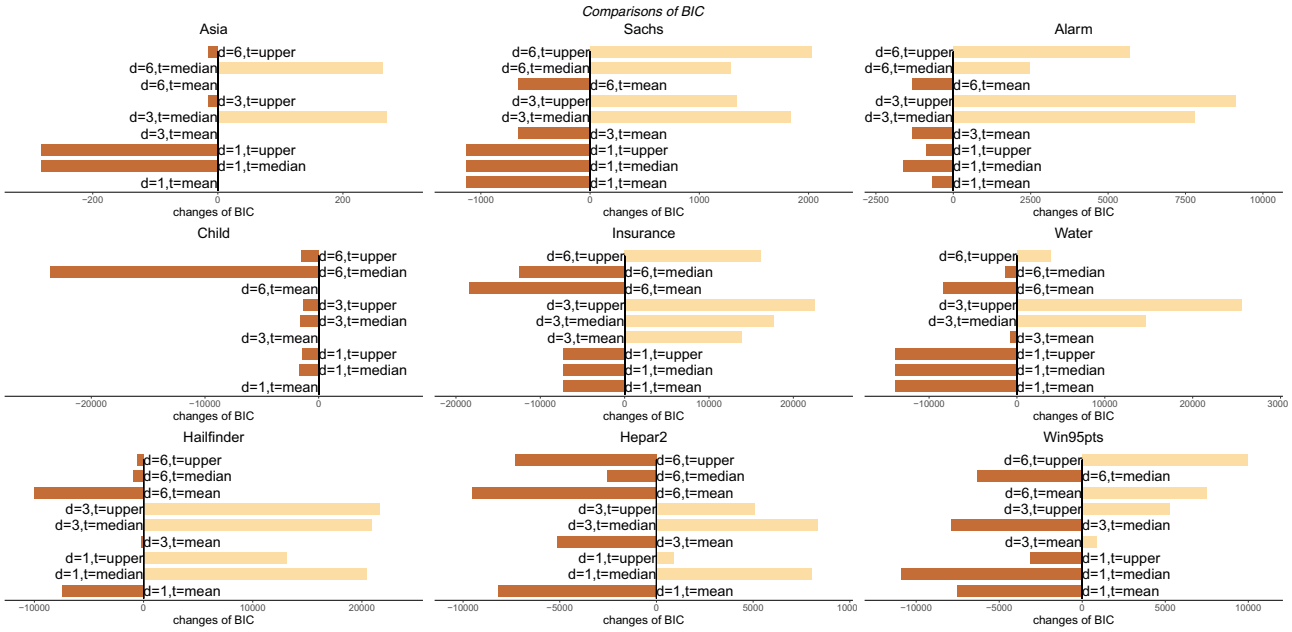

Fig. 4. Comparisons of PPGES and GES in terms of BIC.

performance is limited by the conditional independence tests. PPGES and ILGES obtain higher scores than GES in most models. In detail, ILGES outputs the Hepar2 model with the highest score, whereas PPGES outperforms other competitive solvers in 6/9 models. Moreover, although PPGES is not the best method for the Sachs and Hepar2 models, it also gets excellent scores that are very close to the highest score.

The results of 5000 instances are similar to those of 1000 instances, as shown in Table 3. ARGES successfully outputs a CPDAG in the Insurance model, but it fails in Table 2. It also maintains excellent performance in the Asia model but still obtains a poor result on the Alarm model. Obviously, PPGES still outperforms other algorithms on 6/9 models. Although PPGES is not the best algorithm on models Asia, Sachs, and Child models, its score is only 1.37%, 2.53%, and 0.43% away from the optimal score, respectively.

As the amount of data increases, algorithms perform better in all models, such as DAG-GNN, which scored only 0.47% worse than the highest score in the Water model. Obviously, PPGES stably outputs better CPDAG for 7/9 models with 10,000 instances. In these seven networks, PPGES improves from 0.30% to 2.11% over the

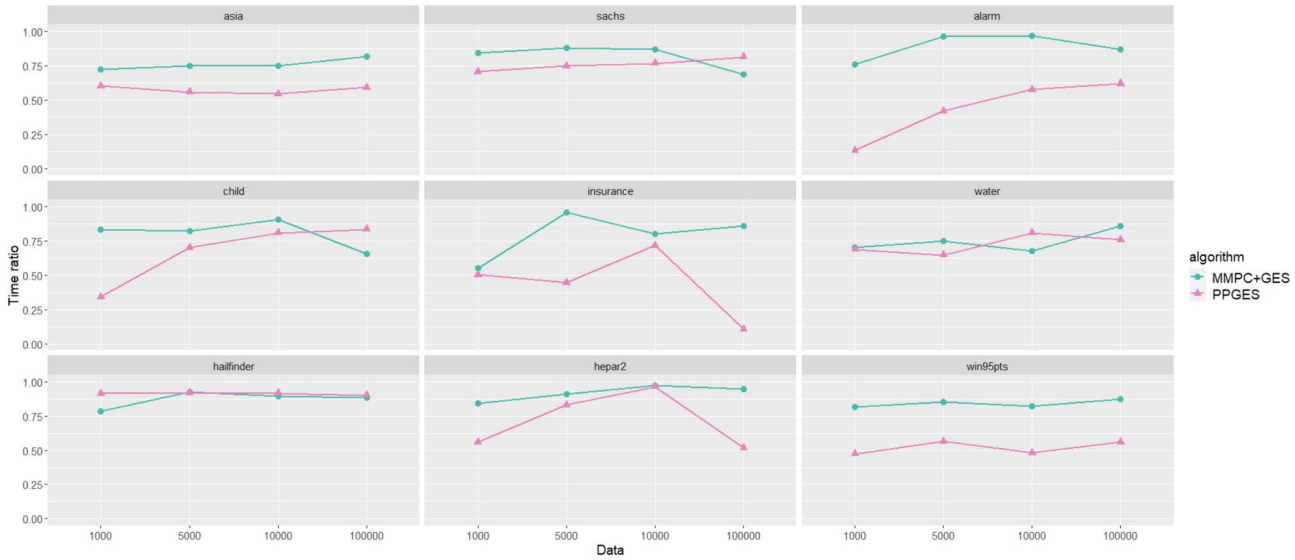

Fig. 5. The ratio of which PPGES and MMPC+GES improved efficiency with different amounts of data.

suboptimal algorithm. For the Asia and Child models, PPGES is only 1.05% and 0.11% less than the global optimal, respectively.

In the comparison results of 100,000 instances, seven comparative algorithms search for the global optimal in the Sachs model, as shown in Table 5. PPGES performs better than the results in Tables 2, 3, and 4. In the results of the Child model, the proposed method outputs the same score as GOBNILP. Although MMPC+GES obtains the same result as GES for the Asia and Sachs models, it performs worse than GES for 6/9 models.

As for SHD, PPGES does not perform as well as in comparisons of BIC, especially in Table 6. That is because GES is an S&S algorithm, and the search for the best results is based on the score function. Similarly, the structures with global optimal scores found by GONBNILP do not necessarily have the lowest SHD. As the amount of data increased, the trend of BIC and SHD gradually matched, and PPGES performed better and better. In the results of Table 9, PPGES found the CPDAG on 6/9 models that were closest to the real structure. In contrast to the BIC comparison results, MAHC performed better in the SHD comparison, possibly because the method used the strategy of model averaging. Similar to BIC, MMPC+GES is still not better than GES in terms of any amount of data.

As for the comparisons between PPGES and MMPC+GES, the proposed algorithm can improve GES in terms of both efficiency and accuracy, while MMPC+GES only enhances efficiency and even worsens accuracy in many cases.

It is also worth noting that a better BIC score does not necessarily imply a lower SHD, because BIC and SHD evaluate different aspects of model quality. As a score-equivalent and consistent scoring function, BIC is guaranteed to asymptotically identify the p-map of the underlying distribution. In contrast, SHD is a purely combinatorial metric that counts edge-level differences between graphs, which makes SHD more sensitive to local structural variations, sample fluctuations, and characteristic of benchmark networks. Consequently, it is possible for BIC to show clear improvement while SHD appears unstable in certain cases. Nevertheless, in line with the asymptotic property of BIC (see Theorem 2), our experiments show that the SHD of PPGES improves with increasing sample size across most benchmark networks (see Tables 6–9).

Comparisons on real data

Since it is much more difficult to obtain the true causal structure from real datasets, in this section we only select PPGES and MAHC, which performed best in SHD in the previous comparisons, and apply them to a real dataset. This dataset is derived from a research[61] that recorded the development of the 2019 coronavirus disease (COVID-19) in the UK over 866 days from January 2020 to June 2022, forming raw data with 866 rows and 18 columns. The raw data, processed data, processing methods, ground-truth graph, and data set descriptions are available from the literature[61].

The scoring performances of PPGES and MAHC are listed in Table 10. Undoubtedly, PPGES has more advantages compared with MAHC on this real dataset, whether in terms of structure score or graphical evaluation. Then, to observe the causal structures obtained by the two methods more intuitively, the subgraphs of the causal structures they output are shown in Fig. 6. In Fig. 6a and b, the correct edges, the extra edges (the edges that do not exist in the ground-truth graph but are output by the algorithm), and the incorrect edges (the missing edges compared to the ground-truth graph and the edges in the opposite direction) are marked in black, blue dash lines, and red, respectively.

In detail, MAHC recovers five correct edges, all of which are also correctly discovered by PPGES. PPGES correctly determine the causal relationships, $New\_infections \rightarrow Positive\_tests$ and $Hospital\_admissions \rightarrow Patients\_in\_hospital$, while MAHC fails in these causality. Moreover,

	PC.stable	GOBNILP	MAHC	MMHC	DAG-GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	− 2532.46	**− 2281.80**	− 3074.99	− 2484.06	− 2367.16	− 2730.15	− 2425.36	− 2325.87	− 2325.87	− 2325.46	**− 2281.80**	**− 2281.80**
Sachs	− 8121.20	**− 7611.09**	− 9435.24	− 8027.27	− 8441.59	− 8380.75	− 8093.59	− 7801.89	− 8035.76	− 7676.35	− 8763.48	− 7629.53
Alarm	− 15,010.74	OM	− 20,550.75	− 13,774.95	− 13,608.17	− 15,875.65	− 12,963.79	− 12,320.59	− 13,152.11	− 11,864.48	− 25,624.25	**− 11670.45**
Child	− 14,007.36	**− 12795.86**	− 17,290.74	− 13,122.31	− 14,949.39	− 15,488.73	− 15,188.81	− 12,938.10	− 12,985.27	− 12,929.48	− 13,948.77	− 12,975.54
Insurance	− 16,755.22	OM	− 21,719.10	− 15,900.91	− 17,366.45	− 17,813.25	− 17,098.79	− 16,983.49	− 17,010.86	− 15,468.04	–	**− 14,943.68**
Water	**− 13,609.72**	OM	− 16,742.02	− 13,817.19	− 13,828.35	− 15,008.54	− 13,735.08	− 14,927.37	− 13,817.19	− 14,898.83	–	**− 13,609.72**
Hailfinder	− 59,873.89	OM	− 70,149.00	− 59,080.54	− 63,733.30	− 82,856.05	− 85,712.61	− 55,577.39	− 72,556.22	− 54,013.54	–	**− 53,590.85**
Hepar2	− 34,285.39	OM	− 35,680.19	− 33,833.07	− 34,276.78	− 34,850.87	− 34,235.59	− 33,720.02	− 33,798.10	**− 33,665.94**	− 34,748.84	− 33,697.51
Win95pts	− 12,774.83	OM	− 18,993.26	− 12,637.99	− 15,363.27	− 16,866.02	− 12,100.05	− 11,356.70	− 12,893.15	− 10,826.87	–	**− 10,681.89**

Table 2. The BIC of all algorithms for benchmarks with 1,000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG– GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	− 12,290.01	**− 11,301.47**	− 15,061.48	− 12,290.01	− 11,592.77	− 13,447.32	− 12,050.35	− 11,595.09	− 11,595.09	− 11,346.97	**− 11,301.47**	− 11,456.49
Sachs	− 37,913.35	**− 36,915.45**	− 47,076.54	− 38,493.01	− 42,846.81	− 40,242.40	− 40,580.79	− 38,806.01	− 37,630.90	− 36,770.15	− 41,070.44	− 37,849.80
Alarm	− 64,546.49	OM	− 102,853.39	− 61,433.89	− 61,324.48	− 64,692.92	− 61,436.11	− 60,028.49	− 73,073.82	− 57,417.79	− 100,464.62	**− 55,429.86**
Child	− 63,152.05	**− 61,643.80**	− 85,966.79	− 67,055.41	− 72,613.70	− 73,235.51	− 70,793.75	− 63,114.55	− 68,146.78	**− 61,643.80**	− 79,281.23	− 61,908.81
Insurance	− 75,555.82	OM	− 107,384.78	− 72,428.32	− 81,769.58	− 85,532.74	− 78,655.08	− 98,031.99	− 75,876.54	− 68,717.33	− 118,492.06	**− 68,135.83**
Water	**− 65,688.01**	OM	− 84,125.85	− 65,975.50	− 66,139.82	− 68,369.43	− 65,780.46	− 67,998.46	− 67,991.75	− 67,998.46	–	**− 65,688.01**
Hailfinder	− 297,999.21	OM	− 349,045.45	− 288,225.41	− 308,867.69	− 302,338.05	− 332,997.45	− 257,995.58	− 293,652.88	− 267,001.54	–	**− 256,363.51**
Hepar2	− 166,184.98	OM	− 175,381.05	− 164,997.70	− 169,251.69	− 169,993.48	− 167,889.79	− 165,057.20	− 166,074.96	− 164,100.11	− 164,584.45	**− 163,994.94**
Win95pts	− 58,543.17	OM	− 94,946.49	− 58,083.76	− 94,946.49	− 60,091.38	− 60,870.18	− 55,901.11	− 63,540.88	− 49,139.12	–	**− 46,956.29**

Table 3. The BIC of all algorithms for benchmarks with 5,000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG– GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	– 24,365.10	**– 22,564.14**	– 29,849.16	– 24,365.10	– 23,327.26	– 26,886.01	– 24,104.10	– 23,092.89	– 23,092.89	– 22,610.10	– 22,584.04	– 22,801.28
Sachs	– 75,661.14	**– 72,617.47**	– 94,068.43	– 74,179.78	– 75,962.53	– 80,187.54	– 79,702.32	– 74,867.42	– 74,179.78	– 74,179.78	– 96,660.01	**– 72,617.47**
Alarm	– 120,651.65	OM	– 205,247.98	– 128,418.18	– 123,853.23	– 129,598.19	– 120,676.47	– 108,490.86	– 137,269.17	– 107,445.37	– 159,613.20	**– 106,489.99**
Child	– 125,920.79	**– 123,342.01**	– 172,523.00	– 133,567.85	– 138,731.21	– 147,921.25	– 139,623.59	– 126,024.10	– 126,926.03	**– 123,342.01**	– 138,721.79	– 123,478.51
Insurance	– 143,212.46	OM	– 215,256.10	– 145,724.26	– 158,923.36	– 167,386.57	– 158,351.54	– 143,880.21	– 159,207.82	– 135,414.34	– 1,858,248.05	**– 134,129.76**
Water	– 130,857.23	OM	– 167,556.46	– 130,424.53	– 130,574.10	– 146,969.53	– 131,600.15	– 133,006.82	– 134,669.63	– 133,006.82	–	**– 129,968.48**
Hailfinder	– 587,001.06	OM	– 698,042.51	– 574,581.95	– 589,060.30	– 595,826.44	– 609,970.52	– 512,694.70	– 575,238.88	– 512,694.70	–	**– 507,574.18**
Hepar2	– 333,245.54	OM	– 350,817.96	– 328,353.31	– 338,222.74	– 334,345.96	– 335,751.14	– 328,182.46	– 332,811.49	327,283.25	– 327,735.74	**– 326,307.94**
Win95pts	– 118,603.18	OM	– 190,608.94	– 114,826.97	– 150,466.72	– 126,111.18	– 121,209.18	– 101,139.52	– 120,735.37	– 94,100.65	–	**– 92,527.22**

Table 4. The BIC of all algorithms for benchmarks with 10,000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG– GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	– 242,188.08	**– 223,622.57**	– 297,992.98	– 242,188.08	– 235,852.76	– 269,549.69	– 238,810.19	– 229,126.71	– 229,126.71	**– 223,647.73**	– 223,667.71	– 226,283.37
Sachs	**– 717,467.15**	**– 717,467.15**	– 934,705.16	**– 717,467.15**	– 786,939.79	– 787,794.35	– 781,553.94	**– 717,467.15**	**– 717,467.15**	**– 717,467.15**	– 745,281.66	**– 717,467.15**
Alarm	– 1,089,011.99	OM	– 2,044,186.38	– 1,324,936.86	– 1,194,160.24	– 1,194,080.71	– 1,191,830.27	**– 1,173,838.29**	– 1,389,831.80	– 1,045,398.13	– 2,640,593.08	**– 1,045,331.11**
Child	– 1,220,252.88	**– 1,218,221.22**	– 1,714,975.10	– 1,403,144.57	– 1,295,636.66	– 1,461,888.71	– 1,383,929.29	– 1,250,160.08	– 1,338,746.26	**– 1,218,221.23**	– 2,047,175.93	**– 1,218,221.23**
Insurance	**– 1,444,604.14**	OM	– 2,151,837.02	– 1,462,445.06	– 1,612,730.55	– 1,712,079.97	– 1,559,338.38	– 1,345,038.68	– 1,551,219.47	**– 1,319,151.89**	– 2,050,428.21	**– 1,315,583.42**
Water	– 1,315,030.36	OM	– 1,673,682.07	– 1,294,214.23	– 1,299,843.54	– 1,326,551.32	– 1,312,381.10	– 1,314,488.96	– 1,328,414.28	**– 1,289,470.62**	–	**– 1,283,411.59**
Hailfinder	– 5,948,587.12	OM	– 6,970,752.18	– 5,694,926.33	– 5,752,200.47	– 5,642,508.70	– 5,460,847.31	– 5,872,805.28	– 5,448,337.00	**– 5,090,207.54**	–	**– 4,946,901.71**
Hepar2	– 3,284,440.69	OM	– 3,509,023.49	– 3,272,838.19	– 3,332,210.20	– 3,354,841.81	– 3,353,806.03	– 3,278,225.29	– 3,335,440.60	**– 3,261,227.35**	– 3,259,979.69	**– 3,254,129.17**
Win95pts	– 1,111,588.81	OM	– 1,913,481.54	– 1,097,865.73	– 1,154,007.27	– 1,075,796.31	– 1,217,339.51	– 1,108,643.52	– 1,189,316.59	– 911,140.34	–	**– 906,902.08**

Table 5. The BIC of all algorithms for benchmarks with 100,000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG-GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	6	**1**	**1**	4	11	11	8	6	6	6	**1**	**1**
Sachs	8	1	7	8	16	18	11	10	5	**0**	22	8
Alarm	32	OM	26	28	41	68	34	46	45	**16**	53	27
Child	13	12	16	18	23	45	23	16	17	10	26	**8**
Insurance	44	OM	45	**43**	59	62	51	52	49	46	–	44
Water	60	OM	**58**	60	66	99	60	60	60	**58**	–	60
Hailfinder	43	OM	42	49	70	120	88	58	63	53	–	**34**
Hepar2	125	OM	116	**114**	122	172	120	119	135	114	115	**114**
Win95pts	91	OM	76	92	112	184	103	90	110	110	–	**64**

Table 6. The SHD of all algorithms for benchmarks with 1000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG-GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	4	**1**	3	4	8	12	8	6	6	3	**1**	4
Sachs	8	**0**	2	3	16	21	12	10	2	1	24	8
Alarm	22	OM	23	22	41	82	35	46	43	23	56	**16**
Child	9	**0**	8	16	22	48	24	15	19	**0**	39	6
Insurance	40	OM	40	41	55	84	53	45	45	44	75	**39**
Water	63	OM	54	58	65	93	60	53	61	**47**	–	51
Hailfinder	48	OM	**32**	44	70	119	93	57	66	39	–	47
Hepar2	109	OM	73	93	123	176	119	102	122	72	65	**60**
Win95pts	61	OM	71	68	112	162	104	83	108	90	–	**38**

Table 7. The SHD of all algorithms for benchmarks with 5000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG-GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	4	**0**	3	4	8	10	8	6	6	3	1	4
Sachs	14	**0**	2	2	15	16	11	10	2	2	29	**0**
Alarm	14	OM	23	23	44	73	35	30	42	**7**	63	8
Child	7	**0**	7	14	22	44	24	15	16	**0**	40	4
Insurance	28	OM	38	41	52	68	51	36	47	43	64	**28**
Water	60	OM	48	58	63	97	60	58	60	**46**	–	56
Hailfinder	49	OM	**27**	41	70	109	84	56	65	51	–	46
Hepar2	101	OM	68	87	123	177	119	86	127	79	55	**52**
Win95pts	59	OM	68	65	112	146	109	60	105	85	–	**42**

Table 8. The SHD of all algorithms for benchmarks with 10,000 instances.

	PC.stable	GOBNILP	MAHC	MMHC	DAG-GNN	GOLEM	DAGMA	GES	MMPC+GES	ILGES	ARGES	PPGES
Asia	3	**0**	2	3	8	13	8	5	5	7	2	3
Sachs	**0**	**0**	1	**0**	16	21	11	**0**	**0**	**0**	22	**0**
Alarm	10	OM	29	27	50	86	35	46	43	18	90	**5**
Child	1	**0**	**0**	18	22	44	24	15	18	**0**	60	**0**
Insurance	46	OM	33	38	52	69	52	26	45	45	105	**25**
Water	50	OM	48	50	64	97	60	52	61	57	–	**44**
Hailfinder	45	OM	**26**	42	74	117	85	44	61	56	–	34
Hepar2	91	OM	**23**	66	122	168	119	116	121	75	58	53
Win95pts	108	OM	79	46	110	156	107	44	102	98	–	**22**

Table 9. The SHD of all algorithms for benchmarks with 100,000 instances.

PPGES		MAHC	
BIC	SHD	BIC	SHD
− 10,095.77	34	− 17,110.85	40

Table 10. The performance of PPGES and MAHC on the COVID-19 dataset.

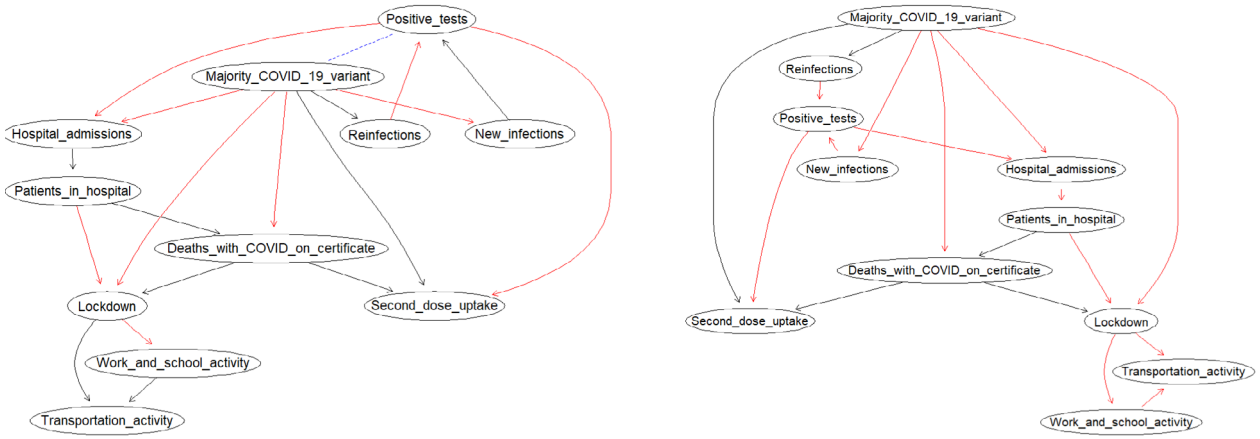

Fig. 6. Subgraphs of causal structures recovered from (**a**) PPGES and (**b**) MAHC in the COVID-19 dataset.

PPGES discovers the causal relationships between the policy node *Lockdown* and the mobility node $Transportation\_activity$, and between the two mobility nodes $Work\_and\_school\_activity$ and $Transportation\_activity$. However, MAHC do not correctly search the edges $Lockdown \rightarrow Transportation\_activity$ and $Work\_and\_school\_activity \rightarrow Transportation\_activity$. PPGES incorrectly finds an undirected edge $Positive\_tests - Majority\_COVID\_19\_variant$, while MAHC does not make the same mistake.

Discussion

In this study, we proposed an improved E-space algorithm called PPGES to improve the performance of GES. First, the strongest dependence relations were determined. In order to find the relations behind the data, we designed a metric called PPRE based on the relative entropy of the prior and posterior distributions. Then, for the proposed metric, we constructed a framework containing well-designed connection rules to obtain an excellent initial graph of GES. The connected graph is converted into E-space and further optimized using GES to obtain the final CPDAG. The experimental results reveal that the proposed algorithm improves the performance of GES in terms of time and scores. Moreover, PPGES outperformed other solvers in most models with different instances. Finally, PPGES also output a more accurate structure for a real dataset about COVID-19.

Our future works will focus on parallelizing the algorithm to further improve its efficiency and studying how to achieve the same excellent results under weak faithfulness conditions.

Data availability

All data generated or analysed during this study are included in this paper (including, but not limited to, repository name, author, and URL).

Received: 20 May 2025; Accepted: 19 September 2025
Published online: 24 October 2025

References

1. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Elsevier, 2014).
2. Jansen, R. et al. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science* **302**, 449–453 (2003).
3. Feng, J., Lansford, J. L., Katsoulakis, M. A. & Vlachos, D. G. Explainable and trustworthy artificial intelligence for correctable modeling in chemical sciences. *Sci. Adv.* **6**, eabc3204 (2020).
4. Wang, B. & Liu, X. Fuzzy-probabilistic time series forecasting combining Bayesian network and fuzzy time series model. *Symmetry* **17**, 275 (2025).
5. Phan, T. D., Smart, J. C., Capon, S. J., Hadwen, W. L. & Sahin, O. Applications of Bayesian belief networks in water resource management: A systematic review. *Environ. Model. Softw.* **85**, 98–111 (2016).
6. Hu, M., Li, X., Li, M., Zhu, R. & Si, B. A framework for analyzing fraud risk warning and interference effects by fusing multivariate heterogeneous data: A Bayesian belief network. *Entropy* **25**, 892 (2023).
7. Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A. & Jordan, M. A linear non-Gaussian acyclic model for causal discovery. *J. Mach. Learn. Res.* **7** (2006).

8. ZHANG, K. On the identifiability of the post-nonlinear causal model. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, Vol. 647 (AUAI Press, 2009).
9. Hoyer, P., Janzing, D., Mooij, J. M., Peters, J. & Schölkopf, B. Nonlinear causal discovery with additive noise models. *Adv. Neural Inf. Process. Syst.* **21** (2008).
10. Glymour, C., Zhang, K. & Spirtes, P. Review of causal discovery methods based on graphical models. *Front. Genet.* **10**, 524 (2019).
11. Spirtes, P. & Glymour, C. An algorithm for fast recovery of sparse causal graphs. *Soc. Sci. Comput. Rev.* **9**, 62–72 (1991).
12. Tsamardinos, I., Aliferis, C. F., Statnikov, A. R. & Statnikov, E. Algorithms for large scale Markov blanket discovery. In *FLAIRS*, Vol. 2, 376–381 (2003).
13. Colombo, D. et al. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.* **15**, 3741–3782 (2014).
14. Wu, X. et al. Accurate Markov boundary discovery for causal feature selection. *IEEE Trans. Cybern.* **50**, 4983–4996 (2019).
15. Jaber, A., Kocaoglu, M., Shanmugam, K. & Bareinboim, E. Causal discovery from soft interventions with unknown targets: Characterization and learning. *Adv. Neural. Inf. Process. Syst.* **33**, 9551–9561 (2020).
16. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* 461–464 (1978).
17. Suzuki, J. A construction of bayesian networks from databases based on an mdl principle. In *Uncertainty in Artificial Intelligence*, 266–273 (Elsevier, 1993).
18. Buntine, W. Theory refinement on Bayesian networks. In *Uncertainty Proceedings 1991*, 52–60 (Elsevier, 1991).
19. Cussens, J. Bayesian network learning with cutting planes. In *27th Conference on Uncertainty in Artificial Intelligence*, 153–160 (AUAI Press, 2011).
20. Bartlett, M. & Cussens, J. Integer linear programming for the Bayesian network structure learning problem. *Artif. Intell.* **244**, 258–271 (2017).
21. Yuan, C., Malone, B. & Wu, X. Learning optimal Bayesian networks using a* search. In *Twenty-second International Joint Conference on Artificial Intelligence* (2011).
22. Wang, Z., Gao, X., Tan, X. & Liu, X. Learning Bayesian networks using a* search with ancestral constraints. *Neurocomputing* **451**, 107–124 (2021).
23. Liao, Z. A., Sharma, C., Cussens, J. & van Beek, P. Finding all bayesian network structures within a factor of optimal. In *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 33, 7892–7899 (2019).
24. Heckerman, D., Geiger, D. & Chickering, D. M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **20**, 197–243 (1995).
25. Liu, X., Gao, X., Wang, Z., Ru, X. & Zhang, Q. A metaheuristic causal discovery method in directed acyclic graphs space. *Knowl.-Based Syst.* **276**, 110749 (2023).
26. Constantinou, A. C., Liu, Y., Kitson, N. K., Chobtham, K. & Guo, Z. Effective and efficient structure learning with pruning and model averaging strategies. *Int. J. Approx. Reason.* **151**, 292–321 (2022).
27. Chickering, D. M. Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.* **2**, 445–498 (2002).
28. Gillispie, S. B. & Perlman, M. D. Enumerating markov equivalence classes of acyclic digraph dels. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 171–177 (2001).
29. Meek, C. *Graphical Models: Selecting Causal and Statistical Models*. Ph.D. thesis, (Carnegie Mellon University, 1997).
30. Teyssier, M. & Koller, D. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 584–590 (2005).
31. Wang, Z., Gao, X., Tan, X. & Liu, X. Determining the direction of the local search in topological ordering space for Bayesian network structure learning. *Knowl.-Based Syst.* **234**, 107566 (2021).
32. Scanagatta, M., de Campos, C. P., Corani, G. & Zaffalon, M. Learning bayesian networks with thousands of variables. *Adv. Neural Inf. Process. Syst.***28** (2015).
33. Tsamardinos, I., Brown, L. E. & Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.* **65**, 31–78 (2006).
34. Scutari, M., Graafland, C. E. & Gutiérrez, J. M. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *Int. J. Approx. Reason.* **115**, 235–253 (2019).
35. Chickering, D. M. Optimal structure identification with greedy search. *J. Mach. Learn. Res.* **3**, 507–554 (2002).
36. Nielsen, J. D., Kočka, T. & Peña, J. M. On local optima in learning Bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 435–442 (2002).
37. Chen, F., Wang, X. & Rao, Y. Learning Bayesian networks using genetic algorithm. *J. Syst. Eng. Electron.* **18**, 142–147 (2007).
38. Zhang, Y., Zhang, W. & Xie, Y. Improved heuristic equivalent search algorithm based on maximal information coefficient for Bayesian network structure learning. *Neurocomputing* **117**, 186–195 (2013).
39. Alonso-Barba, J. I. et al. Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes. *Int. J. Approx. Reason.* **54**, 429–451 (2013).
40. Ramsey, J., Glymour, M., Sanchez-Romero, R. & Glymour, C. A million variables and more: The fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *Int. J. Data Sci. Anal.* **3**, 121–129 (2017).
41. Alonso, J. I., Ossa, L., Gamez, J. A. & Puerta, J. M. On the use of local search heuristics to improve GES-based Bayesian network learning. *Appl. Soft Comput.* **64**, 366–376 (2018).
42. Nandy, P., Hauser, A. & Maathuis, M. H. High-dimensional consistency in score-based and hybrid structure learning. *Ann. Stat.* **46**, 3151–3183 (2018).
43. Liu, X., Gao, X., Ru, X., Tan, X. & Wang, Z. Improving greedy local search methods by switching the search space. *Appl. Intell.* **53**, 22143–22160 (2023).
44. Liu, X., Gao, X., Ru, X. & Wang, Z. A hybrid Bayesian network structure learning algorithm in equivalence class space. In *2023 8th International Conference on Control and Robotics Engineering (ICCRE)*, 1–4 (IEEE, 2023).
45. Laborda, J. D., Torrijos, P., Puerta, J. M. & Gámez, J. A. Parallel structural learning of Bayesian networks: Iterative divide and conquer algorithm based on structural fusion. *Knowl.-Based Syst.* **296**, 111840 (2024).
46. Chen, Y. & Tian, J. Finding the k-best equivalence classes of Bayesian network structures for model averaging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28 (2014).
47. Chen, E. Y.-J., Choi, A. C. & Darwiche, A. Enumerating equivalence classes of Bayesian networks using EC graphs. In *Artificial Intelligence and Statistics*, 591–599 (PMLR, 2016).
48. Zheng, X., Aragam, B., Ravikumar, P. K. & Xing, E. P. Dags with no tears: Continuous optimization for structure learning. *Adv. Neural Inf. Processing Syst.* **31** (2018).
49. Yu, Y., Chen, J., Gao, T. & Yu, M. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, 7154–7163 (PMLR, 2019).
50. Yu, Y., Gao, T., Yin, N. & Ji, Q. DAGS with no curl: An efficient DAG structure learning approach. In *International Conference on Machine Learning*, 12156–12166 (PMLR, 2021).
51. Ng, I., Ghassami, A. & Zhang, K. On the role of sparsity and DAG constraints for learning linear DAGS. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (eds Larochelle, H. et al.) (2020).

52. Bello, K., Aragam, B. & Ravikumar, P. DAGMA: learning DAGS via m-matrices and a log-determinant acyclicity characterization. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022* (eds Koyejo, S. et al.) (2022).
53. VERMA, T. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, 1991* (Elsevier, 1991).
54. Koller, D. & Friedman, N. *Probabilistic Graphical Models: Principles and Techniques* (MIT Press, 2009).
55. Dor, D. & Tarsi, M. A simple algorithm to construct a consistent extension of a partially oriented graph. *Technicial Report R-185, Cognitive Systems Laboratory, UCLA* 45 (1992).
56. Chickering, D. M. A transformational characterization of equivalent bayesian network structures. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 87–98 (1995).
57. Scutari, M. Learning Bayesian networks with the bnlearn r package. *J. Stat. Softw.* **35** (2010).
58. Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H. & Bühlmann, P. Causal inference using graphical models with the r package pcalg. *J. Stat. Softw.* **47**, 1–26 (2012).
59. Zhang, K. *et al.* gcastle: A python toolbox for causal discovery (2021). arXiv:2111.15155.
60. Tukey, J. W. *Exploratory Data Analysis* (Reading/Addison-Wesley, 1977).
61. Constantinou, A. et al. Open problems in causal structure learning: A case study of covid-19 in the UK. *Expert Syst. Appl.* **234**, 121069 (2023).

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61573285, 62262016, 52402453). This work was also sponsored by the Innovation Foundation for Doctor Dissertation of Northwestern Polytechnical University (CX2022047). This work was also supported by the Fundamental Research Funds for the Central Universities (G2022KY0602). This work was also supported by the Open Project of the State Key Laboratory of Intelligent Game(Grant No. ZBKF-23-05). This work was also supported by the Key Research and Development Program of Shaanxi Province (Grant No.2023-GHZD-33).

Author contributions

Conceptualization, Xiaohan Liu; methodology, Xiaohan Liu and Ziyi Yang; software, Xiaohan Liu; validation, Xiaohan Liu; formal analysis, Xiaohan Liu and Qi Feng; investigation, Xiaohan Liu; resources, Xiaohan Liu and Ziyi Yang; data curation, Xiaohan Liu and Ziyi Yang; writing???original draft preparation, Xiaohan Liu and Shuying Wu; writing???review and editing, Xiaohan Liu, Shuying Wu, and Yuqing Yang; visualization, Xiaohan Liu and Chuchao He; supervision, Xiaoguang Gao; project administration, Xiaohan Liu, Qi Feng, Ziyi Yang, Shuying Wu, Yuqing Yang, Jia Ren and Chuchao He; funding acquisition, Xiaoguang Gao.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to X.G.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.