entropy

MDPI

Article

Learning Causes of Functional Dynamic Targets: Screening and Local Methods

Ruiqi Zhao [1], Xiaoxia Yang [2,*] and Yangbo He [1,*]

[1] School of Mathematical Sciences, Peking University, Beijing 100871, China; zhaorq@pku.edu.cn
[2] College of Science, Beijing Forestry University, Beijing 100083, China
* Correspondence: yxx77@bjfu.edu.cn (X.Y.); heyb@math.pku.edu.cn (Y.H.)

Abstract: This paper addresses the challenge of identifying causes for functional dynamic targets, which are functions of various variables over time. We develop screening and local learning methods to learn the direct causes of the target, as well as all indirect causes up to a given distance. We first discuss the modeling of the functional dynamic target. Then, we propose a screening method to select the variables that are significantly correlated with the target. On this basis, we introduce an algorithm that combines screening and structural learning techniques to uncover the causal structure among the target and its causes. To tackle the distance effect, where long causal paths weaken correlation, we propose a local method to discover the direct causes of the target in these significant variables and further sequentially find all indirect causes up to a given distance. We show theoretically that our proposed methods can learn the causes correctly under some regular assumptions. Experiments based on synthetic data also show that the proposed methods perform well in learning the causes of the target.

Keywords: screening method; local structure learning; functional dynamic target; direct causes; indirect causes

1. Introduction

Identifying the causes of a target variable is a primary objective in numerous research studies. Sometimes, these target variables are dynamic, observed at distinct time intervals, and typically characterized by functions or distinct curves that depend on other variables and time. We call them functional dynamic targets. For example, in nature, the growth of animals and plants is usually multistage and nonlinear with respect to time [1–3]. The popular growth curve functions, including Logistic, Gompertz, Richards, Hossfeld IV, and Double-Logistic functions, have *S* shapes [3], and have been widely used to model the patterns of growth. In psychological and cognitive science, researchers usually fit individual learning and forgetting curves by power functions; individuals may have different curve parameters [4,5].

The causal graphical model is widely used for the automated derivation of causal influences in variables [6–9] and demonstrates excellent performance in presenting complex causal relationships between multiple variables and expressing causal hypotheses [7,10,11]. In this paper, we aim to identify the underlying causes of these functional dynamic targets using the graphical model. There are three main challenges for this purpose. Firstly, identifying the causes is generally more challenging than exploring associations, even though the latter has received substantial attention, as evidenced by the extensive use of Genome-Wide Association Studies (GWAS) within the field of bioinformatics. Secondly, it is difficult to use a causal graphical model to represent the generating mechanism of dynamic targets and to find the causes of the targets from observational data when the number of variables is very large. For example, one needs to find the genes that affect the growth curve of individuals from more than thousands of Single-Nucleotide Polymorphisms (SNPs). Finally, the variables considered are mixed, which increases the complexity of representing and learning the causal model. We discuss these three challenges in detail below.

check for updates

Citation: Zhao, R.; Yang, X.; He, Y. Learning Causes of Functional Dynamic Targets: Screening and Local Methods. *Entropy* **2024**, *26*, 541. https://doi.org/10.3390/e26070541

Academic Editor: Adam Lipowski

Received: 2 April 2024
Revised: 13 June 2024
Accepted: 23 June 2024
Published: 24 June 2024

First of all, traditional statistical methods can only discover correlations between variables, rather than causal relationships, which may give false positive or false negative results for finding the real causes of the target. In fact, the association between the target and variables may originate from different causal mechanisms. For example, Figure 1 displays several different causal mechanisms possibly resulting in a statistically significant association between the target and variables. In Figure 1, X_1, X_2, X_3 are three random variables, $\mathbf{Y} = (Y_1, \cdots, Y_n)$ is a vector representing a functional dynamic target, in which $Y_i, i = 1, \ldots, n$ are states of the target in n time points, and direct edges represent direct causal relations among them. Using the statistical methods, we are very likely to find that X_1 is associated with $\mathbf{Y}$ significantly in all four cases. However, it is hard to identify whether X_1 is a real cause of $\mathbf{Y}$ without further causal learning. As shown in Figure 1, X_1 might be a direct cause of $\mathbf{Y}$ in Figure 1a,c, a cause but not a direct cause in Figure 1b, and not a cause in Figure 1d.

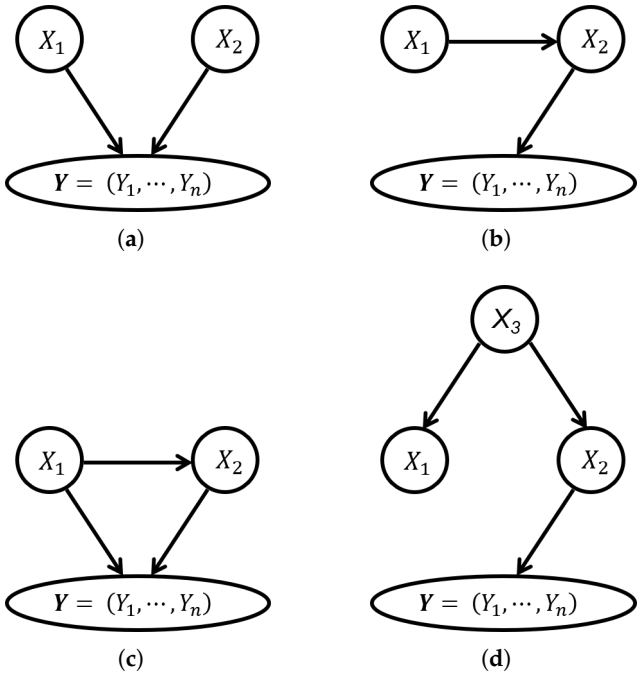

Figure 1. The causal graphs when the variable X_1 is significantly associated with $\mathbf{Y}$: (**a**) V-structures. (**b**) Chains. (**c**) Triangles. (**d**) Forks.

In addition, when the number of candidate variables is very huge, both learning causal structures and discovering target causes become very difficult. In fact, learning the complete causal graph is redundant and wasteful for the task of finding causes, as the focus should be on the target variable's local structure. PCD-by-PCD algorithm [12] is adept at identifying such local structures and efficiently distinguishing parents, children, and some descendants. The MB-by-MB method [13] , in contrast, simplifies this by learning Markov Blanket (MB) sets for identifying direct causes/effects, leveraging simpler and quicker techniques compared with PCD sets with methods like PCMB, STMB, and EEMB [14–16]. The CMB algorithm further streamlines this process using a topology-based MB discovery approach [17]. However, Ling [18] pointed out that Expand-Backtracking-type algorithms, such as the PCD-by-PCD and CMB algorithms, may overlook some v-structures, leading to numerous incorrect edge orientations. To tackle these issues, the APSL algorithm was introduced and designed to learn the subgraph within a specific distance centered around the target variable. Nonetheless, its dependence on the FCBF method for Markov Blanket learning tends to produce approximate sets rather than precise ones [19]. Furthermore, Ling [18] emphasized learning the local graph within a certain distance from the target rather than focusing on the causes of the target.

Finally, the variables in question are varied; specifically, the targets consist of dynamic time series or complex curves, while the other variables may be either discrete or continuous. Consequently, measuring the connections between the target and other variables presents significant challenges. For instance, traditional statistical methods used to assess independence or conditional independence between variables and complex targets might not only be inefficient but also ineffective, especially when there is an insufficient sample size to accurately measure high-order conditional independence.

In this paper, we introduce a causal graphical model tailored for analyzing dynamic targets and propose two methods to identify the causes of such a functional dynamic target assuming no hidden variables or selection biases. Initially, after establishing our dynamic target causal graphical model, we conduct an association analysis to filter out most variables unrelated to the target. With data from the remaining significantly associated variables, we then combine the screening method with structural learning algorithms and introduce the SSL algorithm to identify the causes of the target. Finally, to mitigate the distance effects that can mask the association between a cause and the target in data sets where the causal chain from cause to target is excessively long, we propose a local method. This method initially identifies the direct causes of the target and then proceeds to learn the causes sequentially in reverse order along the causal path.

The main contributions of this paper include the following:

- We introduce a causal graphical model that combines Bayesian networks and functional dynamic targets to represent the causal mechanism of variables and the target.
- We present a screening method that significantly reduces the dimensions of potential factors and combines it with structural learning algorithms to learn the causes of a given target and prove that all identifiable causes can be learned correctly.
- We propose a screening-based and local method to learn the causes of the functional dynamic target up to any given distance among all factors. This method is helpful when association disappears due to the long distance between indirect causes and the target.
- We experimentally study our proposed method on a simulation data set to demonstrate the validity of the proposed methods.

2. Preliminary

Before introducing the main results of this paper, we need to clarify some definitions and notations related to graphs. Furthermore, unless otherwise specified, we use capital letters such as V to denote variables or vertices, boldface letters such as $\mathbf{V}$ to denote variable sets or vectors, and lowercase letters such as v and $\mathbf{v}$ to denote the realization of a variable or vector, respectively.

A graph $\mathcal{G}$ is a pair $(\mathbf{V}, \mathbf{E})$, in which $\mathbf{V} = \{V_1, \cdots, V_p\}$ is the vertex set and $\mathbf{E} \subseteq \mathbf{E}^*(\mathbf{V}) := (\mathbf{V} \times \mathbf{V}) \backslash \{(V_i, V_i) \mid V_i \in \mathbf{V}\}$ is the edge set. To simplify the symbols, we use $\mathbf{V}$ to represent both random variables and the corresponding nodes in the graph. For any two nodes $V_i, V_j \in \mathbf{V}$, an undirected edge between V_i and V_j, denoted by $V_i - V_j$, is an edge satisfying $(V_i, V_j) \in \mathbf{E}$ and $(V_j, V_i) \in \mathbf{E}$, while a directed edge between V_i and V_j, denoted by $V_i \to V_j$, is an edge satisfying $(V_i, V_j) \in \mathbf{E}$ and $(V_j, V_i) \notin \mathbf{E}$. If all edges in a graph are undirected (directed), the graph is called an undirected (directed) graph. If a graph has both undirected and directed edges, then it is called a partially directed graph. For a given graph $\mathcal{G}$, we use $\mathbf{V}(\mathcal{G})$ and $\mathbf{E}(\mathcal{G})$ to denote its vertex set and edge set, respectively, where $\mathcal{G}$ can be an undirected, directed, or partially directed graph. For any $\mathbf{V}' \subseteq \mathbf{V}$, the induced subgraph of $\mathcal{G}$ over $\mathbf{V}'$, denoted by $\mathcal{G}(\mathbf{V}')$ or $\mathcal{G}_{\mathbf{V}'}$, is the graph with vertex set $\mathbf{V}'$ and edge set $\mathbf{E}(\mathbf{V}') \subseteq \mathbf{E}$ containing all and only edges between vertices in $\mathbf{V}'$, that is, $\mathcal{G}_{\mathbf{V}'} = (\mathbf{V}', \mathbf{E}(\mathbf{V}'))$, where $\mathbf{E}(\mathbf{V}') := \mathbf{E} \cap (\mathbf{V}' \times \mathbf{V}')$.

In a graph $\mathcal{G}$, V_i is a parent of V_j and V_j is a child of V_i if the directed edge $V_i \to V_j$ is in $\mathcal{G}$. V_i and V_j are neighbors of each other if the undirected edge $V_i - V_j$ is in $\mathcal{G}$. V_i and V_j are called adjacent if they are connected by an edge, regardless of whether the edge is directed or undirected. We use $Pa(V_i, \mathcal{G}), Ch(V_i, \mathcal{G}), Ne(V_i, \mathcal{G}), Adj(V_i, \mathcal{G})$ to denote the sets of parents, children, neighbors, and adjacent vertices of V_i in $\mathcal{G}$, respectively. For any vertex

set $\mathbf{V}' \subseteq \mathbf{V}$, the parent set of $\mathbf{V}'$ in $\mathcal{G}$ can be defined as $Pa(\mathbf{V}', \mathcal{G}) = \cup_{V_i \in \mathbf{V}'} Pa(V_i, \mathcal{G}) \backslash \mathbf{V}'$. The sets of children, neighbors, and adjacent vertices of $\mathbf{V}'$ in $\mathcal{G}$ can be defined similarly. A root vertex is the vertex without parents. For any vertex $V_i \in \mathbf{V}$, the degree of V_i in $\mathcal{G}$, denoted by $deg(V_i, \mathcal{G})$, is the number of V_i's adjacent vertices, that is, $deg(V_i, \mathcal{G}) = |Adj(V_i, \mathcal{G})|$. The skeleton of $\mathcal{G}$, denoted by $\mathcal{G}_s$, is an undirected graph obtained by transforming all directed edges in $\mathcal{G}$ to undirected edges, that is, $\mathcal{G}_s := (\mathbf{V}, \mathbf{E}_S)$, where $\mathbf{E}_s := \{(V_i, V_j) \in \mathbf{V} \times \mathbf{V} \mid V_i \in Adj(V_j, \mathcal{G})\}$.

The sequence $< V_{i_1}, \cdots, V_{i_n} >$ in graph $\mathcal{G}$ is an ordered collection of distinct vertices $V_{i_1}, \cdots, V_{i_n}$. A sequence becomes a path, denoted by $(V_{i_1}, \cdots, V_{i_n})$, if every pair of consecutive vertices in the sequence is adjacent in $\mathcal{G}$. The vertices V_{i_1} and V_{i_n} serve as the endpoints, with the rest being intermediate vertices. For a path $\pi = (V_{i_1}, \cdots, V_{i_n})$ in $\mathcal{G}$, and for any $1 \leq k \leq n$, the subpath from V_{i_1} to V_{i_k} is $\pi(V_{i_1}, V_{i_k}) = (V_{i_1}, \cdots, V_{i_k})$, and path π can thus be represented as a combination of its subpaths, denoted by $\pi = \pi(V_{i_1}, V_{i_k}) \oplus \pi(V_{i_k}, V_{i_n})$. A path is partially directed if there is no directed edge $V_{i_{k+1}} \to V_{i_k}$ in $\mathcal{G}$ for any $k = 1, \ldots, n-1$. A partially directed path is directed (or undirected) if all its edges are directed (or undirected). A vertex V_i is an ancestor of V_j and V_j is a descendant of V_i if there exists a directed path from V_i to V_j or $V_i = V_j$. The sets of ancestors and descendants of V_i in the graph $\mathcal{G}$ are denoted by $An(V_i, \mathcal{G})$ and $De(V_i, \mathcal{G})$, respectively. Furthermore, a vertex V_i is a possible ancestor of V_j and V_j is a possible descendant of V_i if there is a partially directed path from V_i to V_j. The sets of possible ancestors and possible descendants of V_i in graph $\mathcal{G}$ are denoted by $PossAn(V_i, \mathcal{G})$ and $PossDe(V_i, \mathcal{G})$, respectively. For any vertex set $\mathbf{V}' \subseteq \mathbf{V}$, the ancestor set of $\mathbf{V}'$ in graph $\mathcal{G}$ is $An(\mathbf{V}', \mathcal{G}) := \cup_{V_i \in \mathbf{V}'} An(V_i, \mathcal{G})$. The sets of possible ancestors and (possible) descendants of $\mathbf{V}'$ in graph $\mathcal{G}$ can be defined similarly.

A (directed, partially directed, or undirected) cycle is a (directed, partially directed, or undirected) path from a node to itself. The length of a path (cycle) is the number of edges on the path (cycle). The distance between two variables V_i and V_j is the length of the shortest directed path from V_i to V_j. A directed acyclic graph (DAG) is a directed graph without directed cycles, and a partially directed acyclic graph (PDAG) is a partially directed graph without directed cycles. A chain graph is a partially directed graph in which all partially directed cycles are undirected. This indicates that both DAGs and undirected graphs can be considered as specific types of chain graphs.

In a graph $\mathcal{G}$, a v-structure is a tuple (V_i, V_j, V_k) satisfying $V_i \to V_j \leftarrow V_k$ with $V_i \notin Adj(V_k, \mathcal{G})$, in which V_j is called a collider. A path π is d-separated (blocked) by a set of vertices $\mathbf{Z}$ if (1) π contains a chain $V_i \to V_j \to V_k$ or a fork $V_i \leftarrow V_j \to V_k$ with $V_j \in \mathbf{Z}$; (2) π contains a v-structure $V_i \to V_j \leftarrow V_k$ with $De(V_j, \mathcal{G}) \notin \mathbf{Z}$, and is d-connected otherwise [20]. Sets of vertices $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ if and only if $\mathbf{Z}$ blocks all paths from any vertex $V_i \in \mathbf{X}$ to any vertex $V_j \in \mathbf{Y}$, denoted by $\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z}$. Furthermore, for any distribution P, $\mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$ denotes that $\mathbf{X}$ and $\mathbf{Y}$ are conditional independent given $\mathbf{Z}$. Given a DAG $\mathcal{G}$ and a distribution P, the Markov condition holds if $\mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z}$, while faithfulness holds if $\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$. In fact, for any distribution, there exists at least one DAG such that the Markov condition holds, but there are some certain distributions that do not satisfy faithfulness to any DAG. Therefore, unlike the Markov condition, faithfulness is often regarded as an assumption. In this paper, unless otherwise stated, we assume that faithfulness holds, that is, $\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Leftrightarrow \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$. For simplicity, we use the symbol $\perp\!\!\!\perp$ to denote both (conditional) independence and d-separation.

From the concepts described, it can be inferred that a DAG characterizes the (conditional) independence relationships among a set of variables. In fact, multiple different DAGs may characterize the same conditional independent relationship. According to the Markov condition and faithfulness assumption, if the d-separation relationship contained in two DAGs is exactly the same, then these two DAGs are said to be Markov equivalent. Furthermore, two DAGs are Markov equivalent if and only if they share the same skeleton and v-structures [21]. All Markov equivalent DAGs constitute a Markov equivalent class, which can be represented by a completely partially directed acyclic graph (CPDAG) $\mathcal{G}^*$. Two vertices are adjacent in the CPDAG $\mathcal{G}^*$ if and only if they are adjacent in all DAGs in

the equivalent class. The directed edge $V_i \to V_j$ in CPDAG $\mathcal{G}^*$ indicates that this directed edge appears in all DAGs within the equivalent class, whereas the undirected edge $V_i - V_j$ signifies that $V_i \to V_j$ is present in some DAGs and $V_i \leftarrow V_j$ in others within the equivalent class [22]. A CPDAG is a chain graph [23] and can be learned by observational data and Meek's rules [24] (Figure 2).

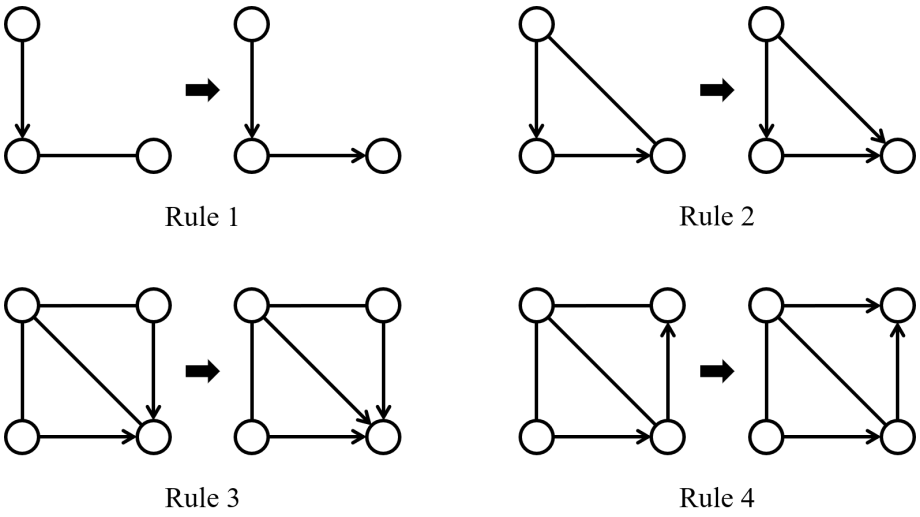

Figure 2. Meek's rules comprise four orientation rules. If the graph on the left-hand side of a rule is an induced subgraph of a PDAG, then the corresponding rule can be applied to replace an undirected edge in the induced subgraph with a directed edge. This replacement results in the induced subgraph transforming into the graph depicted on the right-hand side of the rule.

3. The Causal Graphical Model of Potential Factors and Functional Dynamic Target

Let $\mathbf{X} = \{X_1, \cdots, X_p\}$ be a set of random variables representing potential factors and $\mathbf{Y} = (Y_1, \cdots, Y_q)$ be a functional dynamic target, where Y_i, for $i = 1, \ldots, q$, represents the state of the target at q different time points. Let $\mathcal{G}$ be a DAG defined over $\mathbf{X} \cup \mathbf{Y}$, and let $\mathcal{G}_\mathbf{X}$ be the subgraph induced by $\mathcal{G}$ over the set of potential factors $\mathbf{X}$. Suppose that the causal network of $\mathbf{X}$ can be represented by $\mathcal{G}_\mathbf{X}$, and when combined with the joint probabilities over $\mathbf{X}$, denoted by $P(\cdot)$, we obtain a causal graphical model $(\mathcal{G}_\mathbf{X}, P)$. Consequently, the data generation mechanisms of $\mathbf{X}$ and $\mathbf{Y}$ follow a causal Bayesian network model of $\mathcal{G}_\mathbf{X}$ and a model determined by the direct causes $Pa(\mathbf{Y}, \mathcal{G})$ of $\mathbf{Y}$, respectively. Formally, we define a causal graphical model of the functional dynamic target as follows.

Definition 1. *Let $\mathcal{G}$ be a DAG over $\mathbf{X} \cup \mathbf{Y}$, $Pa(\mathbf{Y}, \mathcal{G})$ denote the direct causes of $\mathbf{Y}$ in $\mathcal{G}$ and $Ch(\mathbf{Y}, \mathcal{G}) = \emptyset$, $P(\cdot)$ be a joint distribution over $\mathbf{X}$, and Θ be parameters determining the expectations of the functional dynamic target $\mathbf{Y}$, which is influenced by $Pa(\mathbf{Y}, \mathcal{G})$. Then, the triple $(\mathcal{G}, P(\cdot), \Theta)$ constitutes a causal graphical model for $\mathbf{Y}$ if the following two conditions hold:*

1. *The pair $(\mathcal{G}_\mathbf{X}, P)$ constitutes a Bayesian network model for $\mathbf{X}$.*
2. *The functional dynamic target $\mathbf{Y}$ follows the following model :*

$$\mathbf{Y} = \tilde{\mu}(\Theta) + \tilde{\epsilon}_\mathbf{Y}, \tag{1}$$

where $\tilde{\mu}(\Theta) = (\mu(t_1, \Theta), \cdots, \mu(t_q, \Theta))$ is the vector of the mean function at time $t_1, \ldots, t_q$, and $\tilde{\epsilon}_\mathbf{Y} = (\epsilon_{\mathbf{Y},t_1}, \ldots, \epsilon_{\mathbf{Y},t_q})$ is the vector of error terms with mean of zero, that is, $E(\epsilon_{\mathbf{Y},t_i}) = 0$, $i = 1, \ldots, q$.

Different functional dynamic targets use different mean functions. For example, the optimal mean function of growth curves of different species varies from the Gompertz function, $\mu(t, (a, b, c)) = ae^{-be^{-ct}}$, the Richards function, $\mu(t, (a, b, c, d)) = a/(1 + be^{-ct})^d$, the Hossfeld function, $\mu(t, (a, k, c)) = at^k/(c + t^k)$, and the Logistic function, $\mu(t, (a, r, c)) = a/(1 +$

$e^{-r(t-c)}$) to Double-Logistic function, $\mu(t,(a_1,r_1,c_1,a_2,r_2,c_2)) = a_1/(1+e^{-r_1(t-c_1)}) + a_2/(1+e^{-r_2(t-c_2)})$ [25–27].

A causal graphical model of the functional dynamic target can be interpreted as a data generation mechanism of variables in $\mathbf{X}$ and $\mathbf{Y}$ as follows. First, the root variables in $\mathcal{G}_{\mathbf{X}}$ are generated according to their marginal probabilities. Then, following the topological ordering of the DAG $\mathcal{G}_{\mathbf{X}}$, for any non-root-variable X, when its parent nodes $Pa(X,\mathcal{G})$ have been generated, X can be drawn from $P(X \mid Pa(X,\mathcal{G}))$, which is the conditional probability of X given its parent set $Pa(X,\mathcal{G})$. Finally, the target is generated by Equation (1). According to Definition 1, the Markov condition holds for the causal graphical model of a dynamic target, that is, for any pair of variables X_i and X_j, the d-separation of X_i and X_j given a set $\mathbf{Z}$ in $\mathcal{G}$ implies that X_i and X_j are conditionally independent given $\mathbf{Z}$.

Given a mean function $\mu(t,\Theta)$, we can estimate parameters $\hat{\Theta}$ as follows,

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{n} \sum_{j=1}^{q} \left(y_{i,t_j} - \mu(t_j,\Theta)\right)^2,$$

where n and q represent the number of individuals and the length of the functional dynamic target, respectively. The residual sum of squares (RSS) is minimized at $\hat{\Theta}$. The Akaike information criterion (AIC) can be used to select the appropriate mean function μ^* to fit the functional dynamic targets. We have

$$\mu^* = \arg\min_{\mu}(nq + nq\log(2\pi) + nq\log(RSS/q) + 2|\hat{\Theta}|).$$

4. Variable Screening for Causal Discovery

For the set of potential factors $\mathbf{X}$ and the functional dynamic target $\mathbf{Y}$, our task is to find the direct causes and all causes of $\mathbf{Y}$ up to a given distance. An intuitive method involves learning the causal graph $\mathcal{G}$ to find all causes of $\mathbf{Y}$. Alternatively, we could first learn the causal graph $\mathcal{G}_{\mathbf{X}}$ and then identify all variables that have directed paths to $\mathbf{Y}$. However, as mentioned in Section 1, this intuitive approach has three main drawbacks. To address these challenges, we propose a variable screening method to reduce the number of potential factors, and a hypothesis testing method to test for (conditional) independence between potential factors and $\mathbf{Y}$. By integrating these methods with structural learning approaches, we have developed an algorithm capable of learning and identifying all causes of functional dynamic targets.

Let X be a variable with level K. The variable X is not independent of $\mathbf{Y}$ if there exists at least two values of X, say $X = x_1$ and $X = x_2$, such that the conditional distributions of $\mathbf{Y}$ given $X = x_1$ and $X = x_2$ are different. Conversely, if the conditional distribution of $\mathbf{Y}$ given $X = x$ remains unchanged for any x, we have that X and $\mathbf{Y}$ are independent. Let Θ_x be the parameter of the mean function of the functional dynamic target with $X = x$. To ascertain whether the variable X is not independent of $\mathbf{Y}$, we implement the following test:

$$H_0 : \Theta_x = \Theta_{x'}, \quad \forall x, x' \in \{1,\ldots,K\}, \tag{2}$$

$$H_1 : \Theta_x \neq \Theta_{x'}, \quad \exists x, x' \in \{1,\ldots,K\}. \tag{3}$$

Let $\mathbf{y}_i = (y_{i,t_1},\ldots,y_{i,t_q})$ be the ith sample of the functional dynamic target with $X = x_i$. Under the null hypothesis, $\mathbf{y}_i$ is modeled as $\mathbf{y}_i = \tilde{\mu}(\Theta) + \tilde{\epsilon}$, whereas under the alternative hypothesis, it is modeled as $\mathbf{y}_i = \tilde{\mu}(\Theta_{x_i}) + \tilde{\epsilon}$. Let $\ln L(\mathbf{Y},\Theta)$ denote the unrestricted log-likelihood of $\mathbf{Y}$ under H_0 and let $\ln L(\mathbf{Y},\Theta^{H_1}) = \sum_{x=1}^{K} \ln L(\mathbf{Y},\Theta_x)$ denote the restricted log-likelihood of $\mathbf{Y}$ under H_1. The likelihood ratio statistic is calculated as follows:

$$LR = -2(\ln L(\mathbf{Y},\Theta) - \ln L(\mathbf{Y},\Theta^{H_1})). \tag{4}$$

Under certain regular conditions, the statistic LR approximately follows χ^2 distribution, and the degrees of freedom of this χ^2 distribution are determined by the difference in the numbers of parameters between H_0 and H_1, as specified in Equations (2) and (3).

Therefore, by applying hypothesis tests described in Equations (2) and (3) to each potential factor, we can identify all variables significantly associated with the dynamic target. We denote these significant variables as $\mathbf{X}_{sig}$, defined as $\mathbf{X}_{sig} = \{X \mid X \in \mathbf{X}, X \not\perp\!\!\!\perp \mathbf{Y}\}$. Indeed, since the mean function of the dynamic target depends on its direct causes, which in turn depend on indirect causes, the dynamic target ultimately depends on all its causes. Therefore, when X is precisely a cause of $\mathbf{Y}$, we can reject the null hypothesis in Equation (2), implying that $\mathbf{X}_{sig}$ includes all causes of the dynamic target, assuming no statistical errors. Therefore, given a dynamic target $\mathbf{Y}$, perform hypothesis testing of H_0 against H_1 as defined in Equations (2) and (3) to each potential factor sequentially, then we can obtain the set $\mathbf{X}_{sig}$ and their corresponding p-values $\{X_{pv}\}_{X \in \mathbf{X}_{sig}}$, in which X_{pv} is the p-value of the variable $X \in \mathbf{X}_{sig}$.

A causal graphical model, as described in Definition 1, necessitates adherence to the Markov conditions for variables and the functional dynamic target. Given the Markov condition and the faithfulness assumption, a natural approach to identifying the causes of the functional dynamic target involves learning the causal structure of $\mathbf{X}_{sig}$ and subsequently discerning the relationship between each variable $X \in \mathbf{X}_{sig}$ and $\mathbf{Y}$. For significant variables $\mathbf{X}_{sig}$, we present the following theorem, with its proof available in Appendix A.2:

Theorem 1. *Suppose that $(\mathcal{G}, P, \Theta)$ constitutes a causal graphical model for the functional dynamic target $\mathbf{Y}$ as defined in Definition 1, with the faithfulness assumption being satisfied. Let $\mathbf{X}_{sig}$ denote the set comprising all variables in $\mathbf{X}$ that are dependent on $\mathbf{Y}$. Then, the following assertions hold:*

1. *$\mathbf{X}_{sig}$ consists of all causes and the descendants of these causes of $\mathbf{Y}$, that is, $\mathbf{X}_{sig} = An(\mathbf{Y}, \mathcal{G}) \cup De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})$.*
2. *For any two variables $X_1, X_2 \in \mathbf{X}_{sig}$, if either X_1 or X_2 is a cause of $\mathbf{Y}$, then X_1, X_2 are not adjacent in $\mathcal{G}_{\mathbf{X}_{sig}}$ if and only if there exists a set $\mathbf{A} \in \mathbf{X}_{sig}$ such that $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$.*
3. *For any two variables $X_1, X_2 \in \mathbf{X}_{sig}$, if there exists a set $\mathbf{A} \in \mathbf{X}_{sig}$ such that $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$, then X_1, X_2 are not adjacent in $\mathcal{G}_{\mathbf{X}_{sig}}$.*

The first result of Theorem 1 implies the soundness and rationality of the method for finding $\mathbf{X}_{sig}$ mentioned above. The second result indicates that when at least one end of an edge is a cause of $\mathbf{Y}$, this edge can be accurately identified (in terms of its skeleton, not its direction) using any well-known structural learning methods, such as the PC algorithm [28] and GES algorithm [29]. Contrasting with the second result, the third specifies that for any pair of variables $X_1, X_2 \in \mathbf{X}_{sig}$, if a separation set exists in $\mathbf{X}_{sig}$ that blocks X_1, X_2, then these variables are not adjacent in the true graph $\mathcal{G}$. However, the converse does not necessarily hold due to the potential presence of a confounder or common cause $X_3 \notin \mathbf{X}_{sig}$, which can led to the appearance of an extraneous edge between X_1 and X_2 in the causal graph $\mathcal{G}$ derived solely from data on $\mathbf{X}_{sig}$. To accommodate this, the CPDAG learned from $\mathbf{X}_{sig}$ is denoted as $\mathcal{G}'_{\mathbf{X}_{sig}}$, and the induced subgraph that corresponds to the true graph $\mathcal{G}$ over $\mathbf{X}_{sig}$ is represented as $\mathcal{G}^*_{\mathbf{X}_{sig}}$. An illustrative example follows to elaborate on this explanation.

Example 1. *In Figure 3, Figure 3a presents a true graph defined over $\mathbf{X} = \{X_1, \ldots, X_5\}$ and $\mathbf{Y}$. Here, the set of significant variables is $\mathbf{X}_{sig} = \{X_1, X_2, X_3, X_5\}$, and X_4 is independent of $\mathbf{Y}$. Figure 3b illustrates the induced subgraph $\mathcal{G}^*_{1,\mathbf{X}_{sig}}$ of the CPDAG $\mathcal{G}^*_1$ over the set $\mathbf{X}_{sig}$, while Figure 3c displays the graph learned through the structural learning method, such as the PC algorithm, applied to $\mathbf{X}_{sig}$. It should be noted that, in $\mathcal{G}_1$, $\{X_4\}$ is a separation set of X_1 and X_2, that is, $X_1 \perp\!\!\!\perp X_2 \mid X_4$. However, since $X_4 \notin \mathbf{X}_{sig}$ and structural learning only utilize data concerning $\mathbf{X}_{sig}$, no separation set exists for X_1 and X_2 in $\mathbf{X}_{sig}$. Consequently, X_1 and X_2 appear adjacent in the learned graph $\mathcal{G}'_{1,\mathbf{X}_{sig}}$. Furthermore, given $X_2 \perp\!\!\!\perp X_3$ and $X_2 \not\perp\!\!\!\perp X_3 \mid X_1$, the structural learning method identifies a v-structure $X_2 \to X_1 \leftarrow X_3$. A similar process yiel $X_1 \to X_2 \leftarrow X_5$.*

Therefore, a bidirected edge $X_1 \leftrightarrow X_2$ appears in the learned graph $\mathcal{G}'_{1,\mathbf{X}_{sig}}$ but not in $\mathcal{G}^*_{1,\mathbf{X}_{sig}}$, as highlighted by the red edge in Figure 3c.

Similarly, Figure 3d presents a true graph $\mathcal{G}_2$ defined over $\mathbf{X} = \{X_1, \ldots, X_5\}$ and $\mathbf{Y}$. In this scenario, the set of significant variables is identified as $\mathbf{X}_{sig} = \{X_1, X_2, X_3, X_5\}$, with X_4 being independent of $\mathbf{Y}$. Figure 3b depicts the induced subgraph $\mathcal{G}^*_{2,\mathbf{X}_{sig}}$ of the CPDAG $\mathcal{G}^*_2$ over $\mathbf{X}_{sig}$, while Figure 3c illustrates the graph learned through the structural learning method, such as the PC algorithm, applied to $\mathbf{X}_{sig}$. In $\mathcal{G}_2$, the set $\{X_1, X_4\}$ acts as a separation set between X_2 and X_3, indicating $X_2 \perp\!\!\!\perp X_3 \mid (X_1, X_4)$. However, with $X_4 \notin \mathbf{X}_{sig}$ and structural learning relying solely on data concerning $\mathbf{X}_{sig}$, a separation set for X_2 and X_3 in $\mathbf{X}_{sig}$ no longer exists. As a result, X_2 and X_3 appear adjacent in the learned graph $\mathcal{G}'_{2,\mathbf{X}_{sig}}$. Furthermore, given $X_3 \perp\!\!\!\perp X_5$ and $X_3 \not\perp\!\!\!\perp X_5 \mid X_2$, the structural learning method is capable of identifying a v-structure $X_3 \rightarrow X_2 \leftarrow X_5$. Therefore, a directed edge $X_3 \rightarrow X_2$ is present in the learned graph $\mathcal{G}'_{2,\mathbf{X}_{sig}}$ but not in $\mathcal{G}^*_{2,\mathbf{X}_{sig}}$, as highlighted by the red edge in Figure 3f.

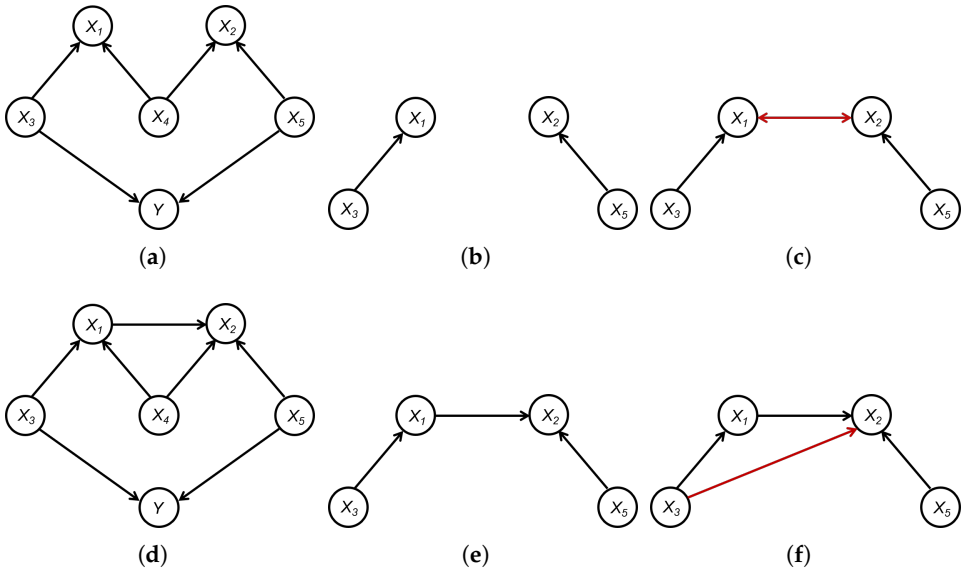

Figure 3. An example to illustrate the difference between $\mathcal{G}'_{\mathbf{X}_{sig}}$ and $\mathcal{G}^*_{\mathbf{X}_{sig}}$: (**a**) True graph $\mathcal{G}_1$. (**b**) $\mathcal{G}^*_{1,\mathbf{X}_{sig}}$. (**c**) $\mathcal{G}'_{1,\mathbf{X}_{sig}}$. (**d**) True graph $\mathcal{G}_2$. (**e**) $\mathcal{G}^*_{2,\mathbf{X}_{sig}}$. (**f**) $\mathcal{G}'_{2,\mathbf{X}_{sig}}$.

Example 1 illustrates two scenarios in which the graph $\mathcal{G}'_{\mathbf{X}_{sig}}$ might include false positive edges that do not exist in $\mathcal{G}^*_{\mathbf{X}_{sig}}$. Importantly, these additional false edges may not appear between the causes of $\mathbf{Y}$. Instead, they may occur between the causes and noncauses of $\mathbf{Y}$, or exclusively among the noncauses of $\mathbf{Y}$, as delineated in Theorem 1. The complete result is given by Proposition A1 in Appendix A.2. Indeed, a more profound inference can be drawn: the presence of extra edges does not compromise the structural integrity concerning the causes of $\mathbf{Y}$, affecting neither the skeleton nor the orientation.

Theorem 2. *The edges in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$, if exists, do not affect the skeleton or orientation of edges among the ancestors of $\mathbf{Y}$ in $\mathcal{G}^*$. Furthermore, we have $An(\mathbf{Y}, \mathcal{G}^*_{\mathbf{X}_{sig} \cup \mathbf{Y}}) = An(\mathbf{Y}, \mathcal{G}'_{\mathbf{X}_{sig} \cup \mathbf{Y}})$ and $PossAn(\mathbf{Y}, \mathcal{G}^*_{\mathbf{X}_{sig} \cup \mathbf{Y}}) \subseteq PossAn(\mathbf{Y}, \mathcal{G}'_{\mathbf{X}_{sig} \cup \mathbf{Y}})$, where $\mathcal{G}^*_{\mathbf{X}_{sig} \cup \mathbf{Y}}$ and $\mathcal{G}'_{\mathbf{X}_{sig} \cup \mathbf{Y}}$ are graphs by adding a node $\mathbf{Y}$ and directed edges from $\mathbf{Y}$'s direct causes to $\mathbf{Y}$ in graphs $\mathcal{G}^*_{\mathbf{X}_{sig}}$ and $\mathcal{G}'_{\mathbf{X}_{sig}}$, respectively.*

According to Theorem 2, it is evident that although the graph $\mathcal{G}'_{\mathbf{X}_{sig}}$ obtained through structural learning does not exactly match the induced subgraph of CPDAG $\mathcal{G}^*$ over $\mathbf{X}_{sig}$ corresponding to the true graph, the causes of the functional dynamic target $\mathbf{Y}$ in these two graphs are identical, including the structure among these causes. Thus, in terms of identifying the causes, the two graphs can be considered equivalent. Furthermore,

Theorem 2 indicates that all possible ancestors of $\mathbf{Y}$ in $\mathcal{G}^*_{\mathbf{X}_{sig}\cup\mathbf{Y}}$ are also possible ancestors in $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$, though the converse may not hold. The detailed proof is available in Appendix A.2.

Example 2. *The true graph $\mathcal{G}$ is given by Figure 4a, and the corresponding CPDAG $\mathcal{G}^*$ is itself, that is, $\mathcal{G} = \mathcal{G}^*$. In this case, the set of significant variables is $\mathbf{X}_{sig} = \{X_1, X_2, X_4\}$. Figure 4b is the induced graph $\mathcal{G}^*_{\mathbf{X}_{sig}\cup\mathbf{Y}}$ of $\mathcal{G}$ ($\mathcal{G}^*$) over $\mathbf{X}_{sig}$, and Figure 4c is the CPDAG $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$ obtained by using the structural learning method on $\mathbf{X}_{sig}$. Then, we have $An(\mathbf{Y}, \mathcal{G}^*_{\mathbf{X}_{sig}\cup\mathbf{Y}}) = An(\mathbf{Y}, \mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}) = \{X_1\}$, while $PossAn(\mathbf{Y}, \mathcal{G}^*_{\mathbf{X}_{sig}\cup\mathbf{Y}}) = \emptyset \subseteq \{X_2, X_4\} = PossAn(\mathbf{Y}, \mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}})$.*

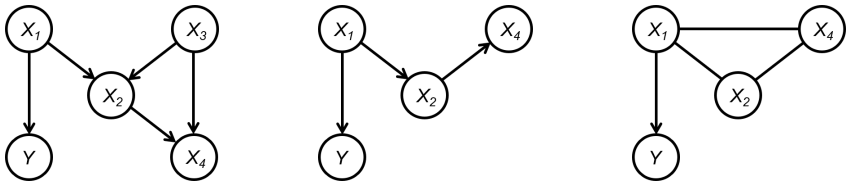

Figure 4. An example to illustrate the results in Theorem 2: (**a**) True graph $\mathcal{G}$. (**b**) $\mathcal{G}^*_{\mathbf{X}_{sig}\cup\mathbf{Y}}$. (**c**) $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$.

According to the causal graphical model in Definition 1 and the faithfulness assumption, $\mathbf{Y}$ is the sum of the mean function and an independent noise, and the mean function is a deterministic function of $\mathbf{Y}$'s direct causes. Therefore, for any nondescendant of $\mathbf{Y}$, say X, given the direct causes of $\mathbf{Y}$, that is, $Pa(\mathbf{Y}, \mathcal{G})$, X is independent of $\mathbf{Y}$. On the contrary, for any $X \in \mathbf{X}_{sig}$, X is a direct cause of $\mathbf{Y}$ if and only if there is no subset $\mathbf{A} \subseteq Adj(X, \mathcal{G})$ such that $X \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}$.

Let $\mathbf{A}$ be a subset of $\mathbf{X}_{sig}$. For any $X \in \mathbf{X}_{sig}$ and $X \notin \mathbf{A}$, to test the conditional independence $X \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}$, consider the following test:

$$H_0 : \Theta_{\mathbf{A}=\mathbf{a},X=x} = \Theta_{\mathbf{A}=\mathbf{a},X=x'}, \quad \forall \mathbf{a}, x, x', \tag{5}$$

$$H_1 : \Theta_{\mathbf{A}=\mathbf{a},X=x} \neq \Theta_{\mathbf{A}=\mathbf{a},X=x'}, \quad \exists \mathbf{a}, x, x'. \tag{6}$$

Under the null hypothesis, the parameter only depends on the value of the set $\mathbf{A}$, which can be denoted as $\Theta_{\mathbf{A}}$, while under the alternative hypothesis, the parameter is determined by the values of both $\mathbf{A}$ and X, which can be denoted as $\Theta_{\mathbf{A},X}$. Let $\ln L(\mathbf{Y}, \Theta_{\mathbf{A}})$ be the log-likelihood of $\mathbf{Y}$ under H_0, and $\ln L(\mathbf{Y}, \Theta_{\mathbf{A},X})$ be the log-likelihood of $\mathbf{Y}$ under H_1. The likelihood ratio statistic is

$$LR = -2(\ln L(\mathbf{Y}, \Theta_{\mathbf{A}}) - \ln L(\mathbf{Y}, \Theta_{\mathbf{A},X})). \tag{7}$$

Under certain regular conditions, the statistic LR approximately follows χ^2 distribution, with degrees of freedom equal to $|\Theta_{\mathbf{A},X}| - |\Theta_{\mathbf{A}}|$.

Based on the above results, we propose a screening and structural learning-based algorithm to identify the causes of the functional dynamic target $\mathbf{Y}$, as detailed in Algorithm 1.

In Algorithm 1, the initial step involves learning the structure over $\mathbf{X}_{sig}$ utilizing data related to $\mathbf{X}_{sig}$ through a structural learning method, detailed in Lines 1–6. The notation $X * - Y$ in Lines 3–4 signifies that the connection between X and Y could be either $X - Y$ or $X \leftarrow Y$. We first learn the skeleton of $\mathbf{X}_{sig}$ following the same procedure as the PC algorithm (Line 1), with the details in Appendix A.1. Nevertheless, due to the potential occurrence of bidirected edges, adjustments are made in identifying v-structures (Lines 2–5), culminating in the elimination of all bidirected edges. According to Theorem 1, these bidirected edges, which are removed directly (Line 5), are present only between causative and noncausative variables or among noncausative variables of the functional dynamic target. Since these variable pairs are inherently (conditional) independent, removing such edges does not compromise the (conditional) independence relationships among the remaining variables,

as shown in Theorem 2 and Example 1. Subsequently, we designate the set of direct causes as $DC := \mathbf{X}_{sig}$ and sequence these variables in ascending order of their correlations with $\mathbf{Y}$ (Lines 7–8). This is because variables with weaker correlation are less likely to be the direct cause of $\mathbf{Y}$. Placing these variables at the beginning of the sequence can quickly exclude non-direct-cause variables in the subsequent conditional independence tests, thereby enhancing the algorithm's efficiency, simplifying its complexity, and reducing the required number of conditional independence tests. Next, we add directed edges from all vertices in $\mathbf{X}_{sig}$ to $\mathbf{Y}$ (Line 9) to construct the graph $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$. For each directed edge, say $X \to \mathbf{Y}$, we check the conditional independence of X and $\mathbf{Y}$ given a subset $\mathbf{A}_X$ of DC (Lines 12–14). In seeking the separation set $\mathbf{A}_X$, the search starts with single-element sets, progressing to sets comprising two elements, and so forth. Upon identifying a separation set, both vertices and directed edges are removed from DC and $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$, respectively (Lines 15–17). Lastly, if the separation set's size k surpasses that of DC, implying that no conditional independence of X and $\mathbf{Y}$ can be found given any subset of DC, the directed edge $X \to \mathbf{Y}$ remains in $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$.

Algorithm 1 SSL: Screening and structural learning-based algorithm

Require: $\mathbf{X}_{sig}$ and their corresponding p-values $\{X_{pv}\}_{X\in\mathbf{X}_{sig}}$, data sets about $\mathbf{X}_{sig}$ and $\mathbf{Y}$.
Ensure: Causes of $\mathbf{Y}$.

1: Learn the skeleton $\mathcal{G}'_s(\mathbf{X}_{sig})$ of the CPDAG $\mathcal{G}'_{\mathbf{X}_{sig}}$ defined on $\mathbf{X}_{sig}$ and obtain corresponding separation sets $\mathcal{S}$ based on the data set related to $\mathbf{X}_{sig}$ via Algorithm A1 in Appendix A.1;
2: **repeat**
3: Find the structure $X * - Y - * Z$ satisfying $X \notin Adj(Z, \mathcal{G})$ in graph $\mathcal{G}'_s(\mathbf{X}_{sig})$;
4: If $Y \notin \mathbf{S}(X, Z)$, then orient as $X* \to Y \leftarrow *Z$;
5: **until** All structures $X * - Y - * Z$ with $X \notin Adj(Z, \mathcal{G})$ in $\mathcal{G}'_s(\mathbf{X}_{sig})$ have been tested;
6: Construct the CPDAG $\mathcal{G}'_{\mathbf{X}_{sig}}$ by deleting all bidirected edges and using Meek's rules to orient as many undirected edges as possible in graph $\mathcal{G}'_s(\mathbf{X}_{sig})$;
7: Let $DC := \mathbf{X}_{sig}$;
8: Sort DC in ascending order of associations with $\mathbf{Y}$ using $\{X_{pv}\}_{X\in\mathbf{X}_{sig}}$;
9: Let $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$ be the graph by adding a node $\mathbf{Y}$ to the graph $\mathcal{G}'_{\mathbf{X}_{sig}}$,and for each $X \in \mathbf{X}_{sig}$, add a directed edge $X \to \mathbf{Y}$ to the graph $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$;
10: Set $k := 1$;
11: **while** $k < |DC|$, **do**
12: **for** each vertex $X \in DC$, **do**
13: **for** each subset $\mathbf{A}_X$ of $DC\backslash\{X\}$ with k vertices, **do**
14: Test the conditional independence $\mathbf{Y} \perp\!\!\!\perp X \mid \mathbf{A}_X$ using Equations (5) and (6);
15: **if** $\mathbf{Y} \perp\!\!\!\perp X \mid \mathbf{A}_X$, **then**
16: Delete the directed edge $X \to \mathbf{Y}$ in graph $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$;
17: Let $DC := DC\backslash\{X\}$;
18: **end if**
19: **end for**
20: **end for**
21: $k := k + 1$;
22: **end while**
23: **return** $\mathcal{G}'_{\mathbf{X}_{sig}\cup\mathbf{Y}}$.

According to Theorem 1 and the discussion after Example 2, DC is the set of all direct causes of $\mathbf{Y}$ if all assumptions in Theorem 1 hold and all statistical tests are correct. Further, according to Theorem 2, all ancestors of $\mathbf{Y}$ can be obtained from the graph $\mathcal{G}'_{\mathbf{X}_{sig} \cup \mathbf{Y}}$. Therefore, Algorithm 1 can learn all the causes of $\mathbf{Y}$ correctly.

Note that in Algorithm 1, we first traverse the sizes of the separation set (Line 11) and then, for each given size, traverse all variables in the DC set and all possible separations with that size (Line 12 and 13) to test for the conditional independence of each variable and $\mathbf{Y}$. That is, first fix the size of the separation set to 1, and then traverse all variables. After all variables are traversed once, increase the size of the separation set to 2, and then traverse all variables again. The advantage of this arrangement is that it can quickly remove the nondirect causes of $\mathbf{Y}$ and reduce the size of the DC set, thereby reducing the number of conditional independence tests and improving their accuracy. Furthermore, it is worth mentioning that the reason why we directly add directed edges from variables in DC to $\mathbf{Y}$ in graph $\mathcal{G}'_{\mathbf{X}_{sig} \cup \mathbf{Y}}$ (Line 9) is because we assume the descendant set of $\mathbf{Y}$ is empty, as shown in Definition 1, and in this case, $\mathbf{Y}$'s adjacent set is exactly the direct causes we are looking for. If there is no such assumption, then it is necessary to judge the variables in $\mathbf{Y}$'s adjacent set and distinguish the parents from the children.

5. A Screening-Based and Local Algorithm

Based on the previous results and discussions, we can conclude that Algorithm 1 is capable of correctly identifying the causes of a functional dynamic target. However, Algorithm 1 requires recovering the complete causal structure of $\mathbf{X}_{sig}$ and $\mathbf{Y}$. As analyzed in Section 1, learning the complete structure is unnecessary for identifying the causes of the target. Furthermore, Algorithm 1 may be influenced by the distance effect, whereby the correlation between a cause and the target may diminish from the data when the path from the cause to the target is too lengthy. Consequently, identifying this cause variable through observational data becomes challenging, potentially leading to missed causes. Therefore, we propose a screening-based and local approach to address these challenges.

In this section, we introduce a three-stage approach to learn the causes of functional dynamic targets. Initially, utilizing the causal graphical model, we apply a hypothesis testing method to screen variables, identifying factors significantly correlated with the target. Subsequently, we employ a constraint-based method to find the direct causes of the target from these significant variables. Lastly, we present a local learning method to discover the causes of these direct causes within any specified distance. We begin with the introduction of a screening-based algorithm that can learn the direct causes of $\mathbf{Y}$, as shown in Algorithm 2.

In Algorithm 2, we initially set the set of direct causes $DC := \mathbf{X}_{sig}$ and arrange these variables in ascending order of their correlations with $\mathbf{Y}$ (Lines 1–2), which is the same as Algorithm 1. We introduce a set $\mathbf{N}_X$ to contain variables determined not to belong to X's separation set, starting as an empty set (Line 3). We then check the conditional independence of each variable $X \in DC$ with $\mathbf{Y}$. During the search for the separation set $\mathbf{A}_X$, $\mathcal{A}$ is set as all subsets of $DC \backslash (X \cup \mathbf{N}_X)$ with k variables and is arranged roughly in descending order of their associations with $\mathbf{Y}$ (Lines 7–8). This is because variables that have a stronger correlation with $\mathbf{Y}$ are more likely to be the direct causes and are also more likely to become the separation set of other variables. Placing these variables at the beginning of the order can quickly find the separation set of nondirect causes and remove these variables from DC, which can reduce the number of conditional independence tests and accelerate the algorithm. Once we find the separation set $\mathbf{A}_X$ for X and $\mathbf{Y}$, we remove X from DC and add X to $\mathbf{N}_V$ for each $V \in \mathbf{A}_X$ (Lines 11–13). This is because when $\mathbf{A}_X$ is the separation set of X and $\mathbf{Y}$, the variables in $\mathbf{A}_X$ appear in the path from X to $\mathbf{Y}$. Consequently, X should not be in the separation set for variables in $\mathbf{A}_X$ with respect to $\mathbf{Y}$. Compared with Algorithm 1, introducing $\mathbf{N}_X$ in Algorithm 2 improves efficiency and speed. While Algorithm 1 requires examining every subset of $DC \backslash X$ (Line 8 in Algorithm 1), Algorithm 2 only needs to evaluate subsets of $DC \backslash (X \cup \mathbf{N}_X)$ (Line 7 in Algorithm 2). The theoretical validation of Algorithm 2's correctness is presented below.

Algorithm 2 Screening-based algorithm for learning direct causes of $\mathbf{Y}$

Require: $\mathbf{X}_{sig}$ and their corresponding p-values $\{X_{pv}\}_{X\in\mathbf{X}_{sig}}$, data sets about $\mathbf{X}_{sig}$ and $\mathbf{Y}$.
Ensure: Direct causes of $\mathbf{Y}$.
1: Let $DC := \mathbf{X}_{sig}$;
2: Sort DC in ascending order of associations with $\mathbf{Y}$ using $\{X_{pv}\}_{X\in\mathbf{X}_{sig}}$;
3: Let $\mathbf{N}_X := \emptyset$ for each $X \in DC$;
4: Set $k := 1$;
5: **while** $k < |DC|$, **do**
6: **for** each vertex X in DC, **do**
7: Let $\mathcal{A}$ be the set of all subsets of $DC\backslash(\{X\} \cup \mathbf{N}_X)$ with k variables;
8: Sort $\mathcal{A}$ approximately in descending order of associations with $\mathbf{Y}$;
9: **for** each $\mathbf{A}_X \in \mathcal{A}$, **do**
10: Test the conditional independence $\mathbf{Y} \perp\!\!\!\perp X \mid \mathbf{A}_X$ using Equations (5) and (6);
11: **if** $\mathbf{Y} \perp\!\!\!\perp X \mid \mathbf{A}_X$, **then**
12: Set $DC := DC\backslash\{X\}$
13: Add X to $\mathbf{N}_V$ for each $V \in \mathbf{A}_X$;
14: break
15: **end if**
16: **end for**
17: **end for**
18: $k := k + 1$
19: **end while**
20: **return** DC.

Theorem 3. *If all assumptions in Theorem 1 hold, and there are no errors in the independence tests, then Algorithm 2 can correctly identify all direct causes of* $\mathbf{Y}$.

Next, we aim to identify all causes of $\mathbf{Y}$ within a specified distance. One natural method is to recursively apply Algorithm 2, starting with $\mathbf{Y}$'s direct causes and then expanding to their direct causes. This process continues until all causes within the set distance are found. However, this method's effectiveness for $\mathbf{Y}$ relies on the assumption that $\mathbf{Y}$ has no descendants, making its adjacent set its parent set. This is not the case for other variables. Thus, we must further analyze and distinguish variables in the adjacent set of other variables. Consequently, we introduce the LPC algorithm in Algorithm 3.

Algorithm 3 $LPC(T, \mathbf{U})$ algorithm

Require: a target node T, a data set over variables $\mathbf{X}$, a non-PC set $\mathbf{U}$.
Ensure: the PCD set of T and set $\mathbf{S}$ containing all separation relations.
1: Set $PCD := \{X : X \in \mathbf{X}\backslash\mathbf{U}$, and $T \not\perp\!\!\!\perp X\}$; $k := 1$; $\mathbf{S} := \emptyset$;
2: **while** $k < |PCD|$, **do**
3: **for** each vertex $X \in PCD$, **do**
4: **if** there exists $\mathbf{A}_X \subseteq PCD\backslash\{X\}$ such that $|\mathbf{A}_X| = k$ and $(T \perp\!\!\!\perp X \mid \mathbf{A}_X)$, **then**
5: Set $PCD := PCD\backslash\{X\}$ and add tuple $(T, X, \mathbf{A}_X)$ to $\mathbf{S}$;
6: **end if**
7: **end for**
8: $k := k + 1$;
9: **end while**
10: **return** PCD and $\mathbf{S}$.

Algorithm 3 aims to learn the local structure of a given target variable T, but in fact, the final PCD set includes T's Parents, Children, and Descendants. This is because when verifying the conditional independence (Line 4), we remove some nonadjacent variables of T in advance (Line 1), resulting in some descendant variables being unable to find the corresponding separation set.

Example 3. *In Figure 5, let* $T = X_1, \mathbf{U} = \emptyset$. *Since* $X_1 \perp\!\!\!\perp X_4$, *we initially have* $PCD = \{X_2, X_3\}$ *(Line* 1 *in Algorithm 3). Note that there originally exists a conditional independent relationship* $X_1 \perp\!\!\!\perp X_3 \mid (X_2, X_4)$ *in the graph, but since we remove the vertex* X_4 *in advance, there is no longer a separation set of* X_1 *and* X_3 *in the set of PCD. Therefore,* X_3 *cannot be removed from PCD further and the output* $PCD_{X_1} = \{X_2, X_3\}$, *that is,* X_3, *which is a descendant of* X_1 *but not a child of* X_1, *is included in* PCD_{X_1}.

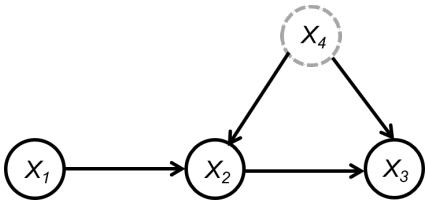

Figure 5. An example to illustrate the *PCD* set obtained by Algorithm 3.

Example 3 illustrates that there may indeed be some nonchildren descendants of the target variable in the *PCD* set obtained by Algorithm 3. Below, we show that one can identify these non-child-descendant variables by repeatedly applying Algorithm 3. For example, in Example 3, the *PCD* set of X_1 is $PCD_{X_1} = \{X_2, X_3\}$. Then, we can apply Algorithm 3 to X_3 and find that the *PCD* set of X_3 is $PCD_{X_3} = \{X_2, X_4\}$. It can be seen that X_1 is not in PCD_{X_3}. Hence, we can conclude that X_3 is a nonchildren descendant of X_1; otherwise, X_1 must be in PCD_{X_3}. Through this method, we can delete the non-child-descendant variables from the *PCD* set, so that the *PCD* set only contains the parents and children of the target variable. Based on this idea, we propose a step-by-step algorithm to learn all causes of a functional dynamic target locally, as shown in Algorithm 4.

Algorithm 4 PC-by-PC: Finding all causes of target **T** within a given distance

Require: a target set $\mathbf{T} \subseteq \mathbf{V} = \mathbf{X} \cup \mathbf{Y}$, data set over **V**, and the maximum distance m.
Ensure: all causes of **T** with length up to m.

1: Set $n := 1, n' := 0, CanC := \mathbf{T}$;
2: Initial graph $\mathcal{G}$ with directed edges from each vertex in **T** to an auxiliary node L;
3: **repeat**
4: Set $X = CanC_n$;
5: Let $\mathbf{U} = \{V : X \notin PC_V, \forall V \in CanC_{1:n-1}\}$;
6: Get the *PCD* set and the separation set $(PC_X, \mathbf{S}_X) = LPC(X, \mathbf{U})$;
7: **for** each $V \in PC_X \cap CanC_{1:n-1}$ **do**
8: **if** $X \in PC_V$ **then**
9: Add an undirected edge $X - V$ to graph $\mathcal{G}$;
10: **else**
11: $PC_X := PC_X \backslash \{V\}, PC_V := PC_V \backslash \{X\}$;
12: **end if**
13: **end for**
14: Update $\mathcal{G}$ by modifying structures like $V_1 - X - V_2$, $X - V_1 - V_2$ and $X - V_1 \leftarrow V_2$ to $V_1 \rightarrow X \leftarrow V_2$, $X \rightarrow V_1 \leftarrow V_2$ and $X \rightarrow V_1 \leftarrow V_2$ respectively, if the middle vertex is not in the separation set of the two end vertices;
15: **if** X is the last vertex of $CanC$ **then**
16: Update $\mathcal{G}$ by orientating undirected edges as much as possible via Meek's rule;
17: **for** each $V \in CanC_{n':n}$, **do**
18: Add $PC_V \backslash CanC$ to the end of $CanC$ if $|Path(V, L)| < m$ or the m-th edge close to L in $Path(V, L)$ is undirected;
19: **end for**
20: **end if**
21: $n' := n, n := n + 1$;
22: **until** X is the last vertex of $CanC$;
23: **return** $\mathcal{G}$ and **S**.

In Algorithm 4, $CanC_n$ represents the n-th variable in the set $CanC$, and $CanC_{1:n-1}$ represents the first to the $(n-1)$-th variable in the set $CanC$. $Path(V, L)$ denotes the shortest path from V to L in graph $\mathcal{G}$. There are many methods to learn the shortest path, such as the Dijkstra algorithm [30]. Algorithm 4 uses the mentioned symmetric validation method to remove descendants from the PCD set (Lines 7–13), and hence, we directly write the PCD set as the PC set (Line 6). When our task is to learn all causes of a functional dynamic target $\mathbf{Y}$, the target set $\mathbf{T}$ as the algorithm input is all direct causes of $\mathbf{Y}$, which can be obtained by Algorithm 2, and the auxiliary node L is exactly the functional dynamic target $\mathbf{Y}$ (Line 2). In fact, we can prove it theoretically, as shown below.

Theorem 4. *If the faithfulness assumption holds and all independence tests are correct, then Algorithm 4 can learn all causes of the input target set* $\mathbf{T}$ *within a given distance* m *correctly. Further, if all assumptions in Theorem 1 holds,* $\mathbf{T}$ *is the set of direct causes of the functional dynamic target* $\mathbf{Y}$*, and the auxiliary node* L *in Algorithm 4 is* $\mathbf{Y}$*, then Algorithm 4 can learn all causes of* $\mathbf{Y}$ *within a given distance* $(m+1)$ *correctly.*

Note that the above algorithm gradually spreads outward from the direct causes of $\mathbf{Y}$, and at each step, the newly added nodes are all in the PC set of previous nodes (Line 18), which only involves the local structure of all causes of $\mathbf{Y}$, greatly improving the efficiency and accuracy of the algorithm. Moreover, Algorithm 4 identifies the shortest path between each cause variable and $\mathbf{Y}$. When the m-th edge on one path from $\mathbf{Y}$ cannot be oriented, it only continues to expand from that path, instead of expanding all paths (Line 18 in Algorithm 4), which simplifies the algorithm and reduces the learning of redundant structures.

6. Experiments

In this section, we compare the effectiveness of different methods for learning the direct and all causes of a functional dynamic target through simulation experiments. As mentioned before, to our knowledge, existing structural learning algorithms lack the specificity needed to identify causes of functionally dynamic targets, so we only compare the methods we proposed, which are as follows:

1. SSL algorithm: The screening and structural learning-based algorithm given in Algorithm 1, which can learn both direct and all causes of a dynamic target simultaneously;
2. S-Local algorithm: First, use the screening-based algorithm given in Algorithm 2, which can learn direct causes of a functional dynamic target, and then use the PC-by-PC algorithm given in Algorithm 4, which can learn all causes of a functional dynamic target.

In fact, our proposed SSL algorithm integrates elements of the screening method with those of traditional constraint-based structural learning techniques, as depicted in Algorithm 1. In its initial phase, the SSL algorithm is a modified version of the PC algorithm, extending its capabilities to effectively handle bidirectional edges introduced by the screening process. This extension of the PC algorithm, tailored to address the causes of the dynamic target, positions the SSL algorithm as a strong candidate for a benchmark.

In this simulation experiment, we randomly generate a causal graph $\mathcal{G}$ consisting of a dynamic target $\mathbf{Y}$ and p = (15, 100, 1000, 10,000) potential factors. Additionally, we randomly select 1 to 2 variables from these potential factors to serve as direct causes for $\mathbf{Y}$. The potential factors are all discrete with finite levels, while the functional dynamic target $\mathbf{Y} = (Y_1, \cdots, Y_{24})$ is a continuous vector, and its mean function is a Double-Logistic function, that is,

$$Y_t = \mu_{t|Pa(\mathbf{Y},\mathcal{G})} + \epsilon_t, \quad t = 1, \ldots, 24,$$

where

$$\mu_{t|Pa(\mathbf{Y},\mathcal{G})} = \frac{a_{1|Pa(\mathbf{Y},\mathcal{G})}}{1+\exp(-r_{1|Pa(\mathbf{Y},\mathcal{G})}(t-c_{1|Pa(\mathbf{Y},\mathcal{G})}))} + \frac{a_{2|Pa(\mathbf{Y},\mathcal{G})}}{1+\exp(-r_{2|Pa(\mathbf{Y},\mathcal{G})}(t-c_{2|Pa(\mathbf{Y},\mathcal{G})}))},$$

and $\epsilon_t = \epsilon_{t-1} + \varepsilon_t, \varepsilon_t \sim N(0, 0.02^2)$. The $Pa(\mathbf{Y}, \mathcal{G})$ in the subscript of the above equations indicates that parameters are affected by the direct causes of $\mathbf{Y}$. For each causal graph $\mathcal{G}$, we randomly generate the corresponding causal mechanism, that is, the marginal and conditional distributions of potential factors and the functional dynamic target, and generate the simulation data from it. We use different sample sizes $n = (50, 100, 200, 500, 1000)$ and repeat the experiment 100 times for each sample size. In addition, we adopt adaptive significance level values in the experiment, because as the number of potential factors increases, the strength of screening also increases. In other words, as the number of potential factors p increases, the significance level α of the (conditional) independence test decreases. For example, α is 0.05 when $p = 100$, while α is 0.0005 when p = 10,000.

To evaluate the effectiveness of different methods, suppose $\mathbf{X}_l$ is the set of learned direct causes of $\mathbf{Y}$ by algorithms, and $\mathbf{X}_d$ is the set of true direct causes of $\mathbf{Y}$ in the generated graph. Then, let $TP = |\mathbf{X}_l \cap \mathbf{X}_d|$, $FP = |\mathbf{X}_l \backslash \mathbf{X}_d|$, $FN = |\mathbf{X}_d \backslash \mathbf{X}_l|$, and we have

$$recall = \frac{TP}{TP + FN}, \quad precision = \frac{TP}{TP + FP}, \quad accuracy = \frac{p - FP - FN}{p},$$

where p is the number of potential factors. It can be seen that the recall measures how much the algorithm has learned among all the true direct causes. Precision measures how much of the direct causes learned by the algorithm are correct. Accuracy measures the proportion of correct judgments on whether each variable is a direct cause or not. The evaluation indicators for learning all causes can also be defined similarly.

The experiment results are shown in Table 1, in which *time* represents the total time (in seconds) consumed by the algorithm, and *rec*, *prec*, *acc* represent the average value of recall, precision, and accuracy over 100 experiments, respectively. In addition, different subscripts represent different methods. *DC* and *AC* denote that algorithms learn direct causes and all causes, respectively.

Table 1. Experimental results of SSL algorithm and S-Local algorithm under different settings.

p	n	$time_{SSL}$	$time_{S-Local}$	$Cause$	rec_{SSL}	$rec_{S-Local}$	$prec_{SSL}$	$prec_{S-Local}$	acc_{SSL}	$acc_{S-Local}$
15	50	50	55	*DC*	0.487	0.500	0.352	0.474	0.866	0.929
				AC	0.157	0.487	0.721	0.814	0.445	0.640
	100	64	49	*DC*	0.557	0.829	0.485	0.814	0.905	0.975
				AC	0.143	0.656	0.786	0.877	0.451	0.733
	200	112	60	*DC*	0.538	0.885	0.386	0.856	0.888	0.981
				AC	0.224	0.882	0.603	0.878	0.476	0.854
	500	676	85	*DC*	0.551	0.939	0.466	0.929	0.927	0.989
				AC	0.363	0.977	0.567	0.893	0.552	0.916
	1000	1126	126	*DC*	0.778	0.981	0.691	0.963	0.957	0.996
				AC	0.566	0.996	0.702	0.890	0.667	0.923
100	50	251	277	*DC*	0.293	0.283	0.072	0.256	0.934	0.984
				AC	0.112	0.223	0.154	0.358	0.867	0.915
	100	224	227	*DC*	0.398	0.755	0.141	0.656	0.956	0.993
				AC	0.104	0.604	0.226	0.688	0.884	0.946
	200	290	235	*DC*	0.292	0.917	0.113	0.828	0.962	0.996
				AC	0.123	0.891	0.205	0.763	0.891	0.961
	500	890	336	*DC*	0.221	0.916	0.071	0.900	0.962	0.997
				AC	0.145	0.966	0.156	0.753	0.892	0.957
	1000	1509	527	*DC*	0.462	0.978	0.156	0.962	0.967	0.999
				AC	0.327	0.996	0.308	0.668	0.908	0.933

Table 1. *Cont.*

p	n	$time_{SSL}$	$time_{S-Local}$	$Cause$	rec_{SSL}	$rec_{S-Local}$	$prec_{SSL}$	$prec_{S-Local}$	acc_{SSL}	$acc_{S-Local}$
1000	50	836	839	*DC*	0.814	1.000	0.073	1.000	0.989	1.000
				AC	0.336	0.204	0.235	0.924	0.985	0.992
	100	936	962	*DC*	0.860	1.000	0.069	1.000	0.988	1.000
				AC	0.587	0.573	0.379	0.867	0.988	0.995
	200	1204	1222	*DC*	0.980	1.000	0.083	1.000	0.989	1.000
				AC	0.724	0.847	0.446	0.861	0.989	0.997
	500	2376	1930	*DC*	1.000	1.000	0.118	1.000	0.992	1.000
				AC	0.804	0.922	0.500	0.814	0.991	0.997
	1000	4015	3118	*DC*	1.000	1.000	0.121	1.000	0.993	1.000
				AC	0.873	0.998	0.523	0.813	0.992	0.998
10,000	50	9109	9480	*DC*	0.667	1.000	0.150	1.000	1.000	1.000
				AC	0.148	0.148	0.194	1.000	0.999	0.999
	100	10,008	10,463	*DC*	0.654	1.000	0.101	1.000	0.999	1.000
				AC	0.376	0.538	0.285	0.884	0.999	1.000
	200	13,710	13,836	*DC*	0.923	1.000	0.101	1.000	0.999	1.000
				AC	0.551	0.833	0.395	0.871	0.999	1.000
	500	21,084	18,343	*DC*	1.000	1.000	0.130	1.000	0.999	1.000
				AC	0.782	0.919	0.502	0.813	0.999	1.000
	1000	31,476	31,862	*DC*	1.000	1.000	0.126	1.000	0.999	1.000
				AC	0.813	0.959	0.505	0.787	0.999	1.000

In Table 1, since the SSL algorithm obtains direct and all causes simultaneously through complete structural learning, for the sake of fairness, we only count the total time for both algorithms. It can be seen that the time of the two algorithms is approximately linearly related to the number of potential factors p. Moreover, when p is fixed, the algorithm takes longer and longer as the sample size n increases. In fact, for SSL algorithms, most of the time is spent on learning the complete graph structure. Therefore, as n increases, the (conditional) independence test becomes more accurate, resulting in an increase in the size of set $\mathbf{X}_{sig}$ and a larger graph to learn, which naturally increases the time required. For the S-Local algorithm, more than 99% of the time is spent on optimizing the log-likelihood function during the (conditional) independence test in the screening stage. As n increases, the optimization time becomes longer and the total time also increases accordingly. This also explains why the time of the S-Local algorithm increases linearly as the number of variables increases, since the number of independence tests required increases roughly linearly. In addition, it can be seen that in most cases, the S-Local algorithm takes less time than the SSL algorithm, especially when p is small. However, when p is large, the time used by the two algorithms is similar. This is mainly because in this experimental setting, the mechanism of the functional dynamic target $\mathbf{Y}$ is relatively complex, and its mean function is a Double-Logistic function with too many parameters, which requires much time for optimization. In fact, even if there is only one binary direct cause, the mean function will have 13 parameters. When the mechanism of the functional dynamic target is relatively simple, the time required for the S-Local algorithm will also be greatly reduced. Besides, it should be noted that more than 99% of the time, the S-Local algorithm is used to check the independence in the screening step, and in practice, this step can be performed in parallel, which will greatly reduce the time required.

When learning the direct cause, whether it is recall, precision, or accuracy, the results of the S-Local algorithm are much higher than those of the SSL algorithm, especially the value of precision. The precision values of the SSL algorithm are very small, mainly because the accuracy of learning the complete graph structure is relatively low, resulting in learning many non-direct-cause variables in the local structure of $\mathbf{Y}$. Particularly when p is large, it is difficult to correctly recover the local structure of $\mathbf{Y}$. What's more, it should be noted that under the same sample size, when p is small, the values of recall, precision and accuracy obtained by S-Local algorithm are not as good as those obtained when p is large. For example, when

$p = 15$, $n = 50$, we have $rec_{S-Local} = 0.500$, $prec_{S-Local} = 0.474$ and $acc_{S-Local} = 0.929$, but when p = 10,000, n = 50, we have $rec_{S-Local}$ = 1.000, $prec_{S-Local}$ = 1.000 and $acc_{S-Local}$ = 1.000. The recall and accuracy values of the SSL algorithm also show similar results. This result does not violate our intuition, as we use adaptive significance levels in the experiment. When p is large, in order to increase the strength of screening and facilitate subsequent learning of all causes, we use a smaller significance level. Therefore, the algorithm is more rigorous in determining whether a variable is the direct cause of **Y** when learning direct causes, making it easier to exclude those non-direct-cause variables.

When learning all causes, the recall and accuracy values of the SSL algorithm and S-Local algorithm increase monotonically with respect to the sample size, and even in cases with many potential factors, both algorithms can achieve very good results. For example, when p = 10,000, the accuracy values of both algorithms are above 99.9%. Of course, overall, the results of the S-Local algorithm are significantly better than those of the SSL algorithm. However, it should be noted that the values of precision of the two algorithms show different trends. The precision value of the SSL algorithm increases monotonically with n when p is large, but the trend is not significant when p is small. This is because the SSL algorithm is affected by the distance effect, and as n gradually increases, (conditional) independence tests also become more accurate. As a result, many causes that are far away from **Y** can be identified. When p is large, the number of causes that are far away from **Y** is also large. Therefore, the precision of the SSL algorithm will gradually increase. However, when p is small, most variables have a short distance from **Y**. Although the SSL algorithm can also obtain more causes (the value of rec_{SSL} increases), it also includes some noncause variables that are strongly related to **Y** in the set of causes. At this time, the value of precision does not have a clear trend. On the other hand, the precision value of the S-Local algorithm monotonically increases with respect to n when p is small, and as p gradually increases, this trend gradually transforms into a monotonic decrease. This is because when p is small, as n increases, the S-Local algorithm can identify more causes through a more accurate (conditional) independence test. However, when p is large, the number of noncause variables obtained by the S-Local algorithm is greater than the number of causes. Therefore, the recall value still increases, but the precision value gradually decreases. In other words, in this case, there is a trade-off between the values of recall and precision of the S-Local algorithm. However, it should be noted that although the trends of precision values are different, the accuracy values of both algorithms increase with the increase in sample size.

It should be noted that the primary objective of the models and algorithms introduced in this paper is to identify the causes of functional dynamic targets, addressing the "Cause of Effect" (CoE) challenge, rather than directly predicting **Y**. However, based on the causal graphical model for these targets, correctly identifying **Y**'s direct causes is indeed sufficient for making accurate predictions. In the simulation experiment, with 15 nodes and 1000 samples, the Mean Squared Error (MSE) of prediction is 0.281 for simulations that incorrectly learn **Y**'s direct causes. This figure dropped to 0.185 when the causes were correctly identified, reflecting a significant reduction in prediction error of approximately 34%. Additionally, as illustrated in Table 1, the S-Local algorithm demonstrated exceptional accuracy in identifying the direct causes, with a success rate consistently above 98% in most cases. This high level of accuracy indicates that our algorithms perform well in predicting **Y** as well.

7. Discussion and Conclusions

In this paper, we first establish a causal graphical model for functional dynamic targets and discuss hypothesis testing methods for testing the (conditional) independence between random variables and functional dynamic targets. In order to deal with situations where there are too many potential factors, we propose a screening algorithm to screen out some variables that are significantly related to the functional dynamic target from a large number of potential factors. On this basis, we propose the SSL algorithm and S-Local algorithm to

learn the direct causes and all causes within a given distance of functional dynamic targets. The former utilizes the screening algorithm and structural learning methods to learn both the direct and all causes of functional dynamic targets simultaneously by recovering the complete graph structure of the screened variables. Its disadvantage is that learning the complete structure of the graph is very difficult and redundant, and it is also affected by the distance effect, resulting in a low accuracy in learning causes. The latter first uses a screening-based algorithm to learn the direct causes of functional dynamic targets, and then uses our proposed PC-by-PC algorithm, a step-by-step locally learning algorithm, to learn all causes within a given distance. The advantage of this algorithm is that all learning processes are controlled within the local structure of current nodes, making the algorithm no longer affected by the distance effect. In fact, this algorithm only focuses on the local structure of each cause variable, rather than learning the complete graph structure, greatly saving time and space. Moreover, the algorithm not only pays attention to the distance, but also can identify the direct path between each cause variable and the functional dynamic target, so that the algorithm does not need to identify the whole structure of a certain part but only learns the part of the local structure involving the cause variables, further reducing the learning of redundant structures.

It should be noted that when the causal mechanism of functional dynamic targets is very complex, the time required for the S-Local algorithm may greatly increase. In addition, the choice of significance level will also have an impact on the precision of the algorithm. Thus, how to simplify the causal model of functional dynamic targets and how to reasonably choose an appropriate significance level are two directions of our future work.

Author Contributions: Conceptualization, R.Z. and X.Y.; methodology, R.Z. and X.Y.; software, R.Z. and X.Y.; validation, R.Z. and Y.H.; formal analysis, R.Z., X.Y. and Y.H.; investigation, R.Z., X.Y. and Y.H.; resources, Y.H.; data curation, R.Z. and X.Y.; writing—original draft preparation, R.Z.; writing—review and editing, R.Z. and Y.H.; visualization, R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China grant number 2022ZD0160300.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The simulated data can be regenerated using the codes, which can be provided to the interested user via an email request to the correspondence author.

Acknowledgments: Thank Qingyuan Zheng for providing technical support.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1

In Algorithm 1, we first screen variables significantly related to $\mathbf{Y}$ through the hypothesis testing of H_0 against H_1 as defined in Equations (2) and (3), and learn the causal structure of these variables via a structural learning method. However, it should be noted that due to the absence of some variables in some separation sets, the graph we learn here is not the true CPDAG, but may have some extra edges, which may be directed or bidirected. Therefore, we need to make some modifications to the structural learning algorithm. As shown in Algorithm 1, we first learn the skeleton through the original structural learning method and then find v-structures by using a variant algorithm (Lines 2–5 in Algorithm 1). Hence, taking the PC algorithm as an example, we give the method to learn the skeleton $\mathcal{G}'_s(\mathbf{X}_{sig})$ below.

Algorithm A1 PC algorithm to learn the skeleton $\mathcal{G}'_s(\mathbf{X}_{sig})$.

Require: $\mathbf{X}_{sig}$, data sets about $\mathbf{X}_{sig}$.
Ensure: the skeleton $\mathcal{G}'_s(\mathbf{X}_{sig})$ and all separation sets $\mathcal{S}$.

Construct a complete undirected graph $\mathcal{G}$ defined over $\mathbf{X}_{sig}$;
Set $k := -1$;
repeat
  $k := k + 1$;
  repeat
    Find an ordered variable pair (X, Y) satisfying $|Adj(X, \mathcal{G})\backslash\{Y\}| \geq k$ in graph $\mathcal{G}$;
    repeat
      Find a subset $\mathbf{S} \subseteq Adj(X, \mathcal{G})\backslash\{Y\}$ with size k;
      if $X \perp\!\!\!\perp Y \mid \mathbf{S}$, **then**
        delete the undirected edge $X - Y$ from graph $\mathcal{G}$;
        save $\mathbf{S}$ as $\mathbf{S}(X, Y)$ and $\mathbf{S}(Y, X)$ and add them into $\mathcal{S}$;
      end if
    until The undirected edge $X - Y$ is deleted or all subsets $\mathbf{S} \subseteq Adj(X, \mathcal{G})\backslash\{Y\}$ have been selected;
  until The conditional independence test has been completed for all ordered variable pairs (X, Y) all sets $\mathbf{S}$ that meet the conditions;
until For each ordered variable pair (X, Y), we have $|Adj(X, \mathcal{G})\backslash\{Y\}| < k$;
return $\mathcal{G}$ and $\mathcal{S}$.

Appendix A.2

Proof of Theorem 1. We know that all causes of $\mathbf{Y}$ and the descendants of these causes are d-connected with $\mathbf{Y}$. Since the faithfulness assumption holds for the causal graphical model $(\mathcal{G}, P, \Theta)$, we have that all causes of $\mathbf{Y}$ and the descendants of these causes are not independent of $\mathbf{Y}$. From the definition $\mathbf{X}_{sig} = \{X \mid X \in \mathbf{X} \text{ and } X \not\perp\!\!\!\perp \mathbf{Y}\}$, we have that $\mathbf{X}_{sig}$ contains all causes of $\mathbf{Y}$ and the descendants of these causes. Meanwhile, according to the definition of the causal graphical model $(\mathcal{G}, P, \Theta)$, the functional dynamic target $\mathbf{Y}$ has no children. Therefore, all d-connected paths from a vertex X to $\mathbf{Y}$ should be either $X \to \cdots \to \cdots \to \mathbf{Y}$ or $X \leftarrow \cdots \leftarrow X' \to \cdots \to \mathbf{Y}$. Clearly, the vertex X is either a cause of $\mathbf{Y}$ or a descendant of a cause of $\mathbf{Y}$. Now, we have shown that the vertices in $\mathbf{X}_{sig}$ are either causes of $\mathbf{Y}$ or descendants of causes of $\mathbf{Y}$. Statement 1 is proved.

Now, we prove the Statement 2. The faithfulness assumption makes sure that X_1 and X_2 are not adjacent in $\mathcal{G}_{\mathbf{X}_{sig}}$ if there exists a set $\mathbf{A} \in \mathbf{X}_{sig}$ such that $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$. We just need to prove the "only if" part, that is, there exists a set $\mathbf{A} \in \mathbf{X}_{sig}$ such that $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$ if X_1 and X_2 are not adjacent in $\mathcal{G}_{\mathbf{X}_{sig}}$.

- Consider the case that both X_1 and X_2 are causes of $\mathbf{Y}$. Since X_1 and X_2 are not adjacent in $\mathcal{G}_{\mathbf{X}_{sig}}$, either X_1 is a nondescendant of X_2 or X_2 is a nondescendant of X_1. Without loss of generality, we assume that X_1 is a nondescendant of X_2. Since X_2 is a cause of $\mathbf{Y}$, according to Statement 1, we have $Pa(X_2, \mathcal{G}) = Pa(X_2, \mathcal{G}_{\mathbf{X}_{sig}}) \subseteq \mathbf{X}_{sig}$. Let $\mathbf{A} = Pa(X_2, \mathcal{G})$, we can obtain the result $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$.
- Consider the case that X_1 is a cause of $\mathbf{Y}$ and X_2 is not a cause of $\mathbf{Y}$. If X_2 is not a descendant of X_1, similar to the discussion in the previous case, we have $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$ where $\mathbf{A} = Pa(X_1, \mathcal{G}) = Pa(X_1, \mathcal{G}_{\mathbf{X}_{sig}}) \subseteq \mathbf{X}_{sig}$. If X_2 is a descendant of X_1, we have that all paths from X_2 to X_1 with the first directed edge as $X_2 \to \cdot$ are d-separated, otherwise, there should be a directed cycle. Let $\mathbf{A}_1$ consist of all parents of X_2 that are d-connected with X_1 in $\mathcal{G}$. Clearly, $\mathbf{A}_1 \subset \mathbf{X}_{sig}$ since X_1 is a cause of $\mathbf{Y}$. Then, let $\mathbf{A} = \mathbf{A}_1 \cup Pa(X_1, \mathcal{G}) \subseteq \mathbf{X}_{sig}$. Since $Pa(X_1, \mathcal{G})$ blocks all d-connected paths between

X_1 and X_2 through $Pa(X_1, \mathcal{G})$, and the set $\mathbf{A}_1$ blocks all paths like $X_1 \to \cdots \to X_2$, we have that the set $\mathbf{A}$ blocks all d-connected paths between X_1 and X_2 in $\mathcal{G}$, that is, $X_1 \perp\!\!\!\perp X_2 \mid \mathbf{A}$ holds.

- Consider the case that X_2 is a cause of $\mathbf{Y}$ and X_1 is not a cause of $\mathbf{Y}$, which is symmetric to the second case and can be discussed similarly.

Therefore, the "only if" part of Statement 2 is proven.

Statement 3 holds directly since the faithfulness assumption holds for the causal graphical model $(\mathcal{G}, P, \Theta)$. □

Proposition A1. *The edges in* $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$*, if they exist, do not appear between two ancestors of* $\mathbf{Y}$ *in* $\mathcal{G}$*.*

Proof of Proposition A1. For any two nonadjacent vertices $X_1, X_2 \in An(\mathbf{Y}, \mathcal{G})$, if there is no directed path from $X_1 (or\ X_2)$ to $X_2 (or\ X_1)$, then $Pa(X_1, \mathcal{G}) (or\ Pa(X_2, \mathcal{G})) \subseteq \mathbf{X}_{sig}$ is a separation set relative to the pair (X_1, X_2), and hence there is no edge between X_1 and X_2 in the graph $\mathcal{G}_{\mathbf{X}_{sig}}$. Therefore, without loss of generality, we assume $X_1 \in An(X_2, \mathcal{G})$. In this case, since X_2 is an ancestor of $\mathbf{Y}$, $\mathbf{S} = Pa(Cn(X_1, X_2, \mathcal{G}), \mathcal{G}) \backslash De(Cn(X_1, X_2, \mathcal{G}), \mathcal{G}) \subseteq \mathbf{X}_{sig}$ is a separation set relative to the pair (X_1, X_2), in which $Cn(X_1, X_2, \mathcal{G})$ is the set of all intermediate nodes on the directed paths from X_1 to X_2 in $\mathcal{G}$. Hence, there is no edge between X_1 and X_2 in the graph $\mathcal{G}_{\mathbf{X}_{sig}}$. □

Proof of Theorem 2. According to Proposition A1, the edge $X_1 - X_2$ in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$ may have two cases:

Case 1. $X_1 \in An(\mathbf{Y}, \mathcal{G}), X_2 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G}) \backslash An(\mathbf{Y}, \mathcal{G})$.

Case 1.1. $X_1 \notin An(X_2, \mathcal{G})$. Since X_2 is a nondescendant of X_1, $Pa(X_1, \mathcal{G}) \subseteq \mathbf{X}_{sig}$ is a separation set relative to (X_1, X_2). Hence, there is no edge between X_1 and X_2 in graph $\mathcal{G}'_{\mathbf{X}_{sig}}$, which is a contradiction.

Case 1.2. $X_1 \in An(X_2, \mathcal{G})$. In this case, there are four possible paths between X_1 and X_2 in the graph $\mathcal{G}$:

Case 1.2.1. Causal path $X_1 \to \cdots \to X_2$. In this case, given any vertex Z on this path different from X_1 and X_2, this path can be blocked, implying that X_1 and X_2 are not adjacent in $\mathcal{G}'_{\mathbf{X}_{sig}}$, which is a contradiction.

Case 1.2.2. Non-causal-path $X_1 \leftarrow \cdots * - * X_2$. Since $X_1 \in An(\mathbf{Y}, \mathcal{G})$, all parents of X_1 are in the set $\mathbf{X}_{sig}$. Hence, conditioning on the parent of X_1 on this path can block this path, implying that X_1 and X_2 are not adjacent in $\mathcal{G}'_{\mathbf{X}_{sig}}$, which is a contradiction.

Case 1.2.3. Non-causal-path $X_1 \to \cdots \leftarrow X_2$. There must exist at least one v-structure in the path, and the path can be blocked given an empty set. Suppose the nearest v-structure to X_2 is $W \to Z \leftarrow \cdots \leftarrow X_2$, then the collider Z in this v-structure can not appear on any causal paths from X_1 to X_2; otherwise, there will be a directed cycle in graph $\mathcal{G}$. Hence, X_1 and X_2 are not adjacent in $\mathcal{G}'_{\mathbf{X}_{sig}}$, which is a contradiction.

Case 1.2.4. Non-causal-path $X_1 \to \cdots \leftarrow \cdots \to X_2$. There must exist at least one v-structure in the path, and the path can be blocked given an empty set. However, different from Case 1.2.3, the colliders of the v-structures in this path may occur on some causal paths from X_1 to X_2. According to Case 1.2.1, these vertices may need to be adjusted to block the causal paths. We use Figure A1 to make an illustration in detail. In Figure A1, with out loss of generality, we suppose $Z_i \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G}) \backslash An(\mathbf{Y}, \mathcal{G}), i = 1, \ldots, p$. If not, let the point in the path that is farthest from X_1 and belongs to the set $An(\mathbf{Y}, \mathcal{G})$ be the new X_1. In this case, in the non causal path $p_0 = X_1 \to Z_1 \leftarrow U_1 \to \cdots \leftarrow U_p \to X_2$, all colliders Z_i are in the causal path $p_1 = X_1 \to Z_1 \to \cdots \to Z_p \to X_2$, and $U_i \notin \mathbf{X}_{sig}, i = 1, \ldots, p$. In order to block the causal path p_1, it is necessary to adjust some vertices in $\{Z_i, i = 1, \ldots, p\}$, say $\{Z_{k1}, \cdots, Z_{kl}\}$. But at this time, the path $p = p_1(X_1, Z_{k1}) \oplus p_0(Z_{k1}, Z_{k1+1}) \oplus p_1(Z_{k1+1}, Z_{k2}) \oplus \cdots \oplus p_0(Z_{kl}, Z_{kl+1}) \oplus p_1(Z_{kl+1}, X_2)$ cannot be blocked. In fact, even if all $Z_i, i = 1, \ldots, p$ are adjusted, the non-causal-path p_0 still cannot be blocked.

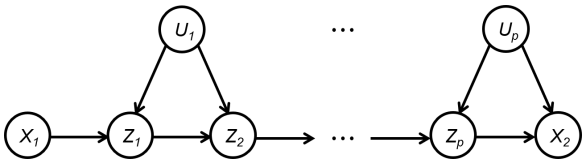

Figure A1. Illustration for Case 1.2.4.

In general, in Case 1, only when the situation in Case 1.2.4 (Figure A1) occurs, the edges in $\mathbf{E}(\mathcal{G}'_{\mathbf{X}_{sig}})\backslash\mathbf{E}(\mathcal{G}^*_{\mathbf{X}_{sig}})$ will appear. Then we have $X_1 \in Adj(Z_i, \mathcal{G}'_{\mathbf{X}_{sig}}), i = 1, \ldots, p$ and $X_1 \in Adj(X_2, \mathcal{G}'_{\mathbf{X}_{sig}})$. Note that Rule 1–Rule 3 of Meek's rules only orient the edges backward. In other words, in Case 1, no matter how the edges in $\mathbf{E}(\mathcal{G}'_{\mathbf{X}_{sig}})\backslash\mathbf{E}(\mathcal{G}^*_{\mathbf{X}_{sig}})$ are oriented, they do not affect the orientation of the edges between vertices in $An(\mathbf{Y}, \mathcal{G})$. For instance, when applying Rule 3 of Meek's rules, as shown in Figure A2, if $X_1 - X_2$ is the edge in $\mathbf{E}(\mathcal{G}'_{\mathbf{X}_{sig}})\backslash\mathbf{E}(\mathcal{G}^*_{\mathbf{X}_{sig}})$ due to Case 1.2.4, then the following hold:

- If $X_1 \in An(\mathbf{Y}, \mathcal{G}), X_2 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$, then because of the directed edge $X_2 \to X_4$, we can obtain $X_4 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$, implying that the new oriented edge $X_1 \to X_4$ is the directed edge out of the set $An(\mathbf{Y}, \mathcal{G})$, which does not affect the orientation of edges between vertices in $An(\mathbf{Y}, \mathcal{G})$.
- If $X_2 \in An(\mathbf{Y}, \mathcal{G}), X_1 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$, then because of the directed edge $X_2 \to X_4$, we have $X_4 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})$. Note that if $X_4 \in An(\mathbf{Y}, \mathcal{G})$, then we have $X_3 \in An(\mathbf{Y}, \mathcal{G})$ and $X_3 \to X_1 \in \mathbf{E}(\mathcal{G})$. Since in Case 1.2.4, only paths between X_2 and X_1 cannot be blocked, and all such paths have an arrow pointing to X_1. Hence, in the process of learning the graph $\mathcal{G}'_{\mathbf{X}_{sig}}$, we have $X_2 \perp\!\!\!\perp X_3$ and $X_2 \not\perp\!\!\!\perp X_3 \mid X_1$, implying that a v-structure $X_3 \to X_1 \leftarrow X_2$ occurs in the graph $\mathcal{G}'_{\mathbf{X}_{sig}}$ before applying Meek's rules, which is a contradiction. Hence, $X_4 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$, implying that the newly oriented edge $X_1 \to X_4$ is the directed edge between vertices in the set $De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$, which does not affect the orientation of edges between vertices in $An(\mathbf{Y}, \mathcal{G})$.

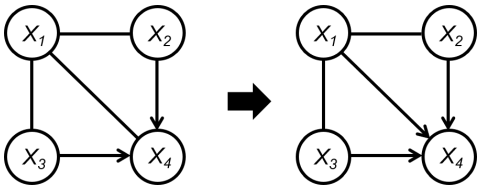

Figure A2. Rule 3 of Meek's rules as an example to illustrate **Case 1**.

Other cases of Meek's rules can be similarly proved. In fact, for Case 1 as shown in Figure A1, if there exists a vertex $W \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G}) \cap Pa(X_2, \mathcal{G})$ such that the edge between X_2 and W may be misoriented in $\mathcal{G}'_{\mathbf{X}_{sig}}$ due to the new edges in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}})\backslash\mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$, we have $W \in Adj(Z_p, \mathcal{G})$; otherwise, $W \to X_2 \leftarrow Z_p$ forms a v-structure and the edge $W \to X_2$ can be oriented correctly in both $\mathcal{G}'_{\mathbf{X}_{sig}}$ and $\mathcal{G}^*_{\mathbf{X}_{sig}}$. And then, we have $W \in Adj(Z_{p-1}, \mathcal{G})$; otherwise, $W \to Z_p$ can be oriented by v-structure and $W \to X_2$ can be oriented correctly in both $\mathcal{G}'_{\mathbf{X}_{sig}}$ and $\mathcal{G}^*_{\mathbf{X}_{sig}}$ by applying Rule 2 of Meek's rules if the edge between Z_p and X_2 is directed and using Lemma 1 in [24] if the edge between Z_p and X_2 is undirected. Similarly, we have $W \in Adj(Z_i, \mathcal{G}), i = 1, \ldots, p$ and $W \in Adj(X_1, \mathcal{G})$. Note that the vertices X_1, X_2 and W form a triangle, implying that the edge between W and X_2 cannot be oriented by applying Meek's rules to the edge between X_1 and X_2 in $\mathcal{G}'_{\mathbf{X}_{sig}}$, which contradicts the assumption.

Case 2. $X_1, X_2 \in De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})\backslash An(\mathbf{Y}, \mathcal{G})$.

Case 2.1. $X_1 \notin An(X_2, \mathcal{G})$ and $X_2 \notin An(X_1, \mathcal{G})$. In this case, there are three possible paths between X_1 and X_2 in the graph $\mathcal{G}$:

Case 2.1.1. Non-causal-path $X_1 \to \cdots \to X_2$. The specific discussion is similar to Case 1.2.3.

Case 2.1.2. Non-causal-path $X_1 \to \cdots \leftarrow \cdots \to X_2$ (or $X_1 \leftarrow \cdots \to \cdots \leftarrow X_2$). There must exist at least one v-structure in the path, and the path can be blocked given an empty set. Note that, different from Case 1.2.4, there is no causal path between X_1 and X_2 at this time, implying that the collider of the v-structure closest to X_2 (or X_1) will not be adjusted. Therefore, X_1 and X_2 are not adjacent in $\mathcal{G}'_{\mathbf{X}_{sig}}$, which is a contradiction.

Case 2.1.3. Non-causal-path $X_1 \leftarrow \cdots \to X_2$. Since some parents of X_1 or X_2 may not belong to the set $\mathbf{X}_{sig}$, this path cannot be blocked. For example, in the fork $X_1 \leftarrow Z \to X_2$, when $Z \notin \mathbf{X}_{sig}$, an edge in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$ appears between X_1 and X_2. Similar to the discussion at the end of Case 1, no matter how this edge is oriented, Meek's rules are backward-oriented, so the orientation of this edge only happens inside the set $De(An(\mathbf{Y}, \mathcal{G}), \mathcal{G})$ and does not affect the orientation within the set $An(\mathbf{Y}, \mathcal{G})$.

Case 2.2. $X_1 \in An(X_2, \mathcal{G})$ (the case of $X_2 \in An(X_1, \mathcal{G})$ can be discussed similarly). In this case, there are four possible paths between X_1 and X_2 in the graph $\mathcal{G}$:

Case 2.2.1. Causal path $X_1 \to \cdots \to X_2$. The discussion is the same as Case 1.2.1.

Case 2.2.2. Non-causal-path $X_1 \to \cdots \leftarrow X_2$ or $X_1 \leftarrow \cdots \to \cdots \leftarrow X_2$. The discussion is the same as Case 1.2.3.

Case 2.2.3. Non-causal-path $X_1 \to \cdots \leftarrow \cdots \to X_2$. The discussion is the same as Case 1.2.4.

Case 2.2.4. Non-causal-path $X_1 \leftarrow \cdots \to X_2$. The discussion is the same as Case 2.1.3.

Case 2.3. This case is symmetric to Case 2.2 and can be discussed similarly.

We already proved that the edges in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$ do not affect the skeleton or orientation of edges between ancestors of $\mathbf{Y}$ in $\mathcal{G}^*$. In fact, it is worth mentioning that, in the above proof, all discussions focus on graph $\mathcal{G}$, implying that the orientation of edges between vertices in $An(\mathbf{Y}, \mathcal{G})$ are not affected by the new edge in $\mathbf{E}_s(\mathcal{G}'_{\mathbf{X}_{sig}}) \backslash \mathbf{E}_s(\mathcal{G}^*_{\mathbf{X}_{sig}})$. In other words, the ancestors of $\mathbf{Y}$ in the two graphs are the same, while the possible ancestors of $\mathbf{Y}$ in $\mathcal{G}^*_{\mathbf{X}_{sig}}$ are also the possible ancestors of $\mathbf{Y}$ in $\mathcal{G}'_{\mathbf{X}_{sig}}$, but not vice versa. □

Proof of Theorem 3. We need to prove that for any $X \in \mathbf{X}_{sig}$, X is a direct cause of $\mathbf{Y}$ if and only if there is no subset $\mathbf{A} \subseteq \mathbf{X}_{sig} \backslash (\{X\} \cup \mathbf{N}_X)$ such that $X \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}$. According to Theorem 1, the "only if" part holds obviously.

Now, we prove the "if" part. According to Theorem 1, for any $X \in \mathbf{X}_{sig}$, X is a direct cause of $\mathbf{Y}$ if and only if there is no subset $\mathbf{A} \subseteq \mathbf{X}_{sig} \backslash \{X\}$ such that $X \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{A}$. Hence, using the definition of $\mathbf{N}_X$ in Algorithm 2, it suffices to prove that if we have $\mathbf{Y} \perp\!\!\!\perp X \mid \mathbf{A}$, then for each non-direct-cause variable $V \in \mathbf{A}$, there exists at least one separation set of V and $\mathbf{Y}$ that does not contain X. In fact, paths between V and $\mathbf{Y}$ can be divided into two types: paths that go through X and paths that do not go through X. For paths that do not go through X, the separation set naturally does not contain X. Then, for any path that goes through X, the path can be represented as $p(V, \mathbf{Y}) = p(V, X) \oplus p(X, \mathbf{Y})$, where $p(X, \mathbf{Y})$ has already been blocked by some variables in set $\mathbf{A}$. Therefore, whether the subpath $p(V, X)$ is blocked or not, the path $p(V, \mathbf{Y})$ can be separated by a set that does not contain X. □

Proof of Theorem 4. We first prove that Algorithm 3 can learn the PCD set PCD_T of a target node T correctly. Similar to the definition of $\mathbf{X}_{sig}$, let $T_{sig} \subseteq \mathbf{X}$ be the set of variables associated with T. As shown in Line 1 in Algorithm 3, the initial value PCD_T^0 of PCD_T is $\{X : X \in \mathbf{V} \backslash \mathbf{U}, \text{ and } T \not\perp\!\!\!\perp X\}$, which is a subset of T_{sig}. In fact, if the non-PC set $\mathbf{U}$ is empty, then we have $PCD_T^0 = T_{sig}$. According to the Markov condition and the faithfulness assumption, for any $X \in T_{sig}$, X is a parent or child of T if and only if there is no subset $\mathbf{A} \subseteq T_{sig} \backslash \{X\}$ such that $X \perp\!\!\!\perp T \mid \mathbf{A}$. Hence, PCD_T contains all parents and children of T. For any nondescendant variable of T, the set of T's parents separates them from T. Due to the lack of a separation set, some descendant variables of T may be included in PCD_T, as shown in Example 3. Therefore, PCD_T obtained by Algorithm 3 consists of T's parents, children, and some descendants.

Now, we prove that Algorithm 4 can learn all causes of $\mathbf{T}$ within a given distance m. First, according to the discussion following Example 3, Algorithm 4 can learn the PC set of

each variable correctly by using Algorithm 3 and symmetric validation method (Lines 7–13 in Algorithm 4). In other words, Algorithm 4 can learn the skeleton of the local structure of each variable correctly. Next, note that once the skeleton of the local structure of a variable is determined, its separation sets from other variables are also obtained at the same time (Line 6 in Algorithm 4). Therefore, all v-structures can be learned correctly because they are determined by local structures and separation sets (Line 14 in Algorithm 4). Combined with Meek's rules, Algorithm 4 learns the orientation of the local structure of each variable correctly. Finally, we show that continuing the algorithm cannot obtain more causes of **T** within a distance m. Notice that we learn the local structure of nodes layer by layer, and we only learn the next layer after all the nodes of a certain layer have been learned (Line 15 in Algorithm 4). Hence, once Algorithm 4 is stopped, it means that all directed paths pointing to **T** with a distance less than or equal to m have been found, and the m-th edge of these paths has been directed. As shown above, we can correctly obtain all edges and v-structures and their orientations. Hence, continuing the algorithm can only orient new edges that are farther away from **T** ($> m$), which is not what we care about.

We already showed that Algorithm 4 can correctly learn all causes of **T** that are within a distance of m from **T**. Note that the distance between a functional dynamic target **Y** and its direct causes is always 1. Thus, obviously, if **T** is exactly the set of **Y**'s direct causes obtained from Algorithm 2, and the node L in Line 2 in Algorithm 4 is exactly **Y**, then according to Theorem 3 and the proof above, Algorithm 4 learns all causes of **Y** within a given distance $m + 1$ correctly. □

References

1. Karkach, F. Trajectories and models of individual growth. *Demogr. Res.* **2006**, *15*, 347–400.
2. Richards, A.S. A flexible growth function for empirical use. *J. Exp. Bot.* **1959**, *10*, 290–301.
3. Zimmerman, D.L.; Núñez-Antón, V. Parametric modelling of growth curve data: An overview. *Test* **2011**, *10*, 1–73.
4. Murre, J.M.; Chessa, A.G. Power laws from individual differences in learning and forgetting: Mathematical analyses. *Psychon. Bull. Rev.* **2001**, *18*, 592–597.
5. Wixted, J.T.; Chessa, A.G. On Common Ground: Jost's (1897) law of forgetting and Ribot's (1881) law of retrograde amnesia. *Psychol. Rev.* **2004**, *111*, 864–879.
6. Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D.A.; Nolan, G.P. Causal protein-signaling networks derived from multiparameter single-cell data. *Science* **2005**, *308*, 523–529.
7. Pearl J. *Causality Models, Reasoning and Inference*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2009.
8. Han, B.; Park, M.; Chen, X.W. A Markov blanket-based method for detecting causal SNPs in GWAS. *BMC Bioinform.* **2010**, *11*, S5.
9. Duren Z.; Wang, Y. A systematic method to identify modulation of transcriptional regulation via chromatin activity reveals regulatory network during mESC differentiation. *Sci. Rep.* **2016**, *6*, 22656.
10. Heckman, J.J. Comment on "Identification of causal effects using instrumental variables". *J. Am. Stat. Assoc.* **1996**, *91*, 459–462.
11. Winship, C.; Morgan, S.L. The estimation of causal effects from observational data. *Annu. Rev. Sociol.* **1999**, *25*, 659–706.
12. Yin, J.; Zhou, Y.; Wang, C.; He, P.; Zheng, C.; Geng, Z. Partial Orientation and Local Structural Learning of Causal Networks for Prediction. In Proceedings of the Causation and Prediction Challenge at WCCI, Hong Kong, China, 1–6 June 2008; pp. 93–105.
13. Wang, C.; Zhou, Y.; Zhao, Q.; Geng, Z. Discovering and Orienting the Edges Connected to a Target Variable in a DAG via a Sequential Local Learning Approach. *Comput. Stat. Data Anal.* **2014**, *77*, 252–266.
14. Pena, J.M.; Nilsson, R.; Bjorkegren, J.; Tegner J. Towards Scalable and Data Efficient Learning of Markov Boundaries. *J. Mach. Learn. Res.* **2007**, *45*, 211–232.
15. Gao, T.; Ji, Q. Efficient Markov Blanket Discovery and Its Application. *IEEE Trans. Cybern.* **2017**, *47*, 1169–1179.
16. Wang, H.; Ling, Z.; Yu, K.; Wu, X. Towards Efficient and Effective Discovery of Markov Blankets for Feature Selection. *Inf. Sci.* **2020**, *509*, 227–242.
17. Gao, T.; Ji, Q. Local Causal Discovery of Direct Causes and Effects. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2512–2520.
18. Ling, Z.; Yu, K.; Wang, H.; Liu, L.; Li, J. Any Part of Bayesian Network Structure Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *14*, 1–14.
19. Yu, L.; Liu, H. Efficient Feature Selection via Analysis of Relevance and Redundancy. *J. Mach. Learn. Res.* **2004**, *5*, 1205–1224.
20. Pearl, J.; Shafer, G. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: San Mateo, CA, USA, 1988.
21. Verma, T.; Pearl, J. Equivalence and synthesis of causal models. In Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, USA, 27–29 July 1990; pp. 220–227.
22. Pearl, J.; Geiger, D.; Verma, T. Conditional independence and its representations. *Kybernetika* **1989**, *25*, 33–44.

23. Andersson, S.A.; Madigan, D.; Perlman, M.D. A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Stat.* **1997**, *25*, 505–541.
24. Meek, C. Causal inference and causal explanation with background knowledge. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–20 August 1995; pp. 403–410.
25. Fekedulegn, D.; Mac Siúrtáin, M.P.; Colbert, J.J. Parameter estimation of nonlinear models in forestry. *Silva Fenn.* **1999**, *33*, 327–336.
26. Gossman, M.; Koops, W. Multiple analysis of growth curves in chickens. *Poulty Sci.* **1988**, *67*, 33–42.
27. Xu, M.J.; Zhu, L.B.; Zhou, S.; Ye, C.G.; Mao, M.X.; Sun, K.; Su, L.D.; Pan, X.H.; Zhang, H.X.; Huang, S.G.; et al. A computational framework for mapping the timing of vegetative phase change. *New Phytol.* **2016**, *211*, 750–760.
28. Spirtes, P.; Glymour, C. An algorithm for fast recovery of sparse causal graphs. *Soc. Sci. Comput. Rev.* **1991**, *9*, 62–72.
29. Chickering, D.M. Optimal structure identification with greedy search. *J. Mach. Learn. Res.* **2002**, *3*, 507–554.
30. West, D. *Introduction to Graph Theory*; Prentice Hall: Upper Saddle River, NJ, USA, 1996.