The Computer Journal Advance Access published April 14, 2012

doi:10.1093/comjnl/bxs032

Exploring Causal Relationships with Streaming Features[*]

Kui Yu[1,*], Xindong Wu[2], Wei Ding[3] and Hao Wang[1]

[1]*Department of Computer Science, Hefei University of Technology, Hefei, China*
[2]*Department of Computer Science, University of Vermont, Burlington, VT, USA*
[3]*Department of Computer Science, University of Massachusetts Boston, Boston, MA, USA*
**Corresponding author: ykui713@gmail.com*

Causal discovery is highly desirable in science and technology. In this paper, we study a new research problem of discovery of causal relationships in the context of streaming features, where the features steam in one by one. With a Bayesian network to represent causal relationships, we propose a novel algorithm called causal discovery from streaming features (CDFSF) which consists of a two-phase scheme. In the first phase, CDFSF dynamically discovers causal relationships between each feature seen so far with an arriving feature, while in the second phase CDFSF removes the false positives of each arrived feature from its current set of direct causes and effects. To improve the efficiency of CDFSF, using the symmetry properties between parents (causes) and children (effects) in a faithful Bayesian network, we present a variant of CDFSF, S-CDFSF. Experimental results validate our algorithms in comparison with the existing algorithms of causal relationship discovery.

Keywords: causal discovery; streaming features; Bayesian networks

Received 14 October 2011; revised 19 January 2012
Handling editor: Franco Zambonelli

1. INTRODUCTION

Causal discovery is of fundamental and practical interest in many areas of science and technology, including medicine, biology, finance and pharmacology. The goal of causal discovery is to uncover causal relationships between features. For example, in the domain of medicine, causal discovery can determine the cause of a disease and help the disease and treatment prevention.

A causal model represents causal relationships with two components: a statistical model, and a causal graph that describes the causal relationships between features. One of the most frequently used causal models is causal Bayesian networks introduced by Pearl [1, 2]. A causal Bayesian network is a Bayesian network in which each edge is described as a direct causal influence between a parent node (feature) and a child node, with regard to the other nodes in the network. Structure learning of causal Bayesian networks in the observational data are essentially the same as structure learning of Bayesian networks, thus, one of the most exciting prospects in the last two decades has been the possibility of using Bayesian networks to discover causal relationships among features in the observed data [3–5], stemming from the seminal work of Pearl [6].

Bayesian network structure learning methods can be classified as global or local learning approaches. Global learning attempts to learn a unified Bayesian network over all the features, but it can only deal with no more than 300 features [7–13]. With emerging datasets containing tens or hundreds of thousands of features, the global learning approach does not reliably scale up to thousands of variables in a reasonable time [14]. Thus, the local learning approach aims to learn a local causal structure around a target feature of interest, for example, the identification of features that are direct causes and direct effects of a target node of interest, or the discovery of the Markov blanket (parents, children and parents of the children of a node in a Bayesian network) of the target [15–18]. When we identify direct causes and direct effects, or the Markov blanket (MB for short) for all features using a local discovery algorithm, we can get the skeleton (i.e. the edges without their orientations) of a Bayesian network. With this skeleton, we can develop efficient

[*]A shorter, preliminary version of this paper with the title “Causal Discovery from Streaming Features” was published in the Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), pp. 1163-1168. The new content added here, compared with the conference version of this paper, includes Sections 2, 4.3 and 4.5, Figures 1, 4–6, Tables 2–4, and major changes in Sections 1, 2 and 3.

local-to-global algorithms to orient the skeleton for getting local causal structures on the subsets of model features or the global causal structures of all features in the data [14–16].

The current state-of-the-art algorithms have significantly advanced the techniques of learning Bayesian networks, but they need all features available before learning starts. However, in many real-world applications, we cannot get/know all features initially. For example, in spam filtering, the feature space could change over time, since new words (features) may appear which are not within the original feature vectors. When new words come, they must be integrated into the current model as new unsolicited commercials come into vogue [19]. Another example is with personalized news filtering. Since the user interests may change over time, new words that can discriminate users' new interests also need to be involved in the current feature vector [20]. Under such circumstances, the feature space is changing over time, and then we cannot get all features before learning takes place, that is, not all features are available in advance.

An intriguing question from the above observations is when the full feature space is unavailable before learning begins, how can we uncover causal relationships between features?

Recently, the concept of streaming features has been proposed to interpret the situation where not all features can present initially [21–23]. Unlike a data stream, with streaming features, feature dimensions are modeled as a feature stream, and features stream in one by one and each feature is processed upon its arrival. With streaming features, it is natural for us to convert our research problem above into learning a causal model in streaming features. Here we will focus on the local learning approach with streaming features. As features stream in one by one, we can design efficient local learning algorithms to dynamically discover direct causes and direct effects, or the Markov blankets for the features seen so far without a full feature space in advance. Meanwhile, integrating streaming features into local learning, the major challenges are 2-fold.

First, aggregating all features to discover causal relationships is practically infeasible with streaming features, since in the context of streaming features, features stream in one by one instead of having all features in advance. We have to devise new causal learning methods for the changing size of feature volumes over time.

Secondly, causal relationships among features seen so far should be all achieved as if no more available features arrive. The current state-of-the-art local learning algorithms, like HITON_PC or MMPC (as described in detail in Section 5), are limited to learn both the direct causes and direct effects only for one target of interest at each run. If we get the direct causes and direct effects for all features, we need to run the HITON_PC or MMPC algorithms for each feature independently. When the features steam in one by one over time, those existing local learning algorithms cannot deal with streaming features since (i) not all features are available in advance and (ii) mining causal relationships for all available features must be done before a new feature arrives or no available features are still arriving. We have to provide effective ongoing mechanisms to process the new features as they arrive.

To tackle the above challenges, with a Bayesian network used as the language to represent causal relationships between features, in this paper, we present a new algorithm, called causal discovery from streaming features (CDFSF) to discover the direct causal relationships between features with streaming features. In order to further improve its efficiency, we use the symmetry properties between parents and children in a faithful Bayesian network to present a Symmetrical CDFSF (S-CDFSF) algorithm.

As it is hard to statistically distinguish between direct causes (parents) and direct effects (children) of a target feature T, in this work, like HITON_PC and MMPC, we identify only the feature sets of the direct causes and direct effects without distinguishing between the two when features stream in.

CDFSF is designed with a two-phase scheme. With the features steaming in one by one, in the first phase, CDFSF dynamically discovers causal relationships between each feature seen so far with an arriving feature. In this phase, when a new feature arrives, CDFSF takes each feature arrived so far as a target T and assesses whether the new one belongs to its candidate parents and children (CPC(T) for short). If so, CDFSF regards the new one as a target T and discovers its CPC(T) from the features arrived already. In the second phase, CDFSF removes the false positives of each arrived feature from its current CPC(T) set. When no more available features arrive, the sets of CPC(T) for all features arrived so far, not just for a special one of interest, are achieved. In order to improve the efficiency of CDFSF, we plug in the symmetry check and present an S-CDFSF algorithm.

State-of-the-art algorithms for the discovery of MB(T) consist of two steps: identification of parents and children of the target (PC(T) for short), and then discovery of the candidate spouses of the target from the candidate parents of each feature within PC(T). The key difference between our algorithms and the existing algorithms is the identification of PC(T).

The rest of the paper is organized as follows: Background work is given in Section 2. Section 3 discusses the proposed algorithms and Section 4 presents the experimental results. Related work is reviewed in Section 5, and finally Section 6 concludes the paper with some discussions on future work.

2. BACKGROUND

In this paper, we denote variables in uppercase letters (X; Y), and states or values of these variables by the same lowercase letters (x; y). We deal with discrete probability distributions and complete datasets only and the words 'variable', 'node' and 'feature' are all used interchangeably in the rest of this paper.

Definition 2.1 (Conditional Independence) [4]. *In a variable set V, two variables $X \in V$ and $Y \in V$ are conditionally*

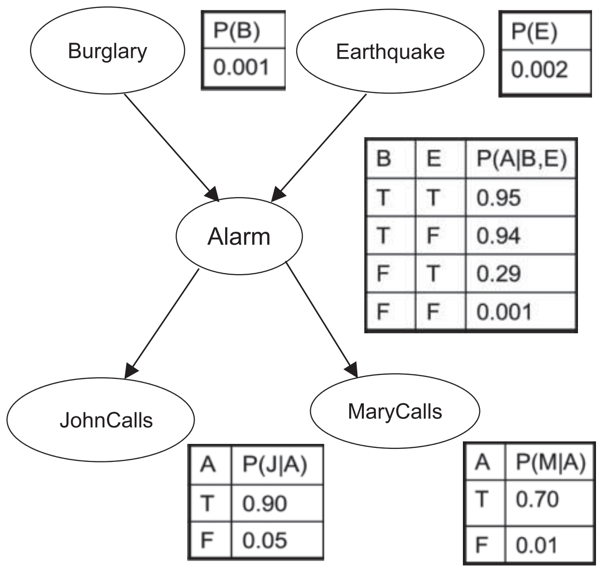

FIGURE 1. A simple example of a Bayesian network [6].

independent given a set of variables $Z \subseteq V$ with respect to a probability distribution P, iff $P(X|Y, Z) = P(X|Z)$.

For notational convenience, the conditional independence is denoted as $\mathrm{Ind}(X,Y|Z)$ and conditional dependence as $\mathrm{Dep}(X,Y|Z)$.

DEFINITION 2.2 (BAYESIAN NETWORK) [6]. *Let P be a discrete joint probability distribution of a set of random variables V via a directed acyclic graph G. We call the triplet $< V, G, P >$ a (discrete) Bayesian network if $< V, G, P >$ satisfies the Markov condition: every variable is independent of any subset of its non-descendant variables conditioned on its parents.*

A simple Bayesian network is shown in Fig. 1, which encodes the joint probability P over a set of variables $V = \{X_1, X_2, \ldots, X_n\}$ and decomposes it into a product of the conditional probability distributions over each variable given its parents in the graph. Assuming $\mathrm{Pa}(X_i)$ is the set of parents of X_i, the joint probability P is written as:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathrm{Pa}(X_i)). \quad (2.1)$$

In this network, the set V contains five variables in the directed acyclic graph G and the conditional probability table P for each variable is given in Fig. 1.

DEFINITION 2.3 (FAITHFULNESS) [3]. *A Bayesian network satisfies the faithfulness condition if and only if every conditional independence entailed by the directed acyclic graph G is also present in P.*

The directed acyclic graph of a network in conjunction with the Markov condition directly encodes some of the independencies of the probability distribution and entails others. A graphical criterion called d-separation captures exactly all the conditional independence relationships that are implied by the Markov condition [6].

DEFINITION 2.4 (d-SEPARATION) [6]. *A collider on a path p is a node with two incoming edges that belong to p. A path between X and Y given a conditioning set Z is open, if (a) every collider of p is in Z or has a descendant in Z and (b) no other nodes on p are in Z. If a path is not open, then it is blocked. Two nodes X and Y are d-separated (denoted as* $\mathrm{Dsep}(X, Y|Z)$*) given a conditioning set Z in a Bayesian network if and only if every path between X and Y is blocked by Z.*

The ‘d’ in d-separation stands for dependence. If two variables are d-separated relative to a set of variables Z in a Bayesian network, then they are independent conditional on Z in all probability distributions which this Bayesian network can represent.

DEFINITION 2.5. *A causal Bayesian network $< V, G, P >$ is a Bayesian network with the additional semantics that $X \in V$ and $Y \in V$ if a node X is a parent of a node Y in G, then X directly causes Y.*

DEFINITION 2.6 (CAUSAL MARKOV CONDITION) [3]. *In a causal Bayesian network, if every node is independent of its non-effects given its direct causes, then the causal Markov condition holds.*

The causal Markov condition permits the joint distribution of the variables in a causal Bayesian network to be factored as in Equation (2.1).

DEFINITION 2.7 (CAUSAL FAITHFULNESS) [3]. *A causal Bayesian network satisfies the faithfulness condition if it satisfies the faithfulness condition of Definition 2.3.*

THEOREM 2.1 [4]. *In a faithful Bayesian network, the following term holds:*

$$\mathrm{Dsep}(X, Y|Z) \Leftrightarrow \mathrm{Ind}(X, Y|Z).$$

Theorem 2.1 proves that d-separation can be replaced with conditional independence tests to compute all the conditional independence relationships in a faithful Bayesian network.

With the causal Markov condition and causal faithfulness assumptions made on structure learning of Bayesian networks, structure learning of causal Bayesian networks in the observational data is essentially the same as structure learning of Bayesian network [3, 4]. Therefore, the task of causal discovery can be converted into the task of learning Bayesian network structure.

3. PROPOSED ALGORITHMS

3.1. The CDSFS algorithm

With a Bayesian network used as the language to represent data-generating processes and causal relationships, we design and implement the CDFSF algorithm to address the challenges on discovery of direct causal relationships with streaming features as defined in Definition 3.1.

Definition 3.1 (Streaming Features). *Streaming features involve a feature vector that streams in one by one over time while the number of training examples is fixed.*

One key contribution of CDFSF is on efficiently discovering causal relationships between each feature seen so far. In a faithful Bayesian network, direct causes and direct effects of feature X are its parents and children, respectively. Thus, Theorem 3.1 proves that finding a direct cause or a direct effect of feature X can be reduced to determine whether or not an edge exists between a feature and feature X in a faithful Bayesian network.

Theorem 3.1 [3, 4]. *In a faithful Bayesian network, there is an edge between the pair of nodes $X \in V$ and $Y \in V$ iff $\forall S \subseteq V \backslash \{X, Y\}$, $s.t.$ $\mathrm{Dep}(X, Y|S)$.*

From Theorem 3.1, when a new feature X arrives, we obtain an immediate method to determine causal relationships between X and a target T: for any feature $X \in V\backslash\{T\}$, find a subset $Z \subseteq V\backslash\{X, T\}$ and test whether $\mathrm{Ind}(X,T|Z)$. If the term $\mathrm{Ind}(X,T|Z)$ holds, then $X \notin \mathrm{PC}(T)$, otherwise $X \in \mathrm{PC}(T)$. The major drawback here is that the variables within V are not all available before learning with streaming features. This method is also very inefficient for testing all subsets of the set V excluding X and T. To improve its efficiency, Aliferis *et al.* [15] proved the following Corollary 3.1, which shows that to test whether $\mathrm{Ind}(X,T|Z)$, it is sufficient to test all subsets $Z \subseteq \mathrm{PC}(X)\backslash T$ and all $Z \subseteq \mathrm{PC}(T)\backslash X$.

Corollary 3.1. *In a faithful Bayesian network, there is an edge between the pair of nodes $X \in V$ and $Y \in V$ iff $\mathrm{Dep}(X, Y|S)$, for all $S \subseteq PC(X)\backslash\{Y\}$ and $S \subseteq PC(Y)\backslash\{X\}$.*

The problem with Corollary 3.1 is that, in real-world applications, we cannot get true $\mathrm{PC}(X)$ and $\mathrm{PC}(T)$. Thus, Corollary 3.2 shows that we can work on a superset of true $\mathrm{PC}(X)$ or $\mathrm{PC}(T)$ [15].

Corollary 3.2. *In a faithful Bayesian network, there is an edge between the pair of nodes $X \in V$ and $Y \in V$ iff $\mathrm{Dep}(X, Y|Z)$, for all $Z \subseteq S$ where $PC(X)\backslash Y \subseteq S \subseteq V\backslash\{X, Y\}$.*

With Corollary 3.2, when the features steam in one by one, we can work on a superset of true $\mathrm{PC}(X)$ or $\mathrm{PC}(T)$ which can start from an empty set or a specified set with some domain knowledge. More specially, we set the candidate PC set of a new feature as an empty set and gradually build the PC sets for arrived features as time goes on. From Theorem 3.1, if $X \in \mathrm{PC}(T)$, $\mathrm{Dep}(X,T|Z)$ for any Z exists. Therefore, we can work on the current PC set of a target T to online determine the causal relationships between a new arriving feature X and T.

The other key contribution of the CDFSF algorithm is on providing effective ongoing mechanisms in dealing with increasing feature volumes. CDFSF adopts an online mechanism with a two-phase scheme: discovering causal relationships and removing false positives. In the first (discovering causal relations) phase, when a new feature X arrives, CDFSF first takes each feature seen so far as a target T and assesses whether X belongs to its $\mathrm{CPC}(T)$. Next, CDFSF regards X as the target, and then discovers its $\mathrm{CPC}(X)$ from the features arrived so far in turn. In the second (removing false positives) phase, CDFSF re-evaluates each feature within the sets of CPC for all features seen so far and eliminates false positives from them. When the process of generating features is over, the sets CPC for all features seen so far are returned.

Algorithm 1 The pseudo-code of CDFSF.

```
Input: Streaming features X
Output: CPC(X), X = {X_1, X_2, ..., X_n}

Initialization
 GFS={},CPC(X)={};
Repeat
/* The discovering phase for causal relations*/
  generate a new feature $X_i$ ;
  $GFS=GFS \cup X_i$;
   for each feature $T \in GFS\backslash X_i$
       if $\forall S \subseteq CPC(T)$ s.t $\mathrm{Dep}(X_i, T|S)$
              $CPC(T)= CPC(T) \cup X_i$;
      end
       if $\forall Z \subseteq CPC(X_i)$ s.t $\mathrm{Dep}(T, X_i|Z)$
              $CPC(X_i)= CPC(X_i) \cup T$;
      end
/*The removing phase for false positives*/
     for each feature $Y \in CPC(T), T \in GFS$
        if $\exists S \subseteq CPC(T)\backslash Y$ s.t $\mathrm{Ind}(Y, T|S)$
              $CPC(T)= CPC(T)-Y$;
        end
     end
   end
until no available features to arrive or a given threshold is satisfied.
```

The pseudo-code of CDFSF is shown in Algorithm 1. The set GFS stores the features arrived so far, $\mathrm{CPC}(X_i)$ denotes the set

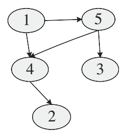

FIGURE 2. An illustrating Bayesian network to demonstrate the trace of the CDFSF algorithm.

of candidate parents and children of X_i, and conditional independence tests use G^2 test which is a modified version of χ^2.

As an illustration, we assume the Bayesian network in Fig. 2 is faithful and so the conditional dependencies and independences can be read off the graph directly using the d-separation criterion. Now we assume that the set $V = \{1, 2, 3, 4, 5\}$ and the set ordering $= \{5, 1, 4, 2, 3\}$ which denotes the ordering of arriving features. We initialize GFS $= \{\}$ and CPC$(X) = \{\}$. A trace of CDFSF is as follows.

(i) First, feature 5 arrives. The set GFS is empty excluding feature 5 and so CPC(5) $= \{\}$.

(ii) Feature 1 arrives where GFS $= \{5, 1\}$, CPC(5) $= \{\}$ and CPC(1) $= \{\}$. In the first phase, feature 5 inside GFS is first regarded as a target, and then our approach determines whether $1 \in$ CPC(5). Since there are all subsets within CPC(5) that make feature 1 conditionally dependent of feature 5, Dep(1,5$|\phi$), we obtain $1 \in$ CPC(5). Next, feature 1 is regarded as a target and our method tests whether $5 \in$ CPC(1). In a similar way, $1 \in$ CPC(5). Since there are new features being added into CPC(5) and CPC(1), the second phase is performed. Finally, CPC(5) $= \{1\}$ and CPC(1) $= \{5\}$.

(iii) Feature 4 arrives where GFS $= \{5, 1, 4\}$, CPC(5) $= \{1\}$ and CPC(1) $= \{5\}$. In the first phase, feature 5 inside GFS is first regarded as a target, then our method tests whether the new feature $4 \in$ CPC(5). Since all subsets of CPC(5) that make 4 conditionally dependent of 5, Dep(4,5$|\phi$) and Dep(4,5$|$1), feature 4 enters CPC(5), and then CPC(5) $= \{1, 4\}$. Then the second phase re-evaluates each feature within CPC(5) to remove false positives. Our method regards feature 1 as a target. As for testing whether feature $4 \in$ CPC(1), in a similar way, we can obtain $4 \in$ CPC(1). Finally, our method keeps the new arriving feature 4 as a target and discovers CPC(4) from the set $\{5, 1\}$. When the phase is over, CPC(5) $= \{1, 4\}$, CPC(1) $= \{5, 4\}$ and CPC(4) $= \{5, 1\}$.

(iv) Feature 2 arrives where GFS $= \{5, 1, 4, 2\}$, CPC(5) $= \{1, 4\}$, CPC(1) $= \{5, 4\}$ and CPC(4) $= \{5, 1\}$. In the first phase, feature 5 inside GFS is first regarded as a target, then our method tests whether the new feature $2 \in$ CPC(5). Since there is a subset within CPC(5) to make Ind(5,2$|$1, 4) exist, we obtain $2 \notin$ CPC(5). Now, feature 1 is considered as a target. Since there is a subset $\{4\}$ within CPC(1) to make $2 \notin$ CPC(1). In a similar way, we obtain $2 \in$ CPC(4). In turn, our method keeps the new arriving feature 2 as a target and discovers CPC(2) from the set $\{5, 1, 4\}$. And then we obtain CPC(2) $= \{5, 1, 4\}$. During the second phase, we cannot remove any features from CPC(5), CPC(1) and CPC(4). We remove features 5 and 1 from CPC(2), since we can find that the subset $\{4\}$ within CPC(2) makes the terms Ind(2,5$|$4) and Ind(2,1$|$4) exist. Finally, CPC(5) $= \{1, 4\}$, CPC(1) $= \{5, 4\}$ and CPC(2) $= \{4\}$, CPC(4) $= \{5, 1, 2\}$.

In a similar way after the arrival of feature 3, the final outputs are CPC(5) $= \{1, 4, 3\}$, CPC(1) $= \{5, 4\}$, CPC(4) $= \{5, 1, 2\}$, CPC(2) $= \{4\}$ and CPC(3) $= \{5\}$.

From the above illustration, we can see that when a new feature arrives, CDFSF needs to perform two conditional independence tests between the new one and each feature inside GFS in the first phase. This is very time-consuming when the feature space is large. In order to improve the efficiency of CDFSF, we propose an S-CDFSF algorithm by plugging the symmetry check into CDFSF.

3.2. The S-CDFSF algorithm

With Theorem 3.1, the following two properties can easily be obtained.

PROPERTY 1. In a faithful Bayesian network, if $X \in \mathrm{PC}(Y)$, then $Y \in \mathrm{PC}(X)$.

PROPERTY 2. In a faithful Bayesian network, if $X \notin \mathrm{PC}(Y)$, then $Y \notin \mathrm{PC}(X)$.

These properties show the symmetrical relationships between parents and children in a faithful Bayesian network. With these two properties, the proposed S-CDFSF algorithm uses two strategies to improve its efficiency. One strategy is to use Property 1 in the causal relationships discovering phase. In this phase, S-CDFSF only regards each feature inside GFS as a target T to test whether the new feature X belongs to CPC(T) while the set CPC(X) is determined according to Property 1. For example, assuming $Y \in$ GFS, when a new feature X arrives and $X \in$ CPC(Y), we directly get $Y \in$ CPC(X) and do not need to regard X as a target feature again.

The other strategy is to use Property 2 in the false-positive removing phase. In a similar way, if X is removed from CPC(T), then T should also be removed from CPC(X) if T is inside CPC(X) according to Property 2. The pseudo-code of S-CDFSF is shown in Algorithm 2.

S-CDFSF performs as follows. In the first phase, when a new feature X_i arrives, S-CDFSF selects the first feature inside GFS as a target T, then tests whether X_i belongs to its CPC(T). If so, X_i is added to CPC(T), otherwise it is discarded. If X_i is added, by Property 1, S-CDFSF also adds T into CPC(X_i).

If X_i is put into CPC(T), the second phase is performed. In this phase, S-CDFSF re-evaluates each feature within CPC(T) and eliminates false positives from CPC(T). For example, if $Y \notin$ CPC(T), S-CDFSF removes Y from CPC(T). According to Property 2, T should be removed from CPC(Y) without performing any tests on whether T is inside CPC(Y). Then S-CDFSF considers a next feature within GFS as a target feature until the last feature inside GFS is visited.

Algorithm 2 The pseudo-code of S-CDFSF.

```
Input: Streaming features X
Output: CPC(X), $X = \{X_1, X_2, \ldots, X_n\}$
Initialization
  $GFS = \{\}, CPC(X) = \{\}$;
repeat
/* The discovering phase for causal relations*/
    generate a new feature $X_i$;
    for each feature $T \in GFS$
            if $\forall S \subseteq CPC(T)$ s.t. Dep($X_i, T|S$)
                    $CPC(T) = CPC(T) \cup X_i$;
                    $CPC(X_i) = CPC(X_i) \cup T$;
            end
/*The removing phase for false positives*/
        for each feature $Y \in CPC(T), T \in GFS$
                if $\exists S \subseteq CPC(T)\backslash Y$ s.t Ind($Y, T|S$)
                    $CPC(T) = CPC(T) - Y$;
                    if T inside CPC(Y)
                        $CPC(Y) = CPC(Y) - T$;
                    end
                end
        end
    end
  $GFS = GFS \cup X_i$;
until no available features to arrive or a given threshold is satisfied.
```

With the same conditions as in the trace of CDFSF, a trace of S-CDFSF is as follows using the Bayesian network example in Fig. 2.

(i) First, feature 5 arrives. The set GFS is empty excluding feature 5 and so CPC(5) = {}.

(ii) Feature 1 arrives where $GFS = \{5, 1\}$, CPC(5) = {} and CPC(1) = {}. In the first phase, feature 5 inside GFS is first regarded as a target, and then we get $1 \in$ CPC(5). According to Property 1, feature 5 is added to CPC(1). After the second phase is performed, CPC(5) = {1}, and CPC(1) = {5}.

(iii) Feature 4 arrives where $GFS = \{5, 1, 4\}$, CPC(5) = {1} and CPC(1) = {5}. In the first phase, feature 5 inside GFS is first regarded as a target, feature 4 enters CPC(5), and then CPC(5) = {1, 4}. According to Property 1, feature 5 is added to CPC(4). Then the second phase re-evaluates each feature within CPC(5) to remove false positives. Next, feature 1 is regarded as a target. We can get $4 \in$ CPC(1). At the same time, we also get $1 \in$ CPC(4). Finally, CPC(5) = {1, 4}, CPC(1) = {5, 4} and CPC(4) = {5, 1}.

(iv) Feature 2 arrives where $GFS = \{5, 1, 4, 2\}$, CPC(5) = {1, 4}, CPC(1) = {5, 4} and CPC(4) = {5, 1}. In the first phase, we obtain $2 \notin$ CPC(5) and $2 \notin$ CPC(1). In a similar way, we obtain $2 \in$ CPC(4). Thus, we obtain CPC(2) = {4}. After the second phase, CPC(5) = {1, 4}, CPC(1) = {5, 4} and CPC(2) = {4}, CPC(4) = {5, 1, 2}.

After the arrival of features 3, the final outputs are the same as those of CDFSF.

3.3. Time complexity analysis

The complexity of the algorithms CDFSF and S-CDFSF depends on the time for the independence tests. For the G^2 test of independence for discrete variables, for example, we use, in our experiments in Section 4, an implementation linear to the sample size and exponential to the number of variables in the conditional set. However, because the latter number is small in practice, tests are relatively efficient.

Since CDFSF interleaves the two phases, the number of tests consists of two parts: the number of tests of the discovering phase and that of the removing phase. Assuming N features arrive at time t, then the number of conditional independence tests of CDFSF is approximately $O(N^2(|\mathrm{CPC}|k^{|\mathrm{CPC}|} + N^2k^{|\mathrm{CPC}|}))$. Since we plug the symmetry check into S-CDFSF, in the discovering phase S-CDFSF does not discover CPC of a new feature from the features within GFS. At the mean time, in the removing phase, S-CDFSF does not need check false positives in CPC of a new feature. Thus, the time complexity of S-CDFSF is approximately $O(N^2|\mathrm{CPC}|k^{|\mathrm{CPC}|})$, where k is the maximum allowable size that a conditioning set may grow and |CPC| is the largest size of the set CPC(T) over all arrived features. Thus, S-CDFSF is more efficient than CDFSF.

4. EXPERIMENTAL RESULTS

4.1. Experimental setup

To simulate the scenario of streaming features, we apply our algorithms to the datasets used in the traditional settings, that is, those of fixed features, but the features steam in one by one over time. Since there is no related work about exploring causal relationships in the context of streaming features, in order to validate our algorithms, we compare them with the state-of-the-art local learning algorithms: HITON_PC and MMPC (as described in detail in Section 5) which are applied to standard

TABLE 1. Summary of selected networks.

Bayesian network	Number of variables
Child	20
Alarm	37
Insurance10	270
Child10	200
Gene	801

settings where all features are available before learning (the software is available on the Web [24]).

We evaluate the four algorithms using data sampled from probability distributions of Bayesian networks used in real decision support systems that capture a wide variety of real-life applications (medicine, gene, device troubleshooting and so on): Child [25], Insurance [26] and Alarm [27]. The networks insurance10 and child10 were created by tiling 10 copies of the real Bayesian networks, the insurance network and the child network, respectively, in a way that retains their structural and probabilistic properties, hoping that the simulated networks will exhibit the same characteristics as the real Bayesian network tiles [28]. Gene was constructed by Tsamardinos *et al.* [14] on gene expression microarray data [29] with the same procedure as in Friedman *et al.* [30]. Since the true structure of each network is known, there exists a rigorous gold standard for assessing the performance of each algorithm, and then five benchmark Bayesian networks are selected as shown in Table 1.

The experiments were conducted on a computer with Windows XP, 2.6 GHz CPU and 2 GB memory. We evaluate the algorithms using the following metrics. Since the HITON_PC algorithm gave extremely similar results as the MMPC algorithm, we adopt the MMPC algorithm in our experiments.

(i) Precision, the number of true positives (real parents or children in a real Bayesian network) in the output divided by the number of features in the output.
(ii) Recall, the number of true positives in the output divided by the number of true positives in a test Bayesian network.
(iii) Distance, combining precision and recall to measure the Euclidean distance from the perfect precision and recall.

$$\text{distance} = \sqrt{(1-\text{precision})^2 + (1-\text{recall})^2}.$$

(iv) Efficiency: we employ two metrics, the execution time and the total number of conditional independence tests.
(v) Asymptotic behavior: we provide a preliminary study on the asymptotic behavior of both CDFSF and S-CDFSF as the number of features increases. The purpose of this study may help us to decide whether to stop an algorithm at some point (e.g. when no more features is available) or to continue until the last feature is added.

The conditional independence tests in our implementation are G^2 tests and the parameter α (alpha) is a statistical significance level for the independence tests which equals to 0.01 and 0.05 in our experiments.

4.2. Comparison of the metrics of precision, recall and distance

We run the algorithms, including CDFSF, S-CDFSF and MMPC, with each feature in each Bayesian network as a target of interest, and then report the average precision, recall and distance over all features for each Bayesian network.

With the value of α up to 0.01, Fig. 3 reports the experimental results by the three algorithms on five networks with different sample sizes. From Fig. 3, we can draw the following conclusions:

(i) CDFSF vs. S-CDFSF. On small networks, like child, alarm, child10 and insurance10, S-CDFSF is highly competitive with CDFSF. On a large network, like gene, with a small sample size, the performance of CDFSF is superior to S-CDFSF. The explanation is that when the size of samples is smaller than the size of dimensions, some conditional independence tests could be unreliable. Thus, the principle of symmetry (Property 1 and Property 2) used in the S-CDFSF algorithm could fail. But when the number of samples is much larger than the number of dimensions, S-CDFSF is highly competitive with CDFSF, even superior to CDFSF on the alarm network. Assuming that all conditional independence tests are reliable, the performance of S-CDFSF might be very competitive with CDFSF. For example, when the sample size is up to 2000, the performance of S-CDFSF is almost the same as CDFSF.
(ii) Our algorithms vs. MMPC. On the precision metric, our algorithms achieve a higher precision with various sample sizes on most networks. Our algorithms usually return fewer false positives. On the recall metric, our algorithms are a little lower than the MMPC algorithm, especially when the number of samples is smaller than the number of dimensions. The explanation is that with streaming features, without the global information of all features, our algorithms cannot discover a best candidate for each target of interest from all features at each time like MMPC, but they can find a 'best so far' candidate for each target of interest. But when the sample size is large, our algorithms are competitive with MMPC. Combining the precision and recall metrics, our algorithms even achieve a shorter distance than the MMPC algorithm on some networks.

Figure 4 reports the experimental results when the value of α is up to 0.05. The observations are as follows:

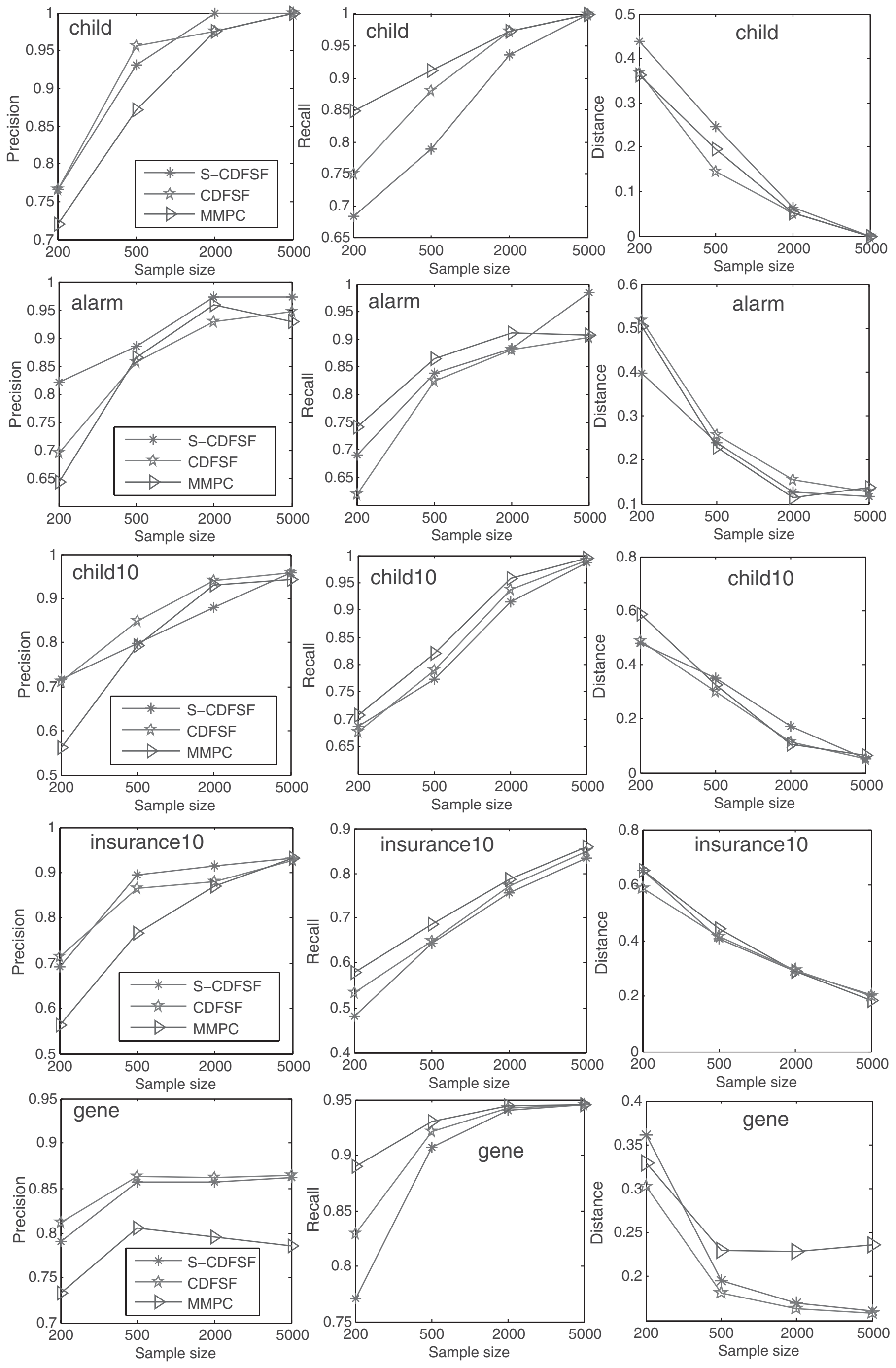

FIGURE 3. Experimental results for three algorithms with $\alpha = 0.01$.

(i) CDFSF vs. S-CDFSF. From Fig. 4, we can see that the performance of our two algorithms is similar with α up to 0.01. S-CDFSF is highly competitive with CDFSF.

(ii) Our algorithms vs. MMPC. On the precision metric, our algorithms achieve a higher precision with various sample sizes on all networks. On the recall metric, our algorithms are still lower than the MMPC algorithm when the sample size is up to 200 or 500. But when the sample size is up to 2000, our algorithms are highly competitive with MMPC on all networks. Combining the precision and recall metrics, our algorithms get a shorter distance than the MMPC algorithm on all networks.

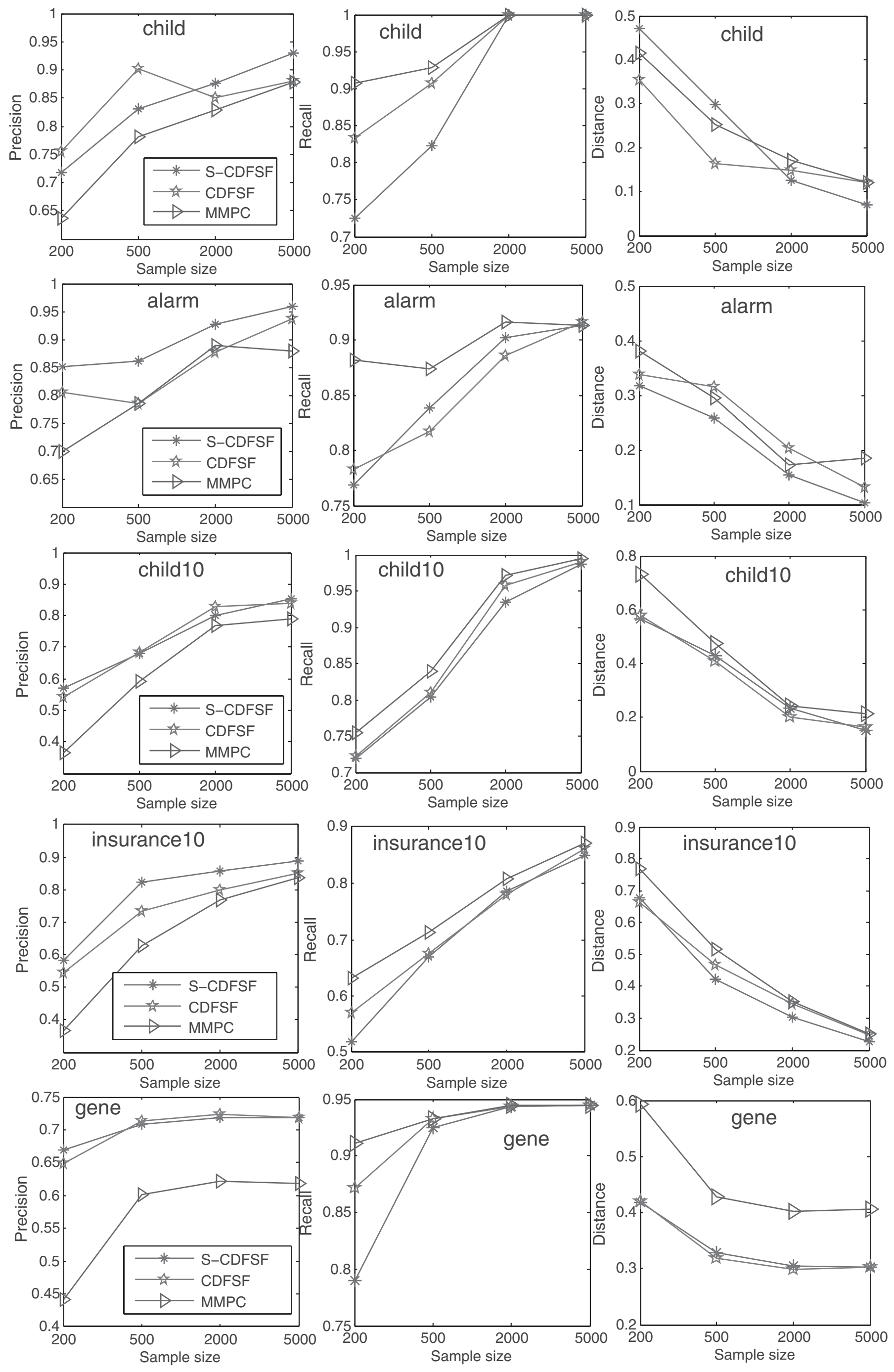

FIGURE 4. Experimental results for three algorithms with $\alpha = 0.05$.

Table 2 summarizes the average precisions, recalls and distances of S-CDFSF, CDFSF and MMPC on all five networks with 200, 500, 2000 and 5000 samples, respectively, under the value of α up to 0.01. In the table, $A/B/C$ denotes the results of S-CDFSF, CDFSF and MMPC, respectively.

From Table 2, we can see that the average precisions of our two algorithms on all five networks are higher than those of MMPC while MMPC gets higher average recalls than our two algorithms. Therefore, those three algorithms all get competitive average distances on five networks.

TABLE 2. Average precisions, recalls and distances of three algorithms on five networks ($\alpha = 0.01$).

Number of samples	Precision	Recall	Distance
200	0.7373/0.7436/0.6351	0.6271/0.6606/0.7259	0.5067/0.4696/0.5098
500	0.8500/0.8729/0.8126	0.7459/0.7794/0.8033	0.3387/0.2909/0.3214
2000	0.9123/0.9098/0.9131	0.8520/0.8657/0.8961	0.2034/0.1903/0.1686
5000	0.9400/0.9360/0.9212	0.9167/0.9110/0.9203	0.1392/0.1315/0.1429

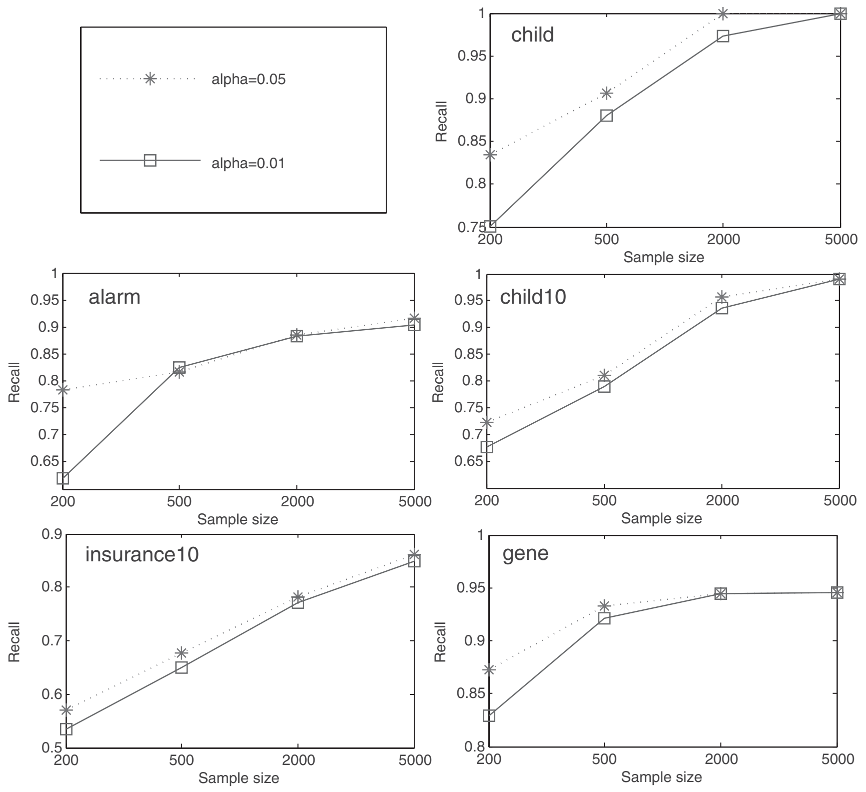

FIGURE 5. The performance of CDFSF with different values of α.

4.3. An analysis of CDSFS algorithms with different values of α

We also have an analysis of the influence of different values of α on the recall metric, as shown in Fig. 5. Our experimental results reveal that with a small sample size, the performance of CDSFS is improved a little when the value of α is up to 0.01. With a large sample size, the performance of CDSFS is little sensitive to the value of α. Since the S-CDSFS algorithm gave extremely similar performances as the CDFSF algorithm, we do not report the results of S-CDFSF with different values of α on the recall metric.

4.4. Comparison of running time

In this section, we employ two metrics to compare our algorithms in terms of the running time. The first metric indicating computational efficiency is the total number of conditional independence tests, while the second metric is the execution time.

Since the MMPC algorithm was implemented in the traditional scenario in Matlab codes while our algorithms were performed in the context of steaming features in C codes, a time–performance comparison between them was not conducted. However, we still give a comparison of the time complexity of MMPC as follows. In the worst case, MMPC will calculate the association of every variable with the target conditioned on all subsets of CPC. Thus, the total number of tests is bounded by $O(N^2 2^{|\mathrm{CPC}|})$ for all features while instead of conditioning on all subsets of the CPC, the number of tests is bound by $\mathrm{O}(N^2|\mathrm{CPC}|^{k+1})$ conditioning on all subsets of sizes up to k [14]. The time complexity of S-CDFSF is approximately $O(N^2|\mathrm{CPC}|k^{|\mathrm{CPC}|})$ where |CPC| is always very

TABLE 3. Normalized numbers of conditional independence tests with $\alpha = 0.01$.

	Sample size			
Networks	200	500	2000	5000
Child	0.57	0.61	0.72	0.70
Alarm	0.49	0.45	0.49	0.47
Child10	0.54	0.53	0.54	0.58
Insurance10	0.52	0.52	0.53	0.55
Gene	0.51	0.50	0.48	0.49

TABLE 4. Normalized numbers of conditional independence tests with $\alpha = 0.05$.

	Sample size			
Networks	200	500	2000	5000
Child	0.54	0.59	0.70	0.71
Alarm	0.45	0.47	0.47	0.47
Child10	0.55	0.56	0.53	0.56
Insurance10	0.50	0.50	0.51	0.51
Gene	0.51	0.49	0.48	0.49

TABLE 5. Running time (seconds) of S-CDFSF vs. CDFSF with $\alpha = 0.01$.

	Sample size			
Networks	200	500	2000	5000
Child	0/0	0/0	0/0	0/1
Alarm	0/0	0/0	0/1	3/7
Insurance10	1/2	3/4	11/21	61/113
Child10	0/1	1/2	8/13	51/83
Gene	10/19	25/46	127/268	1613/3293

TABLE 6. Running time (seconds) of S-CDFSF vs. CDFSF with $\alpha = 0.05$.

	Sample size			
Networks	200	500	2000	5000
Child	0/0	0/0	0/0	0/1
Alarm	0/0	0/0	1/2	4/9
Insurance10	1/2	3/5	14/27	53/109
Child10	1/1	1/3	9/17	67/116
Gene	12/22	32/64	191/410	2306/4668

small. Thus, the time complexity of S-CDFSF is almost the same as MMPC.

Tables 3 and 4 show the normalized numbers of the conditional independence tests performed for the CDFSF and S-CDFSF algorithms that we have implemented when the values of α are up to 0.01 and 0.05, respectively. The normalized number of the conditional independence tests is the number of conditional independence tests performed by S-CDFSF for a particular sample size of a network divided by CDFSF's tests on the same dataset. The normalized value is smaller than the one corresponding to S-CDFSF performing fewer tests than CDFSF. From these two tables, we can see that, in general, S-CDFSF performs fewer tests than CDFSF.

Tables 5 and 6 compare the running time of CDFSF and S-CDFSF with the values of α up to 0.01 and 0.05, respectively. In these two tables, A and B denotes the running time of S-CDFSF and CDFSF, respectively. From the tables, we can see that S-CDFSF is more efficient than CDFSF, especially with large sample sizes. Therefore, we can see that S-CDFSF is not only time-efficient, but also exhibits a highly competitive performance with CDFSF, especially with large sample sizes.

4.5. An analysis of the asymptotic behavior of three algorithms

With streaming features, the following question is also interesting: how close to the real CPCs are the CPCs provided by CDFSF and S-CDFSF as the number of features increases? Studying this question may help us to decide whether to stop each of these algorithms at some point (e.g. when calculating the next feature becomes too expensive or no more available features arrive) or to continue until the last feature is added. This study can also help us to determine how to build CPC sets gradually as the time goes on. The precision metric evaluates the true positives in the output, that is, the number of real parents and children in the output, and the recall metric assesses how close to the real CPCs in a real Bayesian network are the CPCs in the output provided by CDFSF and S-CDFSF. Thus, those two metrics can characterize the gradual building process of CPC sets as the number of features increases.

In this section, we study this asymptotic behavior of our two algorithms using precision and recall metrics to dynamically assess CPC sets by stages. At the same time, compared with the proposed algorithms, the asymptotic behavior of MMPC is also studied. In this situation, we run MMPC separately many times for retrieving the CPC set of each available feature so far after the introduction of each feature.

Figure 6 shows how the average precision and recall measures of all features in a real Bayesian network change as the number of the total features increases, when the parameter α is up to 0.01 and the number of samples is set 5000. From Fig. 6, we can see how much worse our algorithms are on the precision and recall metrics to do the gradual buildup of CPC set of each feature available so far as time goes on. Thus, we can decide whether to stop each algorithm at some point when retrieving the CPC set of each available feature so far reaches satisfied thresholds of the two metrics.

From Fig. 6, we can see that, at every stage, our two algorithms have a higher precision than MMPC except for the child network. On the recall metric, as the number of features increases, the numbers of real parents and children found by the three algorithms gradually approach the number of parents and children in a real Bayesian network. Moreover, the recalls of our two algorithms are almost the same as that of MMPC. Thus, our two algorithms can efficiently deal with the discovery problems of direct causes and effects with streaming features and we can dynamically control the process of generating features according to the asymptotic behavior of our algorithms and some given thresholds.

In summary, from the precision, recall and distance metrics, under the assumption that all independence tests are reliable, S-CDFSF is highly competitive with CDFSF. Compared with MMPC, our algorithms are superior on most of the networks on the precision and distance metrics. On the recall metric, our algorithms are inferior to MMPC when the sample size is small because of lacking the global information of the feature set. But our algorithms are highly competitive with MMPC with large samples.

Moreover, on time efficiency, S-CDFSF is not only efficient, but also exhibits a highly competitive performance with CDFSF, especially with large sample sizes. Compared with MMPC, the time complexity of S-CDFSF is also almost the same.

Finally, we can conclude that our algorithms can not only learn direct causal relationships with streaming features well, but also dynamically control the learning process given a user pre-defined threshold when features stream in over time, as shown in Fig. 6.

5. RELATED WORK

Learning Bayesian networks is one of the most common methods to explore the causal relationships in the observed data. Structure learning of Bayesian networks is broadly classified into two classes of methods: score-based methods and constraint-based methods. Score-based methods compute the probability of the data D given a structure. Exhaustive model selection involves scoring all possible network structures on a given set of features and then picking the structure with the highest scores [7, 9, 11–13, 31–33]. Constraint-based methods use independence/dependence tests between two features to add/remove edges between features and orient them [4, 8, 10]. Typically, the tests are performed using statistical or information theoretic measures.

Structure learning of Bayesian network methods can also be classified as global or local learning approaches. The global learning approach attempts to uncover a complete Bayesian network over all of model features, but can only deal with no more than 300 features. Moreover, learning a full Bayesian network structure is an NP-hard problem from data, as the number of directed acyclic graphs is exponential to the number of variables, and an exhaustive search is intractable [34–36].

The recent explosion of high dimensionality data sets in the biomedical realm and other domains has posed a great challenge to existing global learning algorithms, since they do not reliably scale up to thousands of variables in a reasonable time [14]. Thus, the local learning approach without learning a full Bayesian network in advance has received considerable attention in machine learning [14–17]. In general, local learning as an effective means when dealing with hundreds of thousands of features is emphasized on two specific tasks: (i) identification of features that are direct causes or direct effects of the target of interest and (ii) discovery of the Markov blanket of the target of interest. With the direct causes or direct effects, or Markov blankets of all features by a local learning algorithm, we can first get the skeleton (i.e. edges without their orientation) of a Bayesian network and then orient the skeleton using a greedy search.

With a faithful Bayesian network, the first task is to find the parents and children of a target of interest (PC(T) for short). Two major algorithms HITON_PC and MMPC for the discovery of PC(T) were introduced by Aliferis *et al.* [37] and Tsamardinos *et al.* [38], respectively. The Max–Min Parents and Children (MMPC) algorithm discovers the parents or children of a target of interest using a two-phase scheme. MMPC first includes in CPC (candidate parents and children) the variable with the highest univariate association with T. MMPC chooses to include next into CPC the variable that exhibits the maximum association with T conditioned on the subset of CPC that achieves the minimum association possible for this variable. Intuitively the heuristic is the following: select the variable that, despite our best efforts to make it independent of T (i.e. considering the minimum association conditioned on all possible subsets of CPC) has the highest such minimum association with T among all other candidate variables. In the second phase, MMPC examines whether each variable of CPC can be d-separated from T by conditioning on all possible subsets of CPC. HITON_PC is similar to MMPC and also has a two-phase scheme. The differences are that the former interleaves the two phases and the heuristic used in the first phase is simpler than the one used by MMPC.

Both MMPC and HITON_PC can only discover the parents or children of one target at each run. If we get the parents or children for all features, we need to run MMPC or HITON_PC for each feature independently. Meanwhile, MMPC and HITON_PC identify only the feature set of parents or children of one target of interest without distinguishing between the two.

For the second task, the discovery of the Markov blanket of a target T (MB(T) for short) is to find the set of parents, children and parents of the children for the target of interest in a faithful

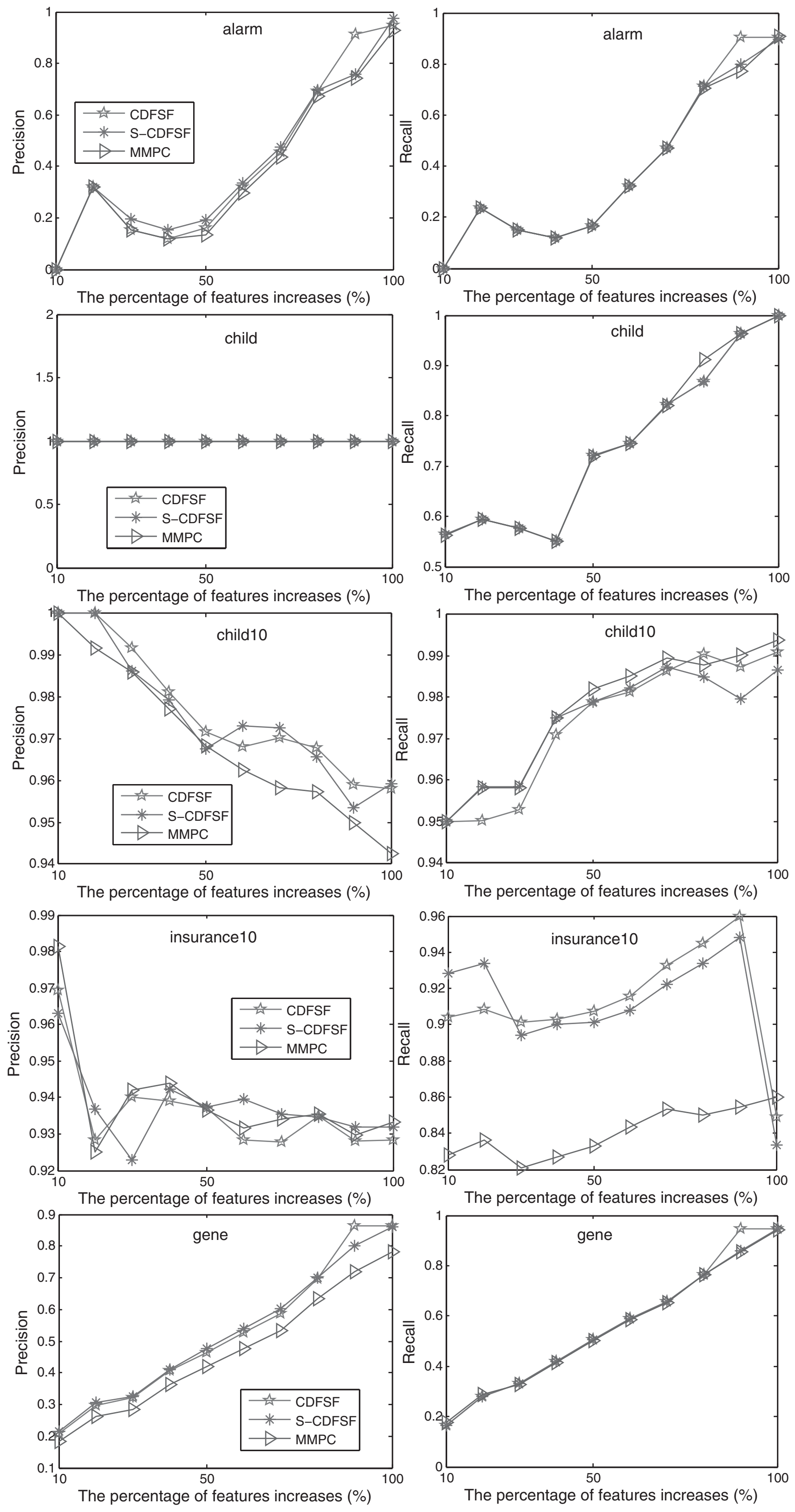

FIGURE 6. Asymptotic behavior as the number of the features seen increases.

Bayesian network. Margaritis and Thrun [17] first invented a sound algorithm, GS for discovery of MB(T). Based on the GS algorithm, an IAMB algorithm was presented which guarantees to find the actual Markov blanket given enough training data and is more sample efficient than GS [39]. However, it still requires a sample size exponential in the size of the Markov blanket. Based on the IAMB algorithm and HITON_MB derived from HITON_PC, MMMB developed from MMPC was introduced without requiring samples exponential to the size of the Markov blanket. Following the idea of MMMB and HITON-MB, PCMB was also proposed to conquer the data inefficiency problem of IAMB [18].

6. CONCLUSIONS

Although there have been many studies on causal discovery, there is no related work for causal discovery in the context of streaming features. In this paper, we have studied this new research problem and presented two novel algorithms. To the best of our knowledge, this is the first attempt to address causal discovery from steaming features. Since statistically distinguishing between direct causes and direct effects of a target feature of interest is a hard problem, in this work, we deal with the problem of identifying the feature set of both the direct causes and direct effects of a target feature from streaming features without distinguishing between the two.

In order to demonstrate the effectiveness of our algorithms, we compared them with the state-of-the-art algorithm of causal discovery, MMPC, which assumes that all features are known in advance. The experimental results showed that our methods are highly competitive with the MMPC algorithm and also revealed the effectiveness of our algorithms for the task of causal discovery from streaming features. Meanwhile, many issues for causal discovery from streaming features remain wide open. We list here two problems which, in our opinion, deserve further investigations. One is that our work only identifies both the direct causes and direct effects of the targets of interest, and how to extend our work for dynamical construction of Bayesian network structure for real-world data sets for classification with streaming features needs to be further explored. The other is to improve the recall metric of our methods on data sets with a small sample size.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers for their constructive comments which have improved this paper.

FUNDING

This work is supported by the National Natural Science Foundation of China (60975034, 61070131, 61175051 and 61005007), the US National Science Foundation (CCF-0905337) and the US NASA Research Award (NNX09AK86G).

REFERENCES

[1] Pearl, J. (1995) Causal diagrams for empirical research (with discussion). *Biometrika*, **82**, 669–710.

[2] Pearl, J. (2000) *Causality: Models, Reasoning, and Inference* (2nd edn, 2009). Cambridge University Press, New York.

[3] Pearl, J. and Verma, T. (1991) A Theory of Inferred Causation. In Allen, J.F., Fikes, R. and Sandewall, E. (eds), *KR'91: Principles of Knowledge Representation and Reasoning*, Cambridge, MA, April 22–25, pp. 441–452. Morgan Kaufmann Publishers, Inc., San Francisco.

[4] Spirtes, P., Glymour, C. and Scheines, R. (2000) *Causation, Prediction, and Search* (2nd edn). MIT Press, Cambridge, MA.

[5] Van Der Gaag, L.C. (1996) Bayesian belief networks: odds and ends. *Comput. J.*, **39**, 97–113.

[6] Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA.

[7] Heckerman, D., Geiger, D. and Chickering, D.M. (1995) Learning Bayesian networks: the combination of knowledge and statistical data. *Mach. Learn.*, **20**, 197–243.

[8] Cheng, J., Greiner, R., Kelly, J., Bell, D. and Liu, W. (2002) Learning Bayesian networks from data: an information-theory based approach. *Artif. Intell.*, **137**, 43–90.

[9] Chickering, D.M. (2002) Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.*, **2**, 445–498.

[10] Dash, D. and Druzdzel, M. (2003) Robust Independence Testing for Constraint-Based Learning of Causal Structure. *Proc. UAI'03*, Acapulco, Mexico, August 8–10, pp. 167–174. Morgan Kaufmann Publishers, Inc., San Francisco.

[11] Daly, R. and Shen, Q. (2009) Learning Bayesian network equivalence classes with ant colony optimization. *J. Artif. Intell. Res.*, **35**, 391–447.

[12] de Campos, C.P. and Ji, Q. (2011) Efficient structure learning of Bayesian networks using constraints. *J. Mach. Learn. Res.*, **12**, 663–689.

[13] Yuan, C., Malone, B. and Wu, X. (2011) Learning Optimal Bayesian Networks Using A* Search. *Proc. IJCAI'11*, Barcelona, Catalonia, July 16–22, pp. 2186–2191, AAAI Press, Menlo Park, CA.

[14] Tsamardinos, I., Brown, L.E. and Aliferis, C.F. (2006) The max–min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, **65**, 31–78.

[15] Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S. and Koutsoukos, X. (2010) Local causal and Markov blanket induction for causal discovery and feature selection for classification part I: algorithms and empirical evaluation. *J. Mach. Learn. Res.*, **11**, 171–234.

[16] Aliferis, C.F, Statnikov, A., Tsamardinos, I., Mani, S. and Koutsoukos, X. (2010) Local causal and Markov blanket induction for causal discovery and feature selection for classification part II: analysis and extensions. *J. Mach. Learn. Res.*, **11**, 235–284.

[17] Margaritis, D. and Thrun, S. (2000) Bayesian Network Induction Via Local Neighborhoods. In Solla, S.A., Leen, T.K. and Muller, K.R. (eds), *Advances in Neural Information Processing Systems*, Vol. 12, pp. 505–511. MIT Press, Cambridge, MA.

[18] Peña, J.M., Nilsson, R., Björkegren, J. and Tegnér, J. (2007) Towards scalable and data efficient learning of Markov boundaries. *Int. J. Approx. Reason.*, **45**, 211–232.

[19] Masud, M., Chen, Q., Khan, L., Aggarwal, C., Gao, J., Han, J. and Thuraisingham, B.I. (2010) Addressing Concept-Evolution in Concept-Drifting Data Stream. *Proc. ICDM'10*, Sydney, December 3–17, pp. 929–934. IEEE Computer Society, Washington, DC.

[20] Wu, X., Xie, F., Wu, G. and Ding, W. (2011) Personalized News Filtering and Summarization on the Web. *The 23rd IEEE Int. Conf. Tools with Artificial Intelligence (ICTAI'11)*, Boca Raton, FL, November 7–9, pp. 414–421. IEEE Computer Society, Washington, DC.

[21] Wu, X., Yu, K., Wang, H. and Ding, W. (2010) Online Streaming Feature Selection. *Proc. ICML'10*, Haifa, Israel, June 21–24, pp. 1159–1166. Omnipress, Madison, WI.

[22] Zhou, J., Foster, D., Stine, R. and Ungar, L. (2006) Streamwise feature selection. *J. Mach. Learn. Res.*, **7**, 1861–1885.

[23] Perkins, S. and Theiler, J. (2003) Online Feature Selection Using Grafting. *Proc. ICML'03*, Washington, DC, August 21–24, pp. 592–599. Omnipress, Madison, WI.

[24] Aliferis, C.F., Tsamardinos, I., Statnikov, A. and Brown, L. (2003) Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery. *Proc. METMBS'03*, Las Vegas, NV, June 23–26, pp. 371–376. CSREA Press, Athens.

[25] Cowell, R.G., Dawid, A.P., Lauritzen, S.L. and Spiegelhalter, D.J. (1999) *Probabilistic Networks and Expert Systems*. Springer, New York.

[26] Binder, J., Koller, D., Russell, S. and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Mach. Learn.*, **29**, 213–244.

[27] Beinlich, I.A., Suernondt, H., Chavez, R. and Cooper, G. (1989) The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. *Proc. 2nd European Conf. AI and Medicine*, London, August, pp. 247–256. Springer, Berlin.

[28] Statnikov, A., Tsamardinos, I. and Aliferis, C.F. (2003) An Algorithm for the Generation of Large Bayesian Networks. Technical Report DSL-03-01, Vanderbilt University.

[29] Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D. and Futcher, B. (1998) Comprehensive identification of cell cycle regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

[30] Friedman, N., Nachman, I. and Pe'er, D. (2000) Using Bayesian networks to analyze expression data. *Comput. Biol.*, **7**, 601–620.

[31] Friedman, N., Nachman, I. and Pe'er, D. (1999) Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate". *Proc. UAI'99*, Stockholm, Sweden, July 30–August 1, pp. 206–215. Morgan Kaufmann Publishers, Inc., San Francisco.

[32] Kojima, K., Perrier, E., Imoto, S. and Miyano, S. (2010) Optimal search on clustered structural constraint for learning Bayesian network structure. *J. Mach. Learn. Res.*, **11**, 285–310.

[33] Patnaik, D., Laxman, S. and Ramakrishnan, N. (2011) Discovering excitatory relationships using dynamic Bayesian networks. *Knowl. Inf. Syst.*, **29**, 273–303.

[34] Chickering, D.M., Heckerman, D. and Meek, C. (2004) Large-sample learning of Bayesian networks is NP-hard. *J. Mach. Learn. Res.*, **5**, 1287–1330.

[35] Yu, X. and Lam, W. (2011) Probabilistic joint models incorporating logic and learning via structured variational approximation for information extraction. *Knowl. Inf. Syst.*, doi: 10.1007/s10115-011-0455-8.

[36] Cercone, N., An, X., Li, J., Gu, Z. and An, A. (2011) Finding best evidence for evidence-based best practice recommendations in health care: the initial decision support system design. *Knowl. Inf. Syst.*, **29**, 159–201.

[37] Aliferis, C.F., Tsamardinos, I. and Statnikov, A. (2003) HITON: A Novel Markov Blanket Algorithm for Optimal Variable Selection. *American Medical Informatics Association (AMIA) Annual Symposium*, Vancouver, British Columbia, November 18–22, pp. 21–25. AMIA, Bethesda, Maryland, USA.

[38] Tsamardinos, I., Aliferis, C.F. and Statnikov, A. (2003). Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. *Proc. KDD'03*, Washington, DC, August 24–27, pp. 673–678. ACM, New York.

[39] Aliferis, C.F. and Tsamardinos, I. (2002). Algorithms for Large-Scale Local Causal Discovery and Feature Selection in the Presence of Small Sample or Large Causal Neighborhoods. Technical Report DSL 02–08, Department of Biomedical Informatics, Vanderbilt University.