Fast Causal Network Inference over Event Streams

Saurav Acharya and Byung Suk Lee

Department of Computer Science, University of Vermont, Burlington, VT 05405, USA
{sacharya,bslee}@uvm.edu

Abstract. This paper addresses causal inference and modeling over event streams where data have high throughput and are unbounded. The availability of large amount of data along with the high data throughput present several new challenges related to causal modeling, such as the need for fast causal inference operations while ensuring consistent and valid results. There is no existing work specifically for such a streaming environment. We meet the challenges by introducing a time-centric causal inference strategy that leverages temporal precedence information to decrease the number of conditional independence tests required to establish the dependencies between the variables in a causal network. Dependency and temporal precedence of cause over effect are the two properties of a causal relationship. We also present the *Temporal Network Inference* algorithm to model the temporal precedence relations into a temporal network. Then, we propose the *Fast Causal Network Inference* algorithm for faster learning of causal network using the temporal network. Experiments using synthetic and real datasets demonstrate the efficacy of the proposed algorithms.

Keywords: Causal inference; Event streams; Temporal data.

1 Introduction

In recent years, there has been a growing need for active systems that can perform causal inference in diverse applications such as health care, stock markets, user activity monitoring, smart electric grids, and network intrusion detection. These applications need to infer the cause of abnormal activities immediately such that informed and timely preventive measures are taken. As a case in point, consider a smart electric grid monitoring application. The failure of a component can cause cascading failures, effectively causing a massive blackout. Therefore, the identification of such cause and effect components in a timely manner enables preventive measures in the case of failure of a cause component, thereby preventing blackouts.

Causal network, a directed acyclic graph where the parent of each node is its direct cause, has been popularly used to model causality [1–7]. There are two distinct types of algorithms for learning a causal network: score-based [1–4] and constraint-based [5–8]. Both types of algorithms are slow and, therefore,

L. Bellatreche and M.K. Mohania (Eds.): DaWaK 2013, LNCS 8057, pp. 222–235, 2013.

not suitable for event streams where prompt causal inference is required. Score-based algorithms perform a greedy search (usually hill climbing) to select a causal network with the highest score from a large number of possible networks. With an increase in the number of variables in the dataset, the number of possible networks grows exponentially, resulting in slow causal network inference. On the other hand, constraint-based algorithms (e.g., PC algorithm [7]) discover the causal structure via a large number of tests on conditional independence(CI). There can be no edge between two conditionally independent variables in the causal network [9]. In a causal network of n variables, two variables X and Y are said to be conditionally independent given a condition set S if there is at least one variable in S such that X and Y are independent. The condition set S consists of all possible 2^{n-2} combinations of the remaining $n-2$ variables, and therefore the computational complexity grows exponentially as the number of variables increases. So, the current techniques for causal inference are slow and not suitable for event streams which have a high data throughput and where the number of variables (i.e., event types) is large.

With this concern, this paper describes a new time-centric causal modeling approach to speed up the causal network inference. Every causal relationship implies temporal precedence relationship [10]. So, the idea is to incorporate temporal precedence information as an important clue to reducing the number of required CI tests and thus maintaining feasible computational complexity. This idea achieves fewer computations of CI test due to two factors. First, since causality requires temporal precedence, we ignore the causality test for those nodes with no temporal precedence relationship between them. Second, in the CI test of an edge, we exclude those nodes from the condition set which do not have temporal precedence relationship with the nodes of the edge. Therefore, it reduces the size of the condition set which is a major cause of the exponential computational complexity. In addition, the temporal precedence relationship intuitively orients the causal edge unlike the constraint-based algorithms where a separate set of rules are needed to infer the causal direction (details in Section 3.3).

The contributions of this paper are summarized as follows. First, it presents a temporal network structure to represent temporal precedence relationships between event types and proposes an algorithm, *Temporal Network Inference(TNI)*, to construct a temporal network applicable in streaming environment. Second, it introduces a time-centric causal modeling strategy and proposes an algorithm, *Fast Causal Network Inference(FCNI)*, to speed up the learning of causal network. Finally, it empirically demonstrates the advantages of the proposed algorithm in terms of the running time and the total number of CI tests required for the learning of causal network by comparing it against the state-of-art algorithm for causal network inference, called the PC algorithm (details in Section 3.3).

The rest of this paper is organized as follows. Section 2 reviews the existing work on causal network inference. Section 3 presents the basic concepts used in the paper. Section 4 and Section 5 propose the learning of temporal network and faster causal network, respectively. Section 6 evaluates the proposed FNCI algorithm. Finally, Section 7 concludes the paper and mentions further research.

2 Related Work

As explained earlier, there are two main approaches for causal network inference.

The first approach, score-based [1–4], performs greedy search (usually hill climbing) over all possible network structures in order to find the network that best represents the data based on the highest score. This approach, however, has two problems. First, it is slow due to the exhaustive search for the best network structure. An increase in the number of variables in the dataset increases the computational complexity exponentially. Second, two or more network structures, called the equivalence classes [11], may represent the same probability distribution, and consequently the causal directions between nodes are quite random. There is no technique for alleviating these problems in a streaming environment. Thus, score-based algorithms are not viable for streams.

The second approach, constraint-based [5–8], does not have the problem of equivalence classes. However, it is slow as it starts with a completely connected undirected graph and thus performs a large number of CI tests to remove the edges between conditionally independent nodes. The number of CI tests increases exponentially with the increase in the number of variables in the dataset. To alleviate this problem, some constraint-based algorithms start with a minimum spanning tree to reduce the initial size of condition sets. However, this idea trades the speed with the accuracy of the causal inference. The constraint-based algorithms include IC* [5], SGS [6], PC [7], and FCI algorithm [7]. The FCI algorithm focuses on the causal network discovery from the dataset with latent variables and selection bias, which is quite different from the scope of this paper. The PC algorithm is computationally more efficient than IC* and SGS. This is why we evaluate the proposed *FCNI* algorithm by comparing it against the PC algorithm. Like the others, the PC algorithm starts with a completely connected undirected graph. To reduce the computational complexity, it performs CI tests in several steps. Each step produces a sparser graph than the earlier step, and consequently, the condition set decreases in the next step. However, the computational complexity is still $O(n^2 \cdot 2^{n-2})$. (The details are explained in Section 3.3.) Therefore, the current constraint-based algorithms are not suitable for fast causal inference over streams.

To the best of our knowledge, there exists no specific work in the causal network inference in a streaming environment. A new approach is needed for faster causal network inference.

3 Basic Concepts

This section presents some key concepts needed to understand the paper.

3.1 Event Streams, Type, and Instance

An event stream in our work is a sequence of continuous and unbounded timestamped events. An event refers to any action that has an effect and is created by one event owner. One event can trigger another event in chain reactions. Each

event instance belongs to one and only one event type which is a prototype for creating the instances. Two event instances are related to each other if they share common attributes such as event owner, location, and time. We call these attributes *common relational attributes(CRAs)*.

In this paper we denote an event type as E_j and an event instance as e_{ij}, where i indicates the *CRA* and j indicates the event type.

Example 1. Consider a diabetic patient monitoring system in a hospital. Each patient is uniquely identifiable, and each clinical test or measurement of each patient makes one event instance. For example, a patient is admitted to the hospital, has their blood pressure and glucose level measured, and takes medication over a period of time. This creates the instances of the above event types as a result. Typical event types from these actions include regular-insulin-dose-given, hypoglycemic-symptoms-exists, blood-glucose-measurement-decreased, increased, etc. Note that the patient ID is the CRA, as the events of the same patient are causally related.

3.2 Conditional Mutual Information

A popular approach for testing the conditional independence, with respect to the joint probability P, of two random variables X and Y given a subset of random variables S is conditional mutual information(CMI) (e.g., [8, 12]). CMI gives the strength of dependency between variables in a measurable quantity, which helps to identify strong and weak causal relationships in the final causal network.

To test whether X and Y are conditionally independent given S, we compute the conditional mutual information $I_{MI}(X, Y|S)$ as

$$I_{MI}(X, Y|S) = \sum_{x \in X} \sum_{y \in Y} \sum_{s \in S} p_{X,Y,S}(x, y, s) log_2 \frac{p_{X,Y|S}(x, y|s)}{p_{X|S}(x|s) p_{Y|S}(y|s)}$$

where p is the probability mass function calculated from the frequencies of variables.

We only keep the record of these frequencies, not the whole events, by updating them as a new batch of events arrives. Consequently, the independence test procedure is incremental in our case.

It is said that two variables X and Y are independent when $I_{MI}(X, Y|S) = 0$; otherwise, they are dependent. However, this presents us with the risk of spurious relationships due to weak dependencies (we cannot assume $I_{MI}(X, Y|S) = 10^{-5}$ and $I_{MI}(X, Y|S) = 10$ provide the same degree of confidence in the dependency). With an increase in the value of $I_{MI}(X, Y|S)$, the dependency between the variables X and Y grows stronger. Therefore, to prune out the weak dependencies, we need to set a threshold value of mutual information below which we ignore the evidence as weak. To do so, we relate CMI with G^2 test statistics [7, 13] as below where N_s is the number of samples.

$$G^2(X, Y|S) = 2 \cdot N_s \cdot log_e 2 \cdot I_{MI}(X, Y|S)$$

Under the independence assumption, G^2 follows the χ^2 distribution [14], with the degree of freedom *df* equal to $(r_x - 1)(r_y - 1)\prod_{s \in S} r_s$, where r_x, r_y, and r_s are the number of possible distinct values of X, Y, and S, respectively. So, we use χ^2 test, which provides a threshold based on *df* and significance level α, to validate the dependency result. We set α as the universally accepted value of 95%.

3.3 The PC Algorithm

The PC algorithm [7] (Algorithm 1) starts with a completely connected undirected graph on which the CI tests are performed to remove edges between independent nodes. The key idea is that a causal network has an edge between X and Y in the topology if and only if X and Y are not independent given all condition subsets of the remaining neighbor nodes [15]. In Algorithm 1, the topology of the causal network is learned in the steps 1 to 10. The network topology is then assigned causal direction in the steps 11 to 17.

Algorithm 1. PC algorithm

1: Construct the completely connected undirected graph G on the n nodes;
2: Initialize Neighbors(G,X) as the set of nodes adjacent to the node X in G, and SepSet(X,Y), a set of nodes that causes independence between X and Y nodes, as empty;
3: $k \leftarrow 0$;
4: **repeat**
5: **repeat**
6: Select any edge $X - Y$ such that $|Neighbors(G, X) \backslash Y| \geq k$;
7: **repeat**
8: Select any subset S of $Neighbors(G, X) \backslash Y$ such that $|Neighbors(G, X) \backslash Y| = k$;
9: If X and Y are independent given S, remove $X - Y$ from G, remove Y from Neighbors(G,X), remove X from Neighbors(G,Y), and add S to SepSet(X,Y) and SepSet(Y,X);
10: **until** every subset S of $Neighbors(G, X) \backslash Y$ such that $|Neighbors(G, X) \backslash Y| = k$ has been selected.
11: **until** every edge $X - Y$ such that $|Neighbors(G, X) \backslash Y| \geq k$ has been selected.
12: k = k + 1;
13: **until** every edge $X' - Y'$ satisfies $|Neighbors(G, X') \backslash Y'| < k$.
14: **for** each triplet of nodes X, Y, Z such that the edges $X - Y$ and $Y - X$ exist in G but not $X - Z$ **do**
15: Orient $X - Y - Z$ as X→Y←Z if and only if SepSet(X,Z) does not contain Y;
16: **end for**
17: **repeat**
18: **If** there exists $X \rightarrow Y$ and $Y - Z$, but not $X - Z$, **then** orient $Y - Z$ as $Y \rightarrow Z$;
19: **If** there exists $X - Y$ and a directed path from X to Y, **then** orient $X - Y$ as $X \rightarrow Y$;
20: **until** no edge can be oriented.

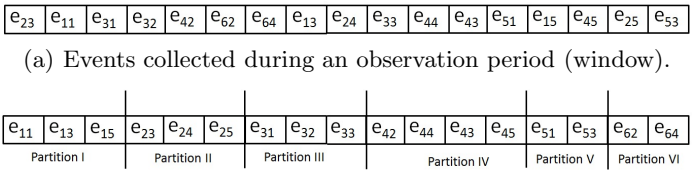
4 Learning Temporal Precedence Relationships

In this section, we describe an incremental approach to model temporal precedence relationships from time-stamped events into a *temporal network*.

4.1 Temporal Network Model

A temporal network is a directed acyclic graph of nodes representing event types where an edge between two nodes represents the temporal precedence relationship between them. To facilitate the handling of events in a streaming environment, we use a time-based window over the stream. Typically, the application offers a natural observation period (e.g., hour) that makes a window.

As mentioned earlier in Section 3.1, two events are related to each other if they share the same *common relational attribute(CRA)*). So, the events in a window are arranged by CRA and ordered by the timestamp as they arrive, producing a *partitioned window* as a result. Figure 1 illustrates it.



(a) Events collected during an observation period (window).

(b) Events in the window partitioned by CRA.

Fig. 1. Partitioned window of events

With the arrival of a new batch of event instances, we augment each partition in the new window by prefixing it with the last instance of the partition with the same CRA value in the previous window. This is necessary in order to identify the temporal precedence between instances that are separated into the two consecutive batches.

To determine when an edge, say $E_i \rightarrow E_j$, should be added in a temporal network, a measure providing an evidence of temporal precedence between the event types should be defined. The evidence we use is the frequency of the observation of an instance of E_j following an instance of E_i. The *temporal strength* of an edge identified is defined below.

Definition 1 (Temporal strength). *Consider an edge $E_i \rightarrow E_j$ $(i \neq j)$ in a temporal network of n event types. Let f_{ij} be the total number of observations in which an event of type E_i precedes an event of type E_j over all partitions in the* partitioned window*. Then, we define* temporal strength*, s_{ij}, of the edge $E_i \rightarrow E_j$ as*

$$s_{ij} \triangleq \frac{f_{ij}}{\sum_{k=0}^{(n-1)} f_{ik}}$$

4.2 Temporal Network Inference Algorithm

The idea behind the *TNI* algorithm is to collect events from an event stream in a *window* and then use temporal precedence information from the sequence of event pairs in the window to construct a temporal network at the event type level. The overall algorithm is centered on a *frequency matrix*, which is initially empty (i.e., all zero elements) and updated with each new batch of events. The algorithm has two steps for each window, outlined in Algorithm 2.

1. Update the frequency matrix *FM* by observing the precedence relationships of event pairs in the partitioned window (steps 3–13 in Algorithm 2). An element f_{ij} in *FM* reflects the total number of times events of type E_i precede events of type E_j ($i \neq j$). Each time an event pair (e_{oi}, e_{oj}) is observed in the event stream such that e_{oi} precedes e_{oj}, increase the value of f_{ij} by 1.
2. Determine the edges of the temporal network using the frequency matrix (steps 14–24 in Algorithm 2). For each pair of an edge and its reversed edge, select the edge with the greater frequency. Calculate the temporal strength of the selected edge, e.g., $E_i \rightarrow E_j$, and store it in the element s_{ij} of the *strength matrix SM*. Set the strength of the ignored edge with the lower frequency to zero. If a cycle is introduced, remove the edge with the lowest temporal strength in the cycle.

5 Learning Causal Network in Reduced Time

In this section, we describe a new approach to reduce the number of CI tests needed to infer the causal structure, thereby speeding up the learning process. The idea is to incorporate temporal precedence relationships to learn the causal network. The correctness of our approach is shown as follows. First, a temporal precedence relationship is a mandatory condition for inferring causality [10]. Therefore, causal relationship subsumes temporal precedence relationship, that is, the causal network is a subgraph of the temporal network. Second, a causal network should satisfy the *Causal Markov Condition (CMC)* [6, 9, 16] where for every node X in the set of nodes N, X is independent of its non-descendants excluding its parents (i.e., $N \backslash (Descendants(X) \cup Parents(X))$) conditional on its parents. In a temporal network of vertex (or node) set N, a node is temporally independent, and therefore causally independent, of all its non-descendants (except its parents) given its parents. In other words, the temporal network obeys CMC which is a necessary condition for the causal network. Therefore, our idea of considering a temporal network as an initial causal network is correct.

5.1 Fast Causal Network Inference Algorithm

The idea behind FCNI algorithm is to reduce the number of CI tests by incorporating temporal precedence information. The algorithm has two steps, as outlined in Algorithm 3.

Algorithm 2. Temporal Network Inference (TNI)

Require: an edgeless network structure TN, event stream(s) S

1: Initialize an empty *frequency matrix (FM)*, an empty strength matrix SM, two empty buffers B_p and B_c (used to store "parent" events and "child" events, respectively);
2: **for** each window W in S **do**
3: **for** each partition P (corresponding to CRA a) in W **do**
4: **for** i = 1 to $t_n - 1$ where t_n is the number of unique timestamp in P **do**
5: Clear B_p and B_c;
6: Insert all events with timestamp t_i and t_{i+1} into B_p and B_c, respectively;
7: **for** all event instances e_{ap} and e_{ac} in B_p and B_c, respectively, **do**
8: **if** type(e_{ac}) $\neq$ type(e_{ap}) {*//There cannot be causal relationships between events of the same type.*} **then**
9: Increase the frequency of element $f_{type(e_{ap}),type(e_{ac})}$ in FM by 1;
10: **end if**
11: **end for**
12: **end for**
13: **end for**
14: **for** each pair of elements f_{ij} and f_{ji} in FM **do**
15: $s_{ij} \leftarrow 0$, $s_{ji} \leftarrow 0$;
16: **if** $f_{ij} > f_{ji}$ **then**
17: Add an edge $E_i \rightarrow E_j$ in TN and set its strength to $s_{ij} = \frac{f_{ij}}{\sum_{k=0}^{n-1} f_{ik}}$;
18: **else if** $f_{ji} > f_{ij}$ **then**
19: Add an edge $E_j \rightarrow E_i$ in TN and set its strength to $s_{ji} = \frac{f_{ji}}{\sum_{k=0}^{n-1} f_{jk}}$;
20: **end if**
21: **end for**
22: **if** an edge is added and it introduces cycle in TN **then**
23: Remove the edge with the lowest temporal strength (in SM) in the cycle;
24: **end if**
25: **end for**

1. The first step is to construct a temporal network by running the *TNI* algorithm. The temporal network is set as the initial causal network. Note that since temporal precedence is a requirement for a causal relationship, all causal relationships are theoretically guaranteed be in the temporal network.
2. The second step is to adapt the ideas of constraint-based algorithms to learn the final causal network by pruning out the edges between independent nodes. We perform CI tests on every edge between nodes in the initial causal network to verify dependency between them. Conditionally independent nodes are considered to be spurious and hence the edge between them is removed. Steps 2 to 22 perform this step. The main difference from the PC algorithm is the manner in which CI tests are performed. In the PC algorithm, the condition set S for an edge $E_i - E_j$ considers the neighbors of both E_i and E_j whereas in the FCNI algorithm, as the edges are already directed, the condition set S for an edge $E_i \rightarrow E_j$ needs to consider only the parents of E_j (E_j is independent of the parents of E_i that do not have edge to E_j). Consequently, we need fewer CI tests.

Algorithm 3. Fast Causal Network Inference (FCNI)

Require: *Window W, Edgeless Causal Network G = (N, ξ).*{N and ξ are the set of nodes and the set of edges, respectively.}

```
1: Run the TNI algorithm and initialize G with the learned temporal network;
2: for each directed edge (E_i, E_j) ∈ ξ do
3:    independent = IsIndependent(E_i, E_j, φ), where φ is the empty set;
      {IsIndependent(E_i, E_j, S) calculates I_MI(E_i, E_j|S) for CI test.}
4:    if independent is true then
5:       Remove (E_i, E_j) from ξ;
6:    end if
7: end for
8: k ← 0;
9: repeat
10:   for each directed edge (E_i, E_j) ∈ ξ do
11:      Construct a set of condition sets, Z, each of cardinality k from the parents of
         E_j excluding E_i;
12:      repeat
13:         Select any subset S from Z;
14:         independent = IsIndependent(E_i, E_j, S);
15:         Remove S from Z;
16:      until Z is empty or independent is true
17:      if independent is true then
18:         Remove (E_i, E_j) from ξ;
19:      end if
20:   end for
21:   k = k + 1;
22: until number of parents of E'_j in every directed edge (E'_i, E'_j) ∈ ξ is less than k.
```

5.2 Complexity Analysis

The complexity of the FNCI algorithm for a causal network G is bounded by the largest degree of each node. Let n be the number of nodes (i.e., event types). Then in the worst case, since the causal network inference starts with a temporal network, the number of CI tests required by the algorithm is given as

$$CI_{max} = \sum_{i=1}^{n} d_i 2^{|Z_i|}$$

where $d_i \equiv (i-1)$ is the maximum degree of incoming edges to the node i (there are $\sum_{i=1}^{n} d_i = \frac{n \cdot (n-1)}{2}$ directed edges in the network G) and Z_i is the maximum condition set to each edge involving node i such that $|Z_i| = d_i - 1 = i - 2$. So the computational complexity of *FCNI* algorithm is $O(n \cdot 2^{n-2})$ in the worst case. In contrast, the PC algorithm (described in Section 3.3), whose condition set of each node is of cardinality $n-2$ nodes, has the worst case computational complexity of $O(n^2 \cdot 2^{n-2})$. Therefore, in the worst case, the *FCNI* algorithm is n times faster than the PC algorithm.

In the best case, the causal network G takes the form of a minimum spanning tree with $n-1$ edges. In this case, the FCNI algorithm and the PC algorithm require $n-1$ and $4n-6$ CI tests, respectively.

Note that the FCNI algorithm starts with a sparse network as it has only those edges that satisfy the temporal precedence relationships. So, in practice, it starts closer to the best case. In contrast, the PC algorithm starts with a completely connected dense network. So, it starts from the worst case.

6 Performance Evaluation

We conducted experiments to compare the proposed FCNI algorithm against the PC algorithm in terms of the accuracy, the running time, and the number of CI tests required on both the algorithms. Section 6.1 describes the experiment setup, including the evaluation metrics and the platform used. Section 6.2 explains the datasets used and Section 6.3 presents the experiment results.

6.1 Experiment Setup

Evaluation Metrics. Intuitively, the quality of causal network learning algorithms are best evaluated by examining how closely the constructed causal network structures resemble the target causal network. In this regard, we adopt the structural Hamming distance proposed by [17] as the quality metric of the output causal network. The nodes (i.e., event types) are fixed as given to the algorithms, and therefore the network structures are compared with respect to the edges between nodes. There are three kinds of possible errors in the causal network construction: reversed edges, missing edges, and spurious edges. We use the number of the erroneous edges of each kind as the evaluation metric.

Platform. The experiments are conducted on RedHat Enterprise Linux 5 operating system using Java(TM) 2 Runtime Environment–SE 1.5.0_07 in Vermont Advanced Computing Core (VACC) cluster computers.

6.2 Datasets

Experiments are conducted using both synthetic and real datasets.

Synthetic Datasets. A synthetic dataset is reverse-engineered from a target causal network. Given the control parameters – the number of event owners n_o and the number of event types n, the idea is to generate a random causal network, and then convert the causal network to an event stream which reflects the underlying probability distribution of the causal network. In the interest of space, the details of the event stream generation are not described here. We assume that the event owner is the *CRA*. The dataset is represented by a collection of files in which the events are shuffled according to the owner ID while preserving the temporal order. We create five datasets (see their profiles in Table 1), representing target causal networks of 4, 8, 12, 16 and 20 nodes each.

Table 1. Profiles of the five synthetic datasets

Dataset	n	$\mathbf{n}_{edges}$	$\mathbf{n}_o$	$\mathbf{n}_{instances}$
DS_1	4	4	5000	13108
DS_2	8	16	30000	108334
DS_3	12	32	500000	3163237
DS_4	16	46	6553600	49008538
DS_5	20	62	52428800	511972810

(n_{edges} is the number of *actual* edges in the network. $n_{instances}$ is the number of event instances in the dataset.)

Real Dataset. The real dataset D_R contains diabetes lab test results [18] of 70 different patients over a period ranging from a few weeks to a few months. The dataset has a total 28143 records, about 402 records for each patient. Each record has four fields – date, time, test code, test value. The clinical data of a patient is independent of other patients. Therefore, the patient ID is the *CRA* for this dataset. There are 20 different test codes appearing in the file from which we define 13 different event types of interest. The details of the event types are omitted due to the space limit.

6.3 Experiment Results

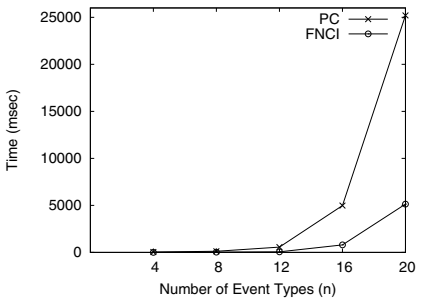
We ran the FCNI and PC algorithms over each of the five synthetic datasets and the real dataset. We present our evaluation results in three parts. First, we compare the quality of the generated networks against the target causal network and determine how closely they resemble the true causal network. (The details of the true causal networks are omitted due to the space limit.) Specifically, we count the number of spurious edges, the number of missing edges, and the number of reversed edges. Second, we compare the running time (CPU time) of the two algorithms, and finally, we evaluate the number of CI tests performed on both algorithms. We show that reducing the number of CI tests is the key to reducing the running time of causal network inference. The experiment is repeated ten times for each dataset (DS_1 through DS_5 and D_R) to calculate the average number of erroneous edges, the average running time, and the average number of CI tests. We assume the events are in temporal order.

Comparison of the Accuracy of the PC and FCNI Algorithms. Table 2 presents the number of erroneous edges in the causal network produced by the PC and FCNI algorithms. The results show that the quality of the causal network from the FCNI algorithm is similar to that of the PC algorithm. First, the number of missing and spurious edges are comparable. This is due to the reliance of both algorithms on the same test statistics (CMI in our case) to infer the independence of two event types. Second, the number of reversed edges is zero for the FCNI algorithm. Clearly the FCNI algorithm, through the temporal network, is much better at determining the correct causal edge direction. It is because the fact that the cause always precedes its effect is embodied in its

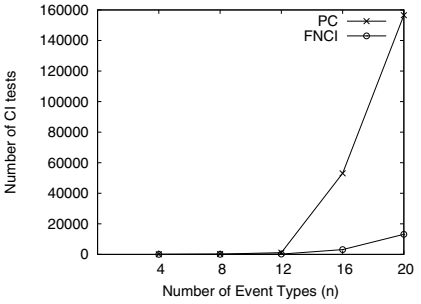
Table 2. Number of erroneous edges

	Algorithms	DS_1	DS_2	DS_3	DS_4	DS_5	D_R
Missing Edges	*PC*	0	0	0	0	1	1
	FCNI	0	1	0	0	1	1
Reversed Edges	*PC*	0	2	0	2	3	2
	FCNI	0	0	0	0	0	0
Spurious Edges	*PC*	0	3	0	4	3	1
	FCNI	0	3	0	4	3	1

temporal precedence relationship. Overall, we conclude that the FCNI algorithm produces almost the same topology as the PC algorithm, while the accuracy of the causal direction is improved.



(a) Running time for varying number of event types.



(b) Number of CI tests for varying number of event types.

Fig. 2. Comparison of the PC and FCNI algorithms

Comparison of the Running Time of the PC and FCNI Algorithms. Figure 2(a) plots the average running time of FCNI and PC algorithms against the number of event types (n) in the synthetic dataset. In all cases, the FCNI algorithm is much faster than the PC algorithm. Clearly, the temporal precedence information helps to reduce the size of condition set and the edges to test. As n increases, the running times of both PC and FCNI algorithms increase. However, the rate of increase of the running time of the PC algorithm is much higher than that of the FCNI algorithm. Therefore, with an increase in n, the ratio of running time between the two algorithms increases. The same observation is made in the real dataset where the running time of the PC and FCNI algorithms are 817 and 118 msecs, respectively. These results verify the important role of temporal precedence relationships to reduce the running time.

Comparison of the Number of CI Tests of the PC and FCNI Algorithms. Figure 2(b) shows that the FCNI algorithm performs fewer CI tests

than the PC algorithm in all synthetic datasets. The CI tests required are minimized, due to the temporal precedence information, by reducing the size of the condition set and the number of edges to test. With an increase in the number of event types (n), the rate of increase of the number of CI tests in the PC algorithm is much higher than that in the FCNI algorithm. A similar observation is made in the real dataset where the number of CI tests of the PC and FCNI algorithms are 1239 and 192, respectively. These results confirm the important role of temporal precedence relationships in reducing the number of CI tests. Note the result of CI tests (Figure 2(b)) looks almost the same as that of the running time (Figure 2(a)). This demonstrates that CI tests are the major performance bottleneck and validates the key idea of our work that reducing the number of CI tests reduces the run time.

7 Conclusion and Future Work

In this paper, we presented a novel strategy to incorporate temporal precedence relationships to learn the causal network over event streams. First, we introduced the *Temporal Network Inference* algorithm to model temporal precedence information. Then, we presented the *Fast Causal Network Inference* algorithm to reduce the running time complexity of learning causal network by eliminating unnecessary CI tests. We showed the experiment results to validate our approach by comparing against the state-of-the-art PC algorithm. For the future work, we plan to explore the temporal semantics further for causal network inference over out-of-order event streams.

References

1. Heckerman, D.: A Bayesian approach to learning causal networks. In: UAI, pp. 285–295 (1995)
2. Ellis, B., Wong, W.H.: Learning causal Bayesian network structures from experimental data. J. American Statistics Association 103(482), 778–789 (2008)
3. Li, G., Leong, T.-Y.: Active learning for causal Bayesian network structure with non-symmetrical entropy. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 290–301. Springer, Heidelberg (2009)
4. Meganck, S., Leray, P., Manderick, B.: Learning causal Bayesian networks from observations and experiments: a decision theoretic approach. In: Torra, V., Narukawa, Y., Valls, A., Domingo-Ferrer, J. (eds.) MDAI 2006. LNCS (LNAI), vol. 3885, pp. 58–69. Springer, Heidelberg (2006)
5. Pearl, J.: Causality: Models, Reasoning and Inference, 2nd edn. Cambridge University Press (2009)
6. Spirtes, P., Glymour, C.N., Scheines, R.: Causality from probability. In: ACSS (1990)
7. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. MIT Press (2000)

8. Cheng, J., Greiner, R., Kelly, J., Bell, D., Liu, W.: Learning Bayesian networks from data: an information-theory based approach. Artificial Intelligence 137(1-2), 43–90 (2002)
9. Pearl, J.: Causal diagrams for empirical research. Biometrika 82, 669–688 (1995)
10. Popper, K.: The Logic of Scientific Discovery, Reprint edn. Routledge (October 1992)
11. Chickering, D.M.: Learning equivalence classes of Bayesian-network structures. J. Machine Learning Research 2, 445–498 (2002)
12. de Campos, L.M.: A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. J. Machine Learning Research 7, 2149–2187 (2006)
13. Bishop, Y.M., Fienberg, S.E., Holland, P.W.: Discrete Multivariate Analysis: Theory and Practice. MIT Press (1975)
14. Kullback, S.: Information Theory and Statistics, 2nd edn. Dover Publication (1968)
15. Spirtes, P., Meek, C.: Learning Bayesian networks with discrete variables from data. In: KDD, pp. 294–299 (1995)
16. Pearl, J.: Graphs, causality, and structural equation models. Sociological Methods and Research 27, 226–284 (1998)
17. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. 65(1), 31–78 (2006)
18. Frank, A., Asuncion, A.: UCI machine learning repository (2010)