Information Sciences 261 (2014) 32–51

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Incremental causal network construction over event streams

Saurav Acharya *, Byung Suk Lee

Department of Computer Science, University of Vermont, 33 Colchester Avenue, Burlington, VT 05405, USA

ARTICLE INFO

Article history:
Received 27 January 2013
Received in revised form 9 August 2013
Accepted 6 September 2013
Available online 17 September 2013

Keywords:
Event stream processing
Causal modeling
Bayesian network
Temporal reasoning
Data mining
Data engineering

ABSTRACT

This paper addresses modeling causal relationships over event streams where data are unbounded and hence incremental modeling is required. There is no existing work for incremental causal modeling over event streams. Our approach is based on Popper's three conditions which are generally accepted for inferring causality – temporal precedence of cause over effect, dependency between cause and effect, and elimination of plausible alternatives. We meet these conditions by proposing a novel *incremental causal network construction* algorithm. This algorithm infers causality by learning the temporal precedence relationships using our own new *incremental temporal network construction* algorithm and the dependency by adopting a state of the art incremental Bayesian network construction algorithm called the *Incremental Hill-Climbing Monte Carlo*. Moreover, we provide a mechanism to infer only strong causality, which provides a way to eliminate weak alternatives. This research benefits causal analysis over event streams by providing a novel two layered causal network without the need for prior knowledge. Experiments using synthetic and real datasets demonstrate the efficacy of the proposed algorithm.

1. Introduction

People tend to build their understanding of events in terms of cause and effect, to answer such questions as "What caused the IBM stock to drop by 20% today?" or "What caused the glucose measurement of this diabetic patient to increase all of a sudden?". In recent years, there has been growing need for active systems that can perform such causal analysis in diverse applications such as patient healthcare monitoring, stock market prediction, user activities monitoring and network intrusion detection systems. These applications need to monitor the events continuously and update an appropriate causal model, thereby enabling causal analysis among the events observed so far.

In this paper, we consider the problem of modeling causality over *event streams* (not necessarily real-time) with a focus on constructing a *causal network*. The causal network, a widely accepted graphical structure to represent causal relationships, is an area of active research. All the existing works [4,6,9,15,22,24,26,28] in this area have been done for an environment where a complete dataset is available at once. However, event instances in an event stream are unbounded, and in such a case an *incremental* approach is imperative. Thus, the goal of our work is to model causal relationships in a causal network structure incrementally over event streams. To the best of our knowledge, there exists no work done by others with this objective.

Bayesian networks are in popular use for non-incremental causal modeling [4,9,15,22,24,26]. While the Bayesian network encodes dependencies among all variables, it by itself is not the causal network. First, the causal network strictly requires that the parent of a node is its direct cause, but the Bayesian network does not. Second, two or more Bayesian network structures, called the equivalence classes [7], can represent the same probability distribution and, consequently, the causal direc-

* Corresponding author. Tel.: +1 802 7353422.
E-mail addresses: sacharya@uvm.edu (S. Acharya), bslee@uvm.edu (B.S. Lee).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.09.022

tions between nodes are quite random. There is no technique for alleviating these problems in an event *stream* environment where the entire dataset is not available at any given time.

To overcome this lack of suitable approach for incremental causal modeling over event streams, we propose the *Incremental Causal Network Construction (ICNC)* algorithm. The ICNC algorithm is a hybrid method to incrementally model a causal network, using the concepts and techniques of both temporal precedences and statistical dependencies. Alone, neither dependency nor temporal precedence provides enough clue about causal relationships. The temporal precedence information is learned incrementally in a temporal network with the proposed Incremental Temporal Network Construction (ITNC) algorithm (see Section 5.2) whereas the statistical dependencies are learned incrementally with a state of the art algorithm called the Incremental Hill Climbing Markov Chain (IHCMC) [1–3]. There are a few works [14,23] where temporal precedence information is used to identify causal relationships between variables (see Section 8), but none of them is for constructing causal networks. In our approach, we further provide measures to eliminate confounding causalities that do not indicate strong enough causality. In this regard, our approach supports Popper's three conditions for inferring causality, which are temporal precedence, dependency, and no confounding causality [30].

We model an incremental causal network with a novel two layered network structure. The first layer is a network of event *types* where an edge between two event types reflects the causality relationship observed between them so far in a stream. The second layer is a network of event *instances*. It is a virtual layer in that there is no explicit link between event instances. Instead, each event type in the first layer maintains a list of its instances which are then connected to instances of another event type through a unique relational attribute (more on this in Section 6.1). The motivations for this two layered causal network model are as follows. First, it allows for an incremental modification of the network structure at the event type level in light of new event instances. Second, the idea of a virtual layer makes the model flexible enough to add new or drop old event instances (drop when the volume of event instances grows too much) while maintaining the overall causal relationships at the event type layer. In addition to the structural novelty, the causal network is semantically enriched with the notions of causal strength and causal direction confidence associated with each edge.

We conduct experiments to evaluate the performance of the proposed *ICNC* algorithm using both synthetic and real datasets. The experiments measure how closely the constructed causal network resembles the true target causal network. Specifically, we compare the Bayesian network produced by IHCMC and the causal network produced by ICNC against a target network. The results show considerable improvements in the accuracy of the causal network over the Bayesian network by the use of temporal precedence relationship between events.

The contributions of this paper are summarized as follows.

- It presents a temporal network structure to represent temporal precedence relationships between event types and proposes an algorithm to construct a temporal network incrementally over event streams.
- It introduces a two-layered causal network with rich causality semantics, and proposes an incremental causal network construction algorithm over event streams. The novelty of the algorithm is in combining temporal precedence and statistical dependency of causality to construct a causal network.
- It empirically demonstrates the advantages of the proposed algorithm in terms of how the temporal network increases the accuracy of the causal network and how close the generated causal network is to the true unknown target causal network.

The rest of the paper is organized as follows. Section 2 presents some preliminary concepts. Section 3 formulates the specific problem addressed in this paper and outlines the proposed approach. Section 4 describes the incremental Bayesian network construction. Sections 5 and 6 propose the incremental temporal network construction and the incremental causal network construction, respectively. Section 7 evaluates the proposed ICNC algorithm. Section 8 discusses related work. Section 9 concludes the paper and suggests future work.

2. Preliminaries

In this section, we present some key concepts needed to understand the rest of the paper. The concepts are illustrated with a representative use case – diabetic patient monitoring system [10]. We select a few important attributes from this real-world case to make the explanations intuitive, and use them in a running example throughout the paper.

2.1. Event stream, instance, type

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. 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. We support concurrent events. In this paper an event instance is often called simply an event or an instance if the context makes it clear.

Each event instance is created by one event owner. An event type can have many instances, and an event owner can create many instances of any type. 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 Example 1, patient ID may be the CRA, as the events of the same patient are causally related.

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 ID.

Example 1. Consider a diabetic patient monitoring system in a hospital. There are hundreds of patients admitted to a hospital, and a majority of the actions are related to clinical tests and measurements. Each patient is uniquely identifiable, and each test or action of each patient makes one event instance. For example, a patient is admitted to the hospital, has blood pressure and glucose level measured, and takes medication, creating the instances of the above event types as a result. These instances are repeated for a couple of weeks or months till the patient is discharged. Typical event types from these actions would include blood-glucose-measurement-decreased (BGMD), blood-glucose-measurement-increased (BGMI), NPH-insulin-dose-given (NIDG), regular-insulin-dose-given (RIDG), and hypoglycemic-symptoms-exists (HSE), etc. (more in Table 3 in Section 7.1.2).

An event type has the following schema: [type ID, type name, event container], where type ID is the primary key and event container is a list of all instances of the type. An event instance has the following schema: [type ID, CRA, timestamp, lifetime, attribute container], where type ID, CRA and timestamp together make the primary key, lifetime is the time duration up to which the event is alive, and attribute container is the set of attribute-value pairs storing any additional information. Note that an event stream contains an indefinitely large number of event instances; hence, an event container, with limited space, cannot store all of them, and therefore an event instance is removed from the event container once its lifetime expires.

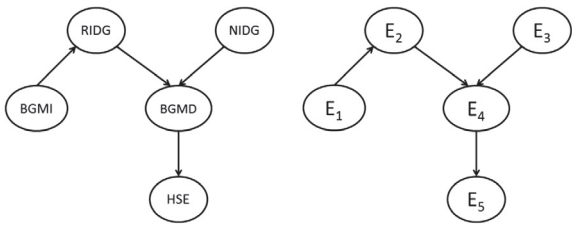
2.2. Causality and causal network

Causality (or causal relationship) is a relationship between a cause and an effect. An event can have multiple cause events; similarly, it can have multiple effect events. The conceptual basis of causality in our work is that the effect is dependent on the cause to occur and the cause must precede the effect. More specifically, we use the following notion of causality.

Definition 1 (*Causality*). An event type E_i is a cause of another event type E_j ($i \neq j$) if a majority of instances of E_i and a majority of instances of E_j are dependent and a majority of instances of E_i precede a majority of instances of E_j. (The specifics of how many constitute a "majority" is application-dependent.) In addition, an event instance e_{ki} is said to be a cause of another event instance e_{kj} ($i \neq j$) if they have causality at the event type level and e_{ki} precedes e_{kj}. Note that these two instances share the same CRA (k).

Causal network is a popularly used data structure for representing causality [4,9,15,22,26]. It is a directed acyclic graph with a strict requirement that, for every directed edge $\langle u, v \rangle$, the parent node u is a *direct* cause of the child node v. We add to this the temporal ordering, i.e., u should precede v, as another requirement.

Fig. 1(a) illustrates a causal network of the five event types mentioned in Example 1. The intuitions of the causality among them are as follows – (1) an insulin dose is given to a patient (RIDG or NIDG) to decrease the blood glucose level (BGMD), hence the edge from RIDG and NIDG to BGMD; (2) an increasing blood glucose level (BGMI) triggers the administration of a regular insulin dose (RIDG), hence the edge from BGMI to RIDG; and (3) it is common medical knowledge that a decrease in blood glucose level (BGMD) can cause hypoglycemic symptom (HSE), hence the edge from BGMD to HSE. (From here on, we denote BGMI, RIDG, NIDG, BGMD, and HSE with E_1, E_2, E_3, E_4, and E_5, respectively, as shown in Fig. 1(b).)



(a) Network with the actual event type names. (b) Network with the simplified event type notations.

Fig. 1. Causal network for Example 1.

3. Problem formulation and the proposed approach

3.1. Problem

There are three issues that constitute the problem addressed in this paper. First, due to the existence of equivalence classes, the directions of edges in the Bayesian network are not reliable and prone to being incorrect. Therefore, we need a method to learn the correct directions of causal relationships and thereby reduce the number of reversed edges in the causal network. Second, there may be a number of spurious or missing causal relationships in a causal network.[1] The number of spurious edges and the number of missing edges are competing factors bringing a tradeoff in the accuracy of the resulting causal network. Thus, an optimal causal network should have the minimum total number of spurious and missing causal relationships. Third, an event stream is unbounded. Unlike the existing works [4,9,15,22,26] where a complete dataset is expected to be available for causal modeling, we need an algorithm that constructs a causal network incrementally. In summary, the problem addressed is to construct a causal network incrementally over event streams and while doing so, to reduce the number of reversed edges and minimize the total number of missing and spurious edges in the resultant causal network.

3.2. Overview of the approach

To learn a causal network incrementally over event streams, we propose the Incremental Causal Network Construction (ICNC) algorithm (see Section 6.1) which models the causal relationships in a two layered network. In the algorithm, we meet Popper's three conditions for inferring causality – temporal precedence, dependency and no confounding causality [30]. We propose the *Incremental Temporal Network Construction (ITNC)* algorithm (see Section 5) to model the temporal precedences from an event stream into a temporal network incrementally. As mentioned earlier, Bayesian network is reliable to model statistical dependencies and thus we adopt an incremental Bayesian network construction algorithm, the *Incremental Hill-Climbing Monte Carlo* (*IHCMC*) algorithm (see Section 4.2). To resolve the equivalence class problem in Bayesian networks, we use the *DAG-to-CPDAG* algorithm [7] to generate a complete partial directed acyclic graph (CPDAG) where unreliable edges are rendered undirected. Then, the ICNC algorithm integrates the temporal network and the CPDAG. During the integration, spurious edges which do not indicate strong enough causality are removed and edges which indicate strong evidence of causality are added with the aim of minimizing the total number of missing and spurious edges, respectively. We rely on the temporal precedence information in the temporal network to identify the correct causal directions of the undirected edges. Section 6 describes the rule for this integration and the algorithm for constructing the incremental causal network.

4. Incremental Bayesian network

4.1. Bayesian network model

Bayesian network is a directed acyclic graph which encodes a joint probability distribution over a set of random variables. The joint probability distribution of a set of n variables $X \equiv \{X_1, \ldots, X_n\}$ is specified as

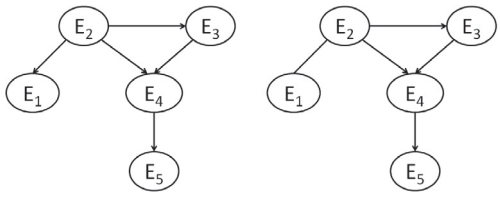
$$P(X) = \prod_{i=1}^{n} P(X_i|Pa_i)$$

where Pa_i is the set of parent nodes of the variable X_i.

Bayesian network encodes the assertions of conditional independence between variables and, thus, is an appropriate network structure to represent statistical dependency between events. Fig. 2(a) (in Section 4.3) shows the Bayesian network corresponding to Example 1.

As already mentioned, however, the same probability distribution can be represented by different Bayesian network structures that are in the same equivalence class [7]. For example in Fig. 2(a), $E_1 \rightarrow E_2 \rightarrow E_3$ and $E_1 \leftarrow E_2 \rightarrow E_3$ may represent the same joint probability distribution and hence the same network topology. The differences in their network structures are in the directions of the edges. Since the edge direction in a causal network indicates the causal direction between events, the causal meaning becomes entirely different with the change in the edge direction. This ambiguity makes Bayesian networks unsuitable to be used as causal networks. In the example above, the causal direction $E_1 \rightarrow E_2$ (recall E_1 and E_2 are BGMI and RIDG, respectively) makes sense as an increase in blood glucose measurement (BGMI) causes the use of insulin (RIDG). However, the reverse causal direction $E_1 \leftarrow E_2$ (in the Bayesian network of Fig. 2(a)) is incorrect as the use of insulin does not increase the blood glucose measurement. So, a Bayesian network cannot be trusted to detect the causal direction between events.

[1] The causal network has a strict requirement that the parents of a node are its direct causes. However, the same is not true for a Bayesian network, and, therefore, dependencies between event types detected in a Bayesian network may not be causal relationships. Such dependencies lead to spurious causal relationships in the causal network.



(a) Input Bayesian Network (b) Output CPDAG

Fig. 2. Illustration of DAG-to-CPDAG. *Source:* Chickering [7].

Bayesian variables in the constructed network represent event types. They are boolean variables indicating whether an instance of the represented event type exists or not in the event stream.

4.2. Incremental Bayesian network construction algorithm

With the continuous arrival of an event stream with no definite end, it is imperative to construct a Bayesian network incrementally by refining the existing network every time a new batch of data becomes available. Discarding and reconstructing the entire network from scratch every time would be too expensive. As already mentioned, we use the Incremental Hill-Climbing Monte Carlo (IHCMC), an incremental version of the *HCMC* algorithm [1,2]. In this subsection we summarize the HCMC and the IHCMC algorithms implemented based on Alcobé's work [1,2].

Algorithm 1. HCMC

Require: a dataset D on $\{X_1,\ldots,X_n\}$ variables, a variable *ncr* indicating whether to use NCR neighborhood or NR neighborhood, a constant MAXTRIALS, a positive integer n

localMaximum = false, trials = 0;
Let G be an edgeless DAG.
Calculate the initial neighborhood $N(G)$ (based on *ncr*) and sufficient statistics for D;
while localMaximum is *false* **do**
  Reverse n randomly chosen edges in G; {*//Escape from the local maximum to find the global maximum by randomly reversing edges.*}
  Let G' be the highest-score DAG in the neighborhood $N(G)$. {*//Select the best network structure in the neighborhood.*}
  if score $(G) \geqslant$ score (G') then localMaximum = true; {*//There are no network structures with higher score than the current one.*}
  if localMaximum is false {*//The local maximum has not been reached.*} **then**
    trials = 0;
    $G = G'$;
  else if trials < MAXTRIALS {*//Repeat the trial up to MAXTRIALS times to find the global maximum.*} **then**
    trials = trials + 1;
    localMaximum = false;
  end if
end while

The HCMC algorithm begins with a network with no edge and, at each iteration, enumerates the neighboring states of the current network state (in the search space) by randomly adding, removing, or reversing edges, and then keeps the network with the highest score. The algorithm terminates when none of the neighboring networks improves the score over the current network. Algorithm 1 summarizes the implemented HCMC algorithm. In the algorithm, the NR (no reversal) neighborhood refers to all DAGs with one arc more or less and do not introduce a directed cycle, whereas the NCR (non-covered reversal) neightborhood refers to the NR neighborhood plus all DAGs with one non-covered edge reversed and does not introduce a directed cycle. (An edge $x \rightarrow y$ in a DAG G is said to be covered in G if $parents(x) \cup x = parents(y)$, that is, all and only the parents of x are the parents of y).

Algorithm 2. IHCMC

Require a new dataset D' on $\{X_1,\ldots,X_n\}$ variables, the highest score network structure G in the old dataset D, the set C_{ij} of candidate parents of each variable X_i and parent X_j from the previous learning step, the ordered set O of operators performed in the previous learning step, and the number n of operators to store in candidate lists. {*In the first run, G is an edgeless network and C_{ij} and O are empty.*}
1: Update the sufficient statistics of the old dataset D to reflect the new dataset D';
2: localMaximum = false; k = 0; run = true;
3: $G' = G$; {*//Start with the network structure from the old dataset.*}
4: Calculate neighborhood $N(G')$ with the operators in O;
5: **while** run is *true* and $k < |O|$ **do**
6: Select the operator, o, that maximizes the score of networks in $N(G')$;
{*//Find the best network from the neighborhood N(G').*}
7: run = false;
8: **if** o is present within the window of operators being considered in O **then**
9: Revise G' by applying the operator o;
10: Calculate neighborhood $N(G')$ with the operators in O;
11: $k = k + 1$;
12: run = true;
13: **end if**
14: **end while**
15: Call the algorithm HCMC with G', sufficient statistics, and the n best candidates in C_{ij};

Initially, the IHCMC algorithm behaves exactly like the HCMC algorithm, which finds a network structure achieving the highest score given the provided data. During this process, the order in which the operators (i.e., add, delete, reverse) coupled with edges are applied is stored for use in the next learning step. Then, with the arrival of a new dataset, the IHCMC algorithm updates the "sufficient statistics" of the old dataset to reflects the new data. (*Sufficient statistics* is a statistical summary of the dataset which contains all information necessary to calculate the scores of the Bayesian network.) Using the new sufficient statistics and the order of operations from the previous step, it determines whether the current network structure should be revised for the new dataset. If a revision is needed, the IHCMC algorithm starts with the highest-score network in the previous step as the initial model. This approach makes the algorithm efficient. When updating the network for the new dataset, IHCMC reduces the search space by restricting the set of operators and edges considered. In order to do that, a certain (used-defined) number of pairs of operators and edges that give the score closest to the best one are stored for the next learning step, and the search is restricted to only the neighboring networks obtained with these stored pairs. At the end, IHCMC calls the HCMC algorithm so that it continues to build the network structure in light of the new data. Algorithm 2 summarizes the implemented IHCMC algorithm.

4.3. DAG-to-CPDAG algorithm

As mentioned earlier, we use the DAG-to-CPDAG algorithm to find the edges with ambiguous direction in a Bayesian network. The algorithm takes a Bayesian network as the input and outputs a completed partial directed acyclic graph (CPDAG) representation of the equivalence class to which that structure belongs. Undirected edges in the CPDAG are the ambiguous edges. We particularly use the implementation proposed by [7]. Algorithm 3 outlines the three steps in the algorithm. First, it performs a topological sort on the vertices in the input Bayesian network so that, for any pair of vertices x and y, x must precede y if x is an ancestor of y. Second, based on the topological sorting, the edges are sorted first in the ascending order of the incident vertices and then in the descending order of the outgoing vertices. Finally, the ordered edges are labeled either "compelled" or "reversible". The "reversible" edges are made undirected while preserving the edge direction of the "compelled" edges in the final CPDAG.

Algorithm 3. DAG-to-CPDAG

Require BayesianNetwork G.
1: Sort the vertices in G such that x precedes y if and only if x is an ancestor of y;
2: Based on the topological order of the vertices, sort the edges in G, first in the ascending order of the incident vertices and then in the descending order of the outgoing vertices;
3: Label every edge in G as "unknown";
4: **while** there exists an edge labeled "unknown" in G

(*continued on next page*)

```
5:    Select the edge of the lowest order, x → y, that is labeled "unknown";
6:    run = true;
7:    for every edge z → x labeled "compelled" do
8:      if z is not a parent of y {//If the path is z → x → y} then
9:        Label x → y and every edge incident into y as "compelled";
10:       run = false;
11:       End this FOR Loop;
12:     else
13:       {//If the path is z → x,z → y.}
14:       Label z → y as "compelled";
15:     end if
16:   end for
17:   if run is true then
18:     if there exists an edge w → y such that w ≠ x and w is not a parent of x then
19:       Label x → y and every "unknown" edge incident to y as "compelled";
20:     else
21;       Label x → y and every "unknown" edge incident to y as "reversible";
22:     end if
23:   end if
24: end while
25: Make all "reversible" edges undirected;
```

Fig. 2 illustrates the Chickering's algorithm [7]. Given the input Bayesian network in Fig. 2(a), the first step (topological sorting of the vertices) gives E_2, E_3, E_1, E_4, and E_5; then, the second step sorts the edges in the following order: $E_2 \rightarrow E_3$, $E_2 \rightarrow E_4$, $E_2 \rightarrow E_1$, $E_3 \rightarrow E_4$, and $E_4 \rightarrow E_5$; finally, the third step labels the edge $E_2 \rightarrow E_1$ as "reversible" and the remaining edges as "compelled" and, therefore, the "reversible" edge $E_2 \rightarrow E_1$ is made undirected, resulting in the final output CPDAG shown in Fig. 2(b).

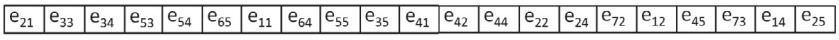
Note that the Chickering's algorithm does not check repeatedly on all edges, hence computationally more efficient than previous rule-based algorithms [25,29]. In these algorithms, the idea is to undirect every edge in a DAG, except for those edges that participate in a v-structure. (Three nodes *x*, *y* and *z* are said to have a v-structure if their edges form the structure $x \rightarrow y \leftarrow z$.) The rules are applied repeatedly on every edge to determine the edge direction until no rule has any effect on the PDAG, that is, no edge becomes undirected.

5. Incremental temporal network

5.1. Temporal network model

In this paper the temporal network models the temporal precedences between pairs of events. It is a directed acyclic graph of nodes representing event types. Its construction has to be incremental as well, as it is over a stream. For this purpose, we place a *window* over the stream. The semantics of the window can be dependent on the application, and we use a time-based window here without loss of generality. Typically, the application offers a natural observation period (e.g., day) that makes a window.

As mentioned in Definition 1, causality is defined between events with the same *common relational attribute* (*CRA*). So, we arrange the events in a window by CRA as they arrive, producing a *partitioned* window as a result. Events in the same partition have the same CRA and are ordered by the timestamp. Fig. 3 illustrates it with an event stream from the patient diabetes monitoring system described in Example 1.



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

(b) Events in the window partitioned by CRA.

Fig. 3. Partitioned window of events.

We assume events farther apart temporally are less likely to have a causal relationship between them. So, in a partitioned window, we observe temporal relationships only between events that are near to each other, that is, at most k instances apart. (We refer to k as the *adjacency span* of events.) For example, given a sequence of events $\{e_{i1}, e_{i2}, e_{i3}, e_{i4}\}$, and k specified as 3, we observe the frequencies of event pairs $\{e_{i1}, e_{i2}\}$, $\{e_{i1}, e_{i3}\}$, $\{e_{i1}, e_{i4}\}$, $\{e_{i2}, e_{i3}\}$, $\{e_{i2}, e_{i4}\}$, and $\{e_{i3}, e_{i4}\}$. These events instances have the same CRA (subscript i) but belong to different event types (E_1, E_2, E_3 and E_4).

With the arrival of a new batch of event instances, we augment each partition in the new window by prefixing it with the last k instances 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 that the observation of an instance of E_j following an instance of E_i is made frequently enough. So, we use the measure *temporal strength*, as defined below.

Definition 2 (*Temporal strength*). Consider an edge $E_i \rightarrow E_j$ ($i \neq j$) in a temporal network. 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

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

where N_{ET} is the number of event types. □

There are two relevant issues in selecting the edges in a temporal network. First, a *temporal strength threshold* (δ_{s1}) should be provided. Only those edges whose temporal strength is greater than δ_{s1} are included in the temporal network. (There is another threshold, δ_{s2} ($>\delta_{s1}$) used in our work. The temporal strength of an edge higher than δ_{s2} indicates even higher probability of the edge representing a causal relationship (see Definition 1). In Section 6, we use δ_{s2} to add a causal relationship missing in the Bayesian network.) Second, a criterion should be set to handle a case in which both an edge and its reverse edge have temporal strengths higher than δ_{s1}. For this, we use a *gap threshold* (δ_g) which makes sure that when the stronger edge direction is selected, the difference between the two opposite temporal strengths is significant enough. For example, an edge $E_i \rightarrow E_j$ (with frequency f_{ij}) is selected instead of its reverse edge $E_j \rightarrow E_i$ (with frequency f_{ji}) if and only if $\frac{f_{ij}-f_{ji}}{f_{ij}+f_{ji}} > \delta_g$.

5.2. Incremental temporal network construction algorithm

The idea behind the *Incremental Temporal Network Construction* (*ITNC*) 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.

Algorithm 4. Incremental temporal network construction

Require: *window W*, event adjacency span (k), gap threshold (δ_g), temporal strength threshold (δ_{s1}), attenuation constant τ, an edgeless network structure *TN*, *attenuated frequency matrix (AFM)*.

Let B_p and B_c be two empty buffers (used to store "parent" events and "child" events, respectively).
for each partition P (corresponding to CRA a) in W {*//Consider one partition at a time as the events of two partitions are unrelated and therefore independent of each other.*} **do**
  for each unique ith timestamp t_i in P **do**
    Clear B_p and Insert all events with timestamp t_i into B_p;
    for $d = 1$ to k such that $i + d \leqslant$ sizeOf (P) {*//Iterate over all succeeding events within the adjacency span "k" in the same partition.*} **do**
      Clear B_c and Insert all events with timestamp t_{i+d} into B_c;
      for each event instance e_{ac} and e_{ap} in B_c and B_p, respectively **do**
      if type (e_{ac}) $\neq$type (e_{ap}) {*//There cannot be causal relationships between events of the same type.*} **then**
        Increase the frequency of element $f_{type(e_{ap}),type(e_{ac})}$ in *AFM* by $e^{-(d-1)\tau}$;
      end if
    end for
  end for
end for
Let *SM* be an empty strength matrix.
for each pair of elements f_{ij} and f_{ji} in *AFM* **do**
  //Calculate the temporal strength of only those edges of which the temporal precedence directions are not ambiguous.

(*continued on next page*)

18: Calculate the evidence of temporal edge direction, $d_{ij} = \frac{f_{ij}-f_{ji}}{f_{ij}+f_{ji}}$;
19: **if** the value of $|d_{ij}|$ is greater than δ_g **then**
20: **if** d_{ij} is positive (i.e., the evidence of f_{ij} is greater than f_{ji}) **then**
21: Calculate S_{ij} (see Eq. (1)) and set S_{ji} to 0;
22: **else if** d_{ij} is negative (i.e., the evidence of f_{ji} is greater than f_{ij}) **then**
23: Calculate S_{ji} (see Eq. (1)) and set S_{ij} to 0;
24: **end if**
25: **end if**
26: **end for**
27: **for** each pair of elements s_{ij} and s_{ji} in *SM* **do**
28: *//Add only those edges whose temporal strengths are greater than* δ_{s1}.
29: **if** $s_{ij} > \delta_{s1}$ **then**
30: Add an edge $E_i \rightarrow E_j$ in *TN*;
31: **else if** $s_{ji} > \delta_{s1}$ **then**
32: Add an edge $E_j \rightarrow E_i$ in *TN*;
33: **end if**
34: **if** an edge is added and it introduces cycle in *TN* **then** remove the edge with the lowest temporal strength (in *SM*) in the cycle;
35: **end for**

The algorithm has three steps. The overall algorithm is centered on a *attenuated frequency matrix*, which is initially empty (i.e., all zero elements) and updated with each new batch of events.

1. Update the attenuated frequency matrix *AFM* by observing the precedence relationships of event pairs within the adjacency span in the partitioned window (see lines 1–14 of Algorithm 4). An element f_{ij} in *AFM* reflects the total number of times events of type E_i precede events of type E_j ($i \neq j$). Each time we observe an event pair (e_{oi}, e_{oj}) in the event stream such that e_{oi} precedes e_{oj}, we increase the value of f_{ij} by $e^{-d\tau}$ where d is the distance between the two events and $\tau \in (0,1)$ is an attenuation constant. The rationale for the attenuation is that, as the distance between events increases, the probability of them being cause and effect decreases.
2. Calculate the temporal strength of each edge in *AFM* and store it in a *strength matrix SM* (see lines 15–26 Algorithm 4). For each pair of an edge and its reversed edge, set the strength of the edge with the lower frequency to zero. The calculated strength of the selected edge, e.g., $E_i \rightarrow E_j$, is stored in the element s_{ij} of *SM*.
3. Determine the edges of the temporal network using the strength matrix (see lines 27–35 of Algorithm 4). Only those edges whose temporal strengths are greater than the strength threshold δ_{s1} are added. If a cycle is introduced, we remove the edge with the lowest temporal strength in the cycle.

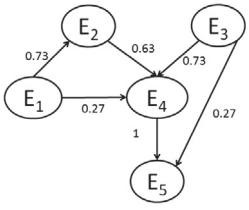
Let us illustrate the *ITNC* algorithm considering the event stream shown in Fig. 3. Suppose the adjacency span (k) and the attenuation constant (τ) are set to 2 and 0.5, respectively. Then, in the first step, Algorithm 4 (lines 1–14) constructs an attenuated frequency matrix shown in Fig. 4(a) from the event stream. In addition, suppose the gap threshold (δ_g) is set to 10%. Then, in the second step, Algorithm 4 (lines 15–26) constructs a strength matrix shown in Fig. 4(b). Note that the edge

$$\begin{bmatrix} 0 & 3 & 0 & 1.104 & 0 \\ 0 & 0 & 1 & 3 & 0.736 \\ 0 & 0 & 0 & 2 & 0.736 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(a) Attenuated frequency matrix

$$\begin{bmatrix} 0 & 0.73 & 0 & 0.27 & 0 \\ 0 & 0 & 0.21 & 0.63 & 0.16 \\ 0 & 0 & 0 & 0.73 & 0.27 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Strength matrix



(c) Output temporal network

Fig. 4. Illustration of temporal network construction from the event stream in Fig. 3.

$E_5 \rightarrow E_4$ fails the gap threshold test and, therefore, its strength is set to 0. In addition, suppose the temporal strength threshold (δ_{s1}) is set to 25%. Then, in the third step, Algorithm 4 (lines 27–35) constructs from the strength matrix the temporal network shown below in Fig. 4. Note that the strengths of the edges $E_2 \rightarrow E_3$ and $E_2 \rightarrow E_5$ (21% and 16%, respectively) are lower than the temporal strength threshold and, consequently, are pruned out.

The computational complexity of the ITNC algorithm for a temporal network is polynomial. Let n and n_{cra} be the number of event types and the number of CRAs, respectively. As explained earlier in Section 5.1, a partitioned window has n_{cra} partitions. For a new batch of events, the step 1 (attenuated frequency matrix construction) of the ITNC algorithm goes through all of n_{cra} partitions to calculate the attenuated frequency. For each partition P, we find the temporal precedence relationship between each event at a unique timestamp and another event at a later timestamp (within the k adjacency span). The maximum number of events that can occur at any timestamp is n, and the maximum value of k is the size of the partition, $|P|$; so, for one partition the worst case running time is $O(|P| \cdot k \cdot n^2)$, which equals $O(|P|^2 \cdot n^2)$ since $k \leqslant |P|$. Thus, the running time of the step 1, for n_{cra} partitions, is $O(|P|^2 \cdot n^2 \cdot n_{cra})$. In the step 2 (strength matrix construction), the running time is $O(n^2)$ as the algorithm iterates $n^2/2$ times on the frequency matrix. The step 3 (temporal network construction) iterates $n^2/2$ times on the strength matrix. At each iteration, the algorithm checks for a cycle if an edge is added to the network. In the worst case, every edge in the network is inspected for a cycle to be detected and there are at most $n \cdot (n-1)/2$ edges in the network. So, the running time of the step 3 is $O(n^4)$. Hence, the total running time for the ITNC algorithm is $O(|P|^2 \cdot n^2 \cdot n_{cra} + n^2 + n^4)$ which equals $O(n^2 \cdot (|P|^2 \cdot n_{cra} + n^2))$.

6. Incremental causal network

6.1. Causal network model

As mentioned earlier, we propose to organize the causal network in two layers for compact and yet versatile causal modeling. Representing causality of an unbounded event stream in a single causal network is not only complex but also raises challenges in maintaining the network with the arrival of new events. So, we prefer a simple network that should be able to represent causality at both the event type level and the event instance level. In the proposed causal network model, the first layer is a network of event types. That means, several thousands of events are aggregated to several dozens of event types, which gives general causal relationships for a majority of the events in the event stream and greatly reduces the size and complexity of the network. The second layer has event instances, and it holds specific causal relationships among them. The event instances are stored in the *event container* of an event type. So, the event instances assume the causal relationship of their event types and an event instance of one event type is causally related to an event instance of another event type through a uniquely identifiable CRA. The temporal precedence relationships between events play a key role in identifying their causal relationship. In short, an event $e_{a_1 i}$ is the cause of another event $e_{a_2 j}$ if and only if (a) they share the same CRA (i.e., $a_1 = a_2$), (b) there exists an edge from E_i to E_j ($i \neq j$) in the causal network (at the type level), and (c) $e_{a_1 i}$ precedes $e_{a_2 j}$ within the adjacency span.

Fig. 5 illustrates the causal network structure, for the event stream shown in Fig. 3. The causality among event types are modeled in the first layer. Both E_2 and E_3 are the direct causes of E_4, while E_1 causes E_2 and E_4 causes E_5. The second (virtual) layer holds the causal relationships between event instances. For example, suppose the adjacency span is 2. Then, the event e_{33} is a cause of the event e_{34}, as there is an edge from E_3 to E_4 and e_{33} precedes e_{34} within the adjacency span of the same partition (i.e., under the same *CRA*) (see Fig. 3); on the other hand, e_{72} does not cause e_{73} even though e_{72} immediately precedes e_{73}, as there is no edge from E_2 to E_3; similarly, e_{42} does not cause e_{45}, as there is no path between them.

Note that the temporal strength gives a measure of the temporal precedence between event types, that is, how often the instances of an event type occur after the instances of another event type. What is deemed more appropriate for causality, however, is a probabilistic measure which determines the strength among all likely causes of an event type. This measure is the *causal strength* defined below. The highest causal strength gives the most likely cause of an event type.

Definition 3 (*Causal strength*). Consider an edge $E_i \rightarrow E_j$ ($i \neq j$) in the causal network and its *frequency* f_{ij} in the temporal network. Then, we define the *causal strength*, c_{ij}, of the edge $E_i \rightarrow E_j$ as

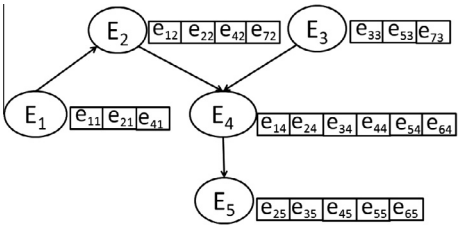


Fig. 5. Two-layered causal network.

$$c_{ij} \triangleq \frac{f_{ij}}{\sum_{k \in parents(E_j)} f_{kj}}$$

In addition to the causal strength, we need a measure of the confidence in the causal direction of an edge.

Definition 4 (*Causal direction confidence*). Consider an edge $E_i \rightarrow E_j$ $(i \neq j)$ in the causal network and its *frequency* f_{ij} in the temporal network. If its reversed edge has a *frequency* f_{ji}, then we define the *causal direction confidence*, d_{ij}, of the edge $E_i \rightarrow E_j$ as

$$d_{ij} \triangleq \frac{|f_{ij} - f_{ji}|}{f_{ij} + f_{ji}}$$

Based on the notion of causality in Definition 1, now we propose the rule for merging a temporal network and a Bayesian network.

Rule 1. [Temporal and Bayesian network integration]Given two event types, we add an edge in the causal network if there exists an edge between them in both the Bayesian network and the temporal network. Moreover, if the temporal strength of an edge in the temporal network is greater than the higher threshold value δ_{s2}, then the edge is added to the causal network even if it does not exist in the Bayesian network. If there are any undirected edges in the Bayesian network (as a result of DAG-to-CPDAG), the direction is set as in the temporal network.

Note that an edge direction in the Bayesian network can be ambiguous due to the existence of equivalence classes. Note as well that an edge can be removed or added in the causal network depending on the values of the temporal strength thresholds $(\delta_{s1}, \delta_{s2})$ and the gap threshold (δ_g).

6.2. Incremental causal network construction algorithm

Algorithm 5. Incremental causal network construction

Require: event stream, a list of event types, temporal strength threshold δ_{s2}
1: Construct an edgeless causal network *CN* with the nodes representing the given event types;
2: **for** each batch in the event stream **do**
3: Run the IHCMC algorithm to update the Bayesian network *BN'* and then run the *DAG-to-CPDAG* algorithm [7] on *BN'* to obtain *BN*; *//DAG-TO-CPDAG removes the edge direction of ambiguous edges.*
4: Run the ITNC algorithm to update the temporal network *TN*;
5: *// Merge TN and BN using Rule 1.*
6: **for** each pair of event types E_i and E_j $(i \neq j)$ **do**
7: **if** there is an edge in both BN and TN **then**
8: Add an edge in CN; set its direction as in BN if the edge in BN is directed or as in TN if undirected;
9: Calculate the causal strength as in Definition 3 and the causal direction confidence as in Definition 4;
10: **else if** there is no edge in BN but an edge in TN and its temporal strength is greater than δ_{s2} **then**
11: Add an edge in CN and set its direction as in TN;
12: **else**
13: Add no edge in CN;
14: **end if**
15: **if** an edge is added and it introduces a cycle in *CN* **then**
16: Remove the edge with the lowest causal strength in the cycle;
17: **end if**
18: **end for**
19: **end for**

The algorithm requires an event stream, an observation period (during which a new set of events are collected from the stream) and a list of event types. Initially, an edgeless causal network structure of the given number of nodes is built. Then it is updated incrementally as soon as a new batch of events arrives, as outlined in Algorithm 5.

Recall that the Bayesian network is reliable in judging the existence of statistical dependency between two event types but not the direction of the dependency. So, the key idea of the algorithm is to rely on the statistical dependency given by the Bayesian network and the causal direction given by the temporal network. In addition, the temporal precedence relationships can provide important clues for spurious edge removal or missing edge addition. The algorithm adopts the IHCMC algorithm which finds incrementally a Bayesian network structure achieving the highest score. The Bayesian network thus

obtained, however, is one of many possible equivalent network structures and so the edge directions are not reliable. Thus, we run the DAG-to-CPDAG algorithm [7] which may render some edges undirected if it regards them ambiguous. The temporal network is constructed using the ITNC algorithm (see Section 5). The remaining key task is to integrate them according to Rule 1 as they are updated incrementally with a new batch of dataset. If a cycle is introduced in the causal network, we remove the edge with the lowest causal strength in the cycle.

It is well known that the learning of Bayesian network structure, i.e., IHCMC in our case, is a NP hard problem [8]. Since the computational complexity of the ITNC algorithm is polynomial (as shown in Section 5.2), the computational complexity of the ICNC algorithm is governed by the computational complexity of the IHCMC algorithm.

7. Performance evaluation

We conduct experiments to evaluate the proposed ICNC algorithm against the IHCMC algorithm. The main focus of the evaluation is on the resultant network structure. One evaluation is with respect to the topologies of the resulting networks, and the other evaluation is with respect to the edge directions. In both cases, the networks generated by ICNC and IHMC are compared against a true causal network unknown to the algorithms. In addition, the run-time overhead of ICNC is evaluated against IHCMC. Section 7.1 describes the experiment setup, including the evaluation metrics, datasets and the platform used, and Section 7.2 presents the experiment results.

7.1. Experiment setup

7.1.1. Evaluation metrics

Intuitively, the performances of causal network construction 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 Tsamardinos et al. [31] 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 relative number of the erroneous edges of each kind with respect to the maximum possible number of erroneous edges of that kind as the evaluation metric here. The maximum number of reversed or missing edges is the actual number of edges in the target causal network. On the other hand, the maximum number of spurious edges is given as $\frac{N_{ET}(N_{ET}-1)}{2} - N_{edges}$, where N_{ET} and N_{edges} are the number of nodes (=number of event types) and the number of edges, respectively, in the target network.

7.1.2. Datasets

Experiments are conducted using both synthetic and real datasets.

7.1.2.1. Synthetic datasets. A synthetic dataset is reverse-engineered from a target causal network. Given control parameters in Table 1, 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. Specifically, there are three steps. First, N_{ET} nodes are created and edges are added randomly, and random conditional probabilities are assigned to each edge. Each node can have up to Max_{NC} edges from cause nodes and up to Max_{NE} edges to effect nodes. (We set both Max_{NC} and Max_{NE} to 3 for the experiments presented here.) Second, a joint probability distribution (JPD) table is built from the conditional probabilities assigned to edges of the target causal network. The rows of the JPD table collectively cover all event sequences possible, while each row has its own probability. Third, the probability for each row in the JPD table is multiplied by N_O to calculate the number of repetitions of that event sequence in the dataset. We assume that the event owner is the *CRA* for the dataset.

The size of a JPD table grows exponentially with N_{ET} and therefore we use parallel processing for the event stream generation. The JPD table is divided into multiple partitions and the dataset is created by running parallel processes over each of these partitions. The dataset is thus represented by a collection of files in which the events are shuffled according to the owner ID while preserving the temporal order.

There are five cases of datasets, DS1 through DS5, according to the number of nodes in the represented target causal networks (see their profiles in Table 2). The target causal networks have 4, 8, 12, 16 and 20 nodes, respectively, They are created

Table 1
Control parameters for synthetic event stream generation.

Parameter	Meaning
N_O	Number of event owners (with unique ID)
N_{ET}	Number of event types (i.e., nodes)
Max_{NC}	Maximum number of cause events (parents)
Max_{NE}	Maximum number of effect events (children)

Table 2
Profiles of the five synthetic datasets.

Dataset	N_{ET}	N_{edges}	N_O	$N_{instances}$
DS1	4	4	5000	15,128
DS2	8	15	30,000	124,475
DS3	12	22	500,000	3,173,246
DS4	16	39	6,553,600	50,247,293
DS5	20	49	52,428,800	510,971,687

N_{edges} is the number of *actual* edges in the network. $N_{instances}$ is the average number of event instances in the datasets of each case.

with 1, 2, 16, 64 and 512 parallel processes, respectively, thus consisting of 1, 2, 16, 64 and 512 files, respectively. Each case has 100 different datasets. So, there are a total of 500 different synthetic datasets representing 500 random causal networks. Each row of a synthetic dataset represents one event instance with the schema [type ID, CRA, timestamp, lifetime, attribute container] as discussed in Section 2.1.

7.1.2.2. Real dataset. The real dataset contains diabetes lab test results [10] of 70 different patients over a period ranging from a few weeks to a few months. The dataset has a total 28,143 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 (shown in the left column of Table 3). We define event types of interest from these test codes (shown in the right column of Table 3).

7.1.3. Platform

The experiments are conducted on RedHat Enterprise Linux 5 operating system using GCC 4.1.2 in Vermont Advanced Computing Core (VACC) cluster computers. VACC uses the IBM Bluemoon cluster with 364 nodes providing roughly 3000 computing cores.

7.2. Experiment results

We run the ICNC and IHCMC algorithms over each of the five cases of synthethic datasets and the real dataset. First, we compare the topologies of the generated networks against the target causal network and determine how closely they resemble the true causal network. Specifically, we count the number of spurious edges and the number of missing edges. Second, we compare the number of reversed edges in the generated networks with the target causal network to evaluate the causal edge directions. In addition, we compare the running time of the ICNC and IHCMC algorithms. In these experiments, we choose the median value of $N_{ET}/2$ for the adjacency span (which ranges from 1 to N_{ET}). Similarly, we select the median value of 0.5 for the attenuation constant τ. We assume events in the stream are in temporal order.

We train and test the causal model to evaluate the performance of the ICNC algorithm. In the training phase, we run the simulated annealing algorithm [18] (SIMULANEALBND function available in MATLAB) to determine the optimal values of

Table 3
Event types defined from the diabetes dataset.

Test Code	Event Type
Regular insulin dose	Regular-insulin-dose-given (RIDG)
NPH insulin dose	NPH-insulin-dose-given (NIDG)
UltraLente insulin dose	UltraLente-insulin-dose-given (UIDG)
Unspecified BGM*	
Pre-breakfast BGM*	
Post-breakfast BGM*	Blood-glucose-measurement-increased (BGMI)
Pre-lunch blood BGM*	
Post-lunch BGM*	Blood-glucose-measurement-decreased (BGMD)
Pre-supper BGM*	
Post-supper BGM*	
Pre-snack BGM*	
Hypoglycemic symptoms	Hypoglycemic-symptoms-exist (HSE)
Typical meal ingestion	Typical-meal-ingested (TMI)
More than usual meal ingestion	More-than-usual-meal-ingested (MTUMI)
Less than usual meal ingestion	Less-than-usual-meal-ingested (LTUMI)
Typical exercise activity	Typical-exercise-taken (TET)
More than usual exercise activity	More-than-usual-exercise-taken (MTUET)
Less than usual exercise activity	Less-than-usual-exercise-taken (LTUET)

Note: BGM*: blood glucose measurement.

temporal strength and gap thresholds ($\delta_{s1}, \delta_{s2}, \delta_g$) at which the topology of the causal network is closest to the target causal network. As discussed in Section 6.1, this topology is influenced by the threshold values. The upper and lower bounds of each threshold are set to 0% and 100%. The optimized thresholds are then used against a new stream of events in the testing phase.

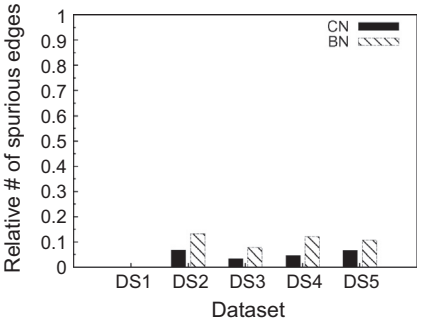
For the synthetic dataset experiment, we randomly divide each dataset into 70% and 30% for training and testing the causal model, respectively. The experiment is repeated ten times for each dataset of each case (DS1 through DS5) to calculate the average relative number of erroneous edges. For the real dataset experiment, we randomly select 70% of the data for training; to test the model, since the dataset is not big enough, we take 50% of the data (30% of the remaining data excluded in the training and 20% of the data used in the training).

7.2.1. Comparison of the network topologies from ICNC and IHCMC

The accuracy of the causal network topology is evaluated by the average relative number of spurious and missing edges.

7.2.1.1. Synthetic dataset experiment. Figs. 6 and 7 show the results of the generated causal network (CN) and Bayesian network (BN). (The results are shown in a chart form as well as a tabular form.) The optimal threshold values obtained in the training phase are shown in Table 4. In addition, Fig. 8 shows the true causal network, the causal network generated by ICNC, and the Bayesian network generated by IHCMC, given the dataset DS3. (Those for the other datasets are omitted in the interest of space.)

Fig. 6 shows that the average relative number of *spurious* edges in CN is smaller than that in BN. It is because the ICNC algorithm, through the temporal network, prunes out all edges whose temporal strengths are below δ_{s1} in the temporal network. Fig. 7 shows that the average relative number of *missing* edges in CN is also smaller than that of BN. This is due to the ICNC algorithm's ability to add edges even if they do not exist in BN when they have temporal strengths greater than δ_{s2} in the temporal network. Moreover, Figs. 6(b) and 7(b) show that the standard deviation in the relative number of spurious edges and missing edges, respectively, is larger for BN than for CN. It is due to the reduction in the number of spurious and missing edges in CN by using the temporal information.

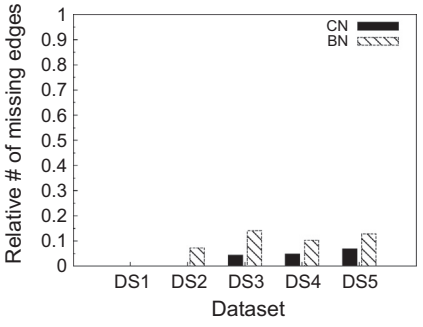


(a) Mean relative number of spurious edges for each case of dataset.

Datasets	CN		BN	
	Mean	Std. dev	Mean	Std. dev
DS_1	0	0	0	0
DS_2	0.067	0.042	0.131	0.080
DS_3	0.034	0.021	0.078	0.046
DS_4	0.046	0.028	0.121	0.073
DS_5	0.066	0.039	0.113	0.066

(b) Mean and Standard deviation of the relative number of spurious edges for each case of dataset.

Fig. 6. Relative number of spurious edges in the causal network (CN) and the Bayesian network (BN) for the synthetic datasets.



(a) Mean relative number of missing edges for each case of dataset.

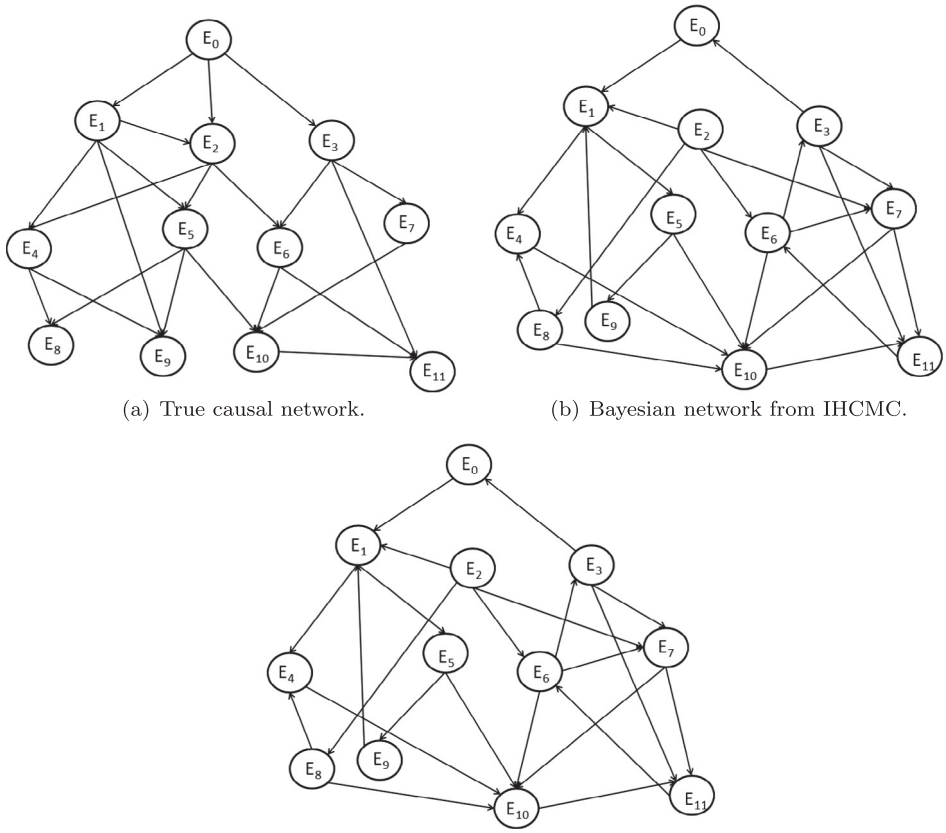
Datasets	CN		BN	
	Mean	Std. dev	Mean	Std. dev
DS_1	0	0	0	0
DS_2	0	0	0.073	0.044
DS_3	0.044	0.026	0.139	0.088
DS_4	0.048	0.028	0.104	0.065
DS_5	0.067	0.041	0.128	0.076

(b) Mean and Standard deviation of the relative number of missing edges for each case of dataset.

Fig. 7. Relative number of missing edges in the causal network (CN) and the Bayesian network (BN) for the synthetic datasets.

Table 4
Optimal threshold values in the synthetic dataset experiments.

Dataset	δ_g	δ_{s1}	δ_{s2}
DS_1	18.3	7.1	84.7
DS_2	6.2	0.5	69.8
DS_3	23.3	5.6	91.4
DS_4	33.7	15.1	86.6
DS_5	30.5	3.8	78.4



(a) True causal network. (b) Bayesian network from IHCMC.

(c) Causal network from ICNC.

Fig. 8. Causal and Bayesian networks from the synthetic dataset DS3.

These results confirm the important role of the temporal network and the threshold mechanism to reduce the number of spurious and missing causal relationships.

7.2.1.2. Real dataset experiment. For the diabetes lab test dataset, the optimal values of δ_g, δ_{s1} and δ_{s2} are 26, 23 and 78, respectively. The causal network shown in Fig. 9(a) is used as the true causal network against which the causal model is trained to determine the optimum threshold values. Fig. 9(b) and (c) are the causal network and the Bayesian network generated by ICNC and IHCMC, respectively. Their relative numbers of spurious and missing edges are compared in Fig. 10.

The causal network from ICNC has only one spurious relationship (RIDG causes NIDG) out of the 55 possible spurious relationships. Clearly, there are a higher number of spurious edges in the Bayesian network from IHCMC – 12 (i.e., 22% of the possible spurious relationships). It is considered to be due to the fact that the parent of a node in a Bayesian network may not necessarily be its cause (unlike the causal network where the parent of a node is always its cause). The lower number of spurious edges in the causal network is due to the temporal strength threshold δ_{s1} and the gap threshold δ_g which prune out the edges with lower temporal strength and narrower gap.

The causal network from the ICNC algorithm has no missing causal relationships out of the 11 causal relationships, whereas the Bayesian network from the IHCMC algorithm has one edge (BGMI causes LTUMI) missing. Evidently, the IHCMC algorithm has failed to identify the causal relationship between BGMI and LTUMI. The ICNC algorithm, on the other hand, has

identified the edge as missing and added it, because the edge has temporal strength higher than the threshold δ_{s2} in the temporal network.

The target causal network (Fig. 9(a)), vetted by a physician, encodes the true causal relationships. We can confirm visually that the causal network from ICNC (Fig. 9(b)) is an almost exact replica of the true causal network. Note the additional causal strength and causal direction confidence labeled on each edge. The resultant causal network shows that NIDG and RIDG respectively cause BGMD. Indeed, it is well known that insulin decreases the blood glucose level[2]; this also explains why BGMI causes RIDG. BGMI also causes LTUMI. Clearly, patients with high blood glucose level are encouraged for less than usual meal ingestion. In addition, BGMI causes MTEUT, which in turn causes BGMD and MTUMI. It is common knowledge that physical activity burns calories, resulting in a decreased blood glucose level (BGMD) and stronger appetite (MTUMI). Note, however, the causal strengths for these two relationships are low, which means MTUET is not a major cause of BGMD. (RIDG and NIDG are the major causes of BGMD.) In addition, BGMD is a strong cause of HSE, LTUET and MTUMI. It is a well-known medical fact that HSE is caused by BGMD and the lower glucose level causes the patients to be prescribed to take less exercise and heavy meal. In fact, HSE, effect of the decrease in blood glucose measurement, is the reason for more than usual meal ingestion (MTUMI).

In summary, these synthetic and real dataset experiments confirm that IHCMC alone is not suitable for building an accurate causal network topology. We observe that ICNC is far superior to IHCMC at detecting causally related event types correctly. The results show that the gap threshold and the temporal strength threshold in the ICNC algorithm provides an effective mechanism to identify and remove spurious relationships or add missing causal relationships.

7.2.2. Comparison of the edge direction of the networks from ICNC and IHCMC

We compare the causal edge directions of the network structures generated by the ICNC and IHCMC algorithms.

7.2.2.1. Synthetic dataset experiment. Fig. 11 shows that the average relative number of *reversed* edges in CN is zero for every dataset. This demonstrates that the ITNC algorithm always detects correct temporal directions from the event stream when generating the temporal network (TN) and the CN generated by the ICNC algorithm inherits these correct temporal directions from TN. In contrast, BN has a higher average relative number of reversed edges for every dataset. In addition, the standard deviation of the relative number of reversed edges in BN is particularly larger than those of spurious or missing edges. This is the effect of *equivalence classes* in BN construction which typically involves different edge orientations. It confirms that a Bayesian network cannot be relied upon for the causal direction.

7.2.2.2. Real dataset experiment. From the causal network and the Bayesian network shown in Fig. 9(b) and (c), we can count the relative number of reversed edges as shown in Fig. 10. Not surprisingly, the Bayesian network has a large number (27%) of reversed causal relationships – for example, the edges between HSE and BGMD, BGMI and MUTET, and BGMD and LTUET. On the other hand, the causal network does not have any reversed causal relationship due to the correct causal direction identified in the temporal network.

Clearly, these synthetic and real dataset experiments confirm that ICNC always gives correct causal directions, thus confirming the crucial merit of temporal precedence relationships in detecting causality. On the other hand, IHCMC can not be relied upon for the causal edge direction.

7.2.3. Comparison of the running time between ICNC and IHCMC

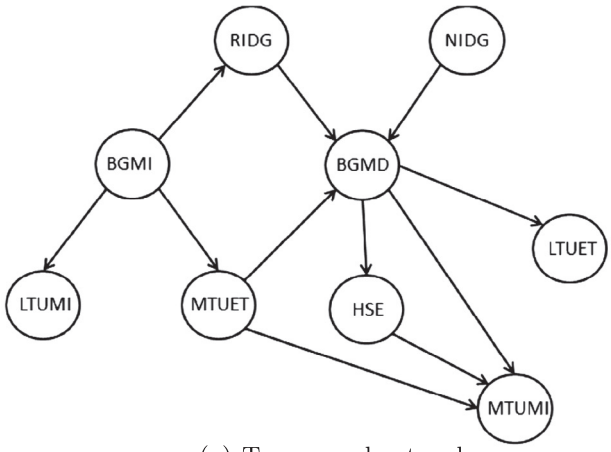
Table 5 shows the CPU time spent on processing the complete event stream and the average CPU time per window, for the IHCMC and ICNC algorithms. As expected, running time of the ICNC algorithm is very close to that of the IHCMC algorithm. The running time of the ICNC algorithm is 15.78%, 12.87%, 12.86%, 6.71%, 6.46%, and 16.23% longer than that of the IHCMC algorithm in $\mathbf{DS}_1$, $\mathbf{DS}_2$, $\mathbf{DS}_3$, $\mathbf{DS}_4$, $\mathbf{DS}_5$, and the real dataset, respectively. This confirms that the IHCMC part consumes most of the running time of the ICNC algorithm and that the running time overhead of the ITNC part is negligible.

8. Related work

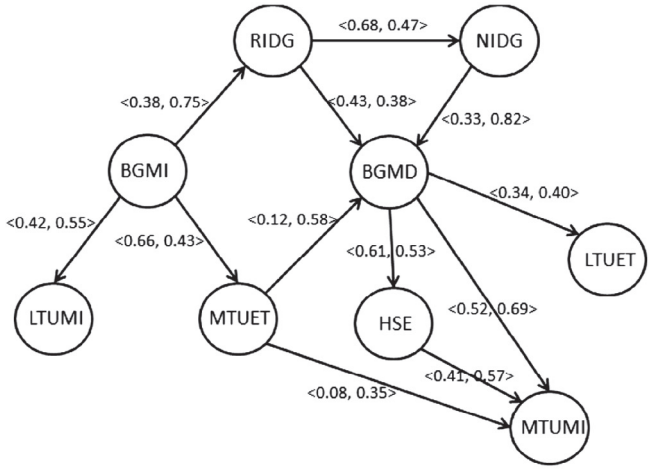
We discuss related work first with respect to the two main approaches used in our work for causal analysis – dependency-based incremental Bayesian network construction and precedence-based incremental temporal network construction. Then, we discuss other work related to causality over data streams and show the uniqueness of our work.

As already mentioned, we use the IHCMC algorithm [2,3] for incremental Bayesian network construction to model the statistical dependencies among events arriving in a stream. Our study concludes that IHCMC is superior to any other existing approaches [5,11,21]. Buntine's approach [5] can construct an alternative Bayesian networks incrementally given a dataset and a proper ordering of the variables, but is designed to update the posterior probabilities only, not the network structure itself. Therefore, this approach is not applicable as we are concerned about learning the network structure and updating it with an arrival of new batch of events. Lam and Bachhus's approach [21] only refines an already existing network under the assumption that it correctly reflects an accurate model and thus the resulting network is biased toward the existing network

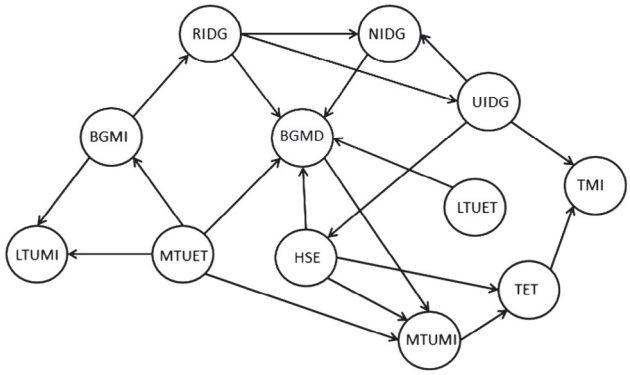
[2] UIDG does not show causing BGMD. In fact, Ultralente insulin is not meant to decrease glucose level. Rather, it provides a baseline level of insulin to support minimum metabolism regardless of ingestion or activity. So, this makes sense.



(a) True causal network.



(b) Causal network from ICNC.



(c) Bayesian network from IHCMC.

Fig. 9. Causal and Bayesian networks from the real dataset.

structure. So, this approach is not suitable for event streams as the network structure needs to be updated as new events arrive. Friedman and Goldszmidt [11] present three approaches which are not appropriate due to the following reasons. The first approach stores all data items, hence is very memory-inefficient. In contrast, the second approach (called Maximum A Posteriori probability) stores only one single network as the summary of past data, and consequently the learning proce-

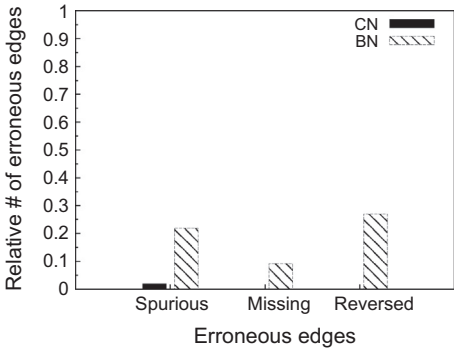
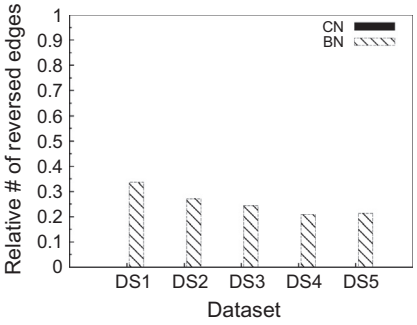


Fig. 10. Relative number of erroneous edges in the causal network (CN) and the Bayesian network (BN) for the real dataset.



(a) Mean relative number of reversed edges for each case of dataset.

Datasets	CN		BN	
	Mean	Std. dev	Mean	Std. dev
DS_1	0	0	0.337	0.203
DS_2	0	0	0.271	0.165
DS_3	0	0	0.243	0.142
DS_4	0	0	0.209	0.119
DS_5	0	0	0.214	0.130

(b) Mean and Standard deviation of the relative number of reversed edges for each case of dataset.

Fig. 11. Relative number of reversed edges in the causal network (CN) and the Bayesian network (BN) for the synthetic datasets.

Table 5
Running time of IHCMC and ICNC algorithms.

Datasets	CPU Time (ms)		N_w	CPU time/window (ms)	
	IHCMC	ICNC		IHCMC	ICNC
DS_1	19	22	1	19	22
DS_2	365	412	2	182.50	206
DS_3	17,790	20,081	16	1111.87	1255.06
DS_4	525,268	560,536	64	8207.21	8758.37
DS_5	5,899,505	6,280,634	512	11522.47	12266.86
DS_R	14,853	15,094	4	3713.25	3773.5

N_w and DS_R are the number of partitioned windows observed and the real dataset, respectively.

dure for new data is biased towards it. The third approach balances between the two, that is, it sacrifices the quality of the constructed network for the reduction in required memory space. Alcobé in his work [1–3] has converted four well known batch-mode Bayesian network construction algorithms (i.e., CL, K2, B and HCMC) to their incremental versions (i.e., ICL, IK2, IB and IHCMC, respectively). Among these, IHCMC performs the most exhaustive search and yields a Bayesian network of the highest quality within a reasonable time [2].

There is rich literature about temporal modeling or reasoning based on temporal precedence (e.g., [12,14,16,19,23]). The most relevant to our project are TIMERS II algorithm by Hamilton and Karimi [14] and temporal causal graph construction by Liu et al. [23], as both propose to use temporal information towards causal analysis. However, neither aims at constructing a causal network as proposed in our work. Specifically, TIMERS II [14] classifies the relationship between decision attribute and condition attributes into instantaneous, causal, or acausal based on temporal information, but it does not build a causal network and only uses simple temporal conditions and decision tree to determine the nature of causal relationship. The temporal causal graph in Liu et al.'s work [23] uses the Granger causality [13], the well known approach for determining causal relationships in a time series data. Given two variables, say X and Y, Granger causality determines that X is a cause of Y if the

past value of X can be used (via regression) to predict the future value of Y. This notion of causality is very different from what is proposed in our work in that ours is based on the frequency of the occurrences of the variable values, not the values themselves, and that each variable in our work can represent any event type, not only a time series variable.

The only existing work we find on causal relationships over data streams is by Kwon and Li [20]. It is about using temporal, spatial, and spatio-temporal relationships between cause and effect to perform causality join query processing on sensor streams. Similarly, there has been recent work by Meliou et al. [27] to support causal query processing in *databases*. However, none of these works addresses causal modeling at all. To the best of our knowledge, there has been no previous work done for causal network modeling by constructing a causal network in a dynamic environment (over event streams). On the other hand, there exists a significant amount of research done on causal analysis in a static environment [4,9,15,22,26]. Semantically, the work by Ishii et al. [17] is closer to the work we are proposing. Their incremental approach is to extract causality for news articles and documents. They employ a lexical approach by comparing subject-verb-object (SVO) tuples for causality detection using natural language processing for each news item. Thus, their work does not infer causality from the observations of the overall data which we are doing in this paper. Our focus is on causality inference over event stream. Besides, it is not useful in event streams where the lexical approach is not a viable option and the causal relationships need to be inferred from the observation of unbounded events.

9. Conclusion and future work

In this paper, we focused on the problem of constructing a causal network over continuous event streams incrementally. We proposed a two layered causal network model and presented an algorithm utilizing temporal precedence relationships and statistical dependency between events in combination. For the temporal precedence relationships, we presented a temporal network model and the temporal network construction algorithm. Then, we combined it with an existing incremental Bayesian network construction algorithm to propose the incremental causal network constructing algorithm, and then through experiments demonstrated that the proposed approach increases the accuracy of causality detection significantly.

In this paper, we assumed the events in a stream are in temporal order. For the future work, we plan to consider the out-of-order event stream which may introduce reversed edges in the temporal network, thereby reducing the reliability of the temporal network. Another interesting direction for the future work is the inclusion of predictive causal query processing. There are applications which can benefit from predicting upcoming events based on the history of prior events.

Acknowledgments

We thank Joseph Roure Alcobé for providing the initial code base for the IHCMC algorithm and Benjamin Littenberg for his comments on the causal network from the diabetes lab test result data set.

References

[1] J.R. Alcobé, Incremental hill-climbing search applied to Bayesian network structure learning, in: First International Workshop on Knowledge Discovery in Data Streams, 2004.
[2] J.R. Alcobé, Incremental Methods for Bayesian Network Structure Learning, Ph.D. thesis, Department of Computer Science, Universitat Politècnica de Catalunya, 2004b.
[3] J.R. Alcobé, Incremental methods for Bayesian network structure learning, Artificial Intelligence Communications 18 (2005) 61–62.
[4] H. Borchani, M. Chaouachi, N. Ben Amor, Learning causal bayesian networks from incomplete observational data and interventions, in: Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU '07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 17–29.
[5] W. Buntine, Theory refinement on Bayesian networks, in: Proceedings of the Seventh Conference (1991) on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 52–60.
[6] P. Chandra, A.D. Kshemkalyani, Causality-based predicate detection across space and time, IEEE Transactions on Computers 54 (2005) 1438–1453.
[7] D.M. Chickering, Learning equivalence classes of Bayesian-network structures, Journal of Machine Learning Research 2 (2002) 445–498.
[8] D.M. Chickering, D. Heckerman, C. Meek, Large-sample learning of bayesian networks is NP-hard, Journal of Machine Learning Research 5 (2004) 1287–1330.
[9] B. Ellis, W.H. Wong, Learning causal Bayesian network structures from experimental data, Journal of the American Statistical Association 103 (2008) 778–789.
[10] A. Frank, A. Asuncion, UCI machine learning repository, 2010.
[11] N. Friedman, M. Goldszmidt, Sequential update of bayesian network structure, in: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, UAI'97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 165–174.
[12] S. Glüge, O.H. Hamid, A. Wendemuth, A simple recurrent network for implicit learning of temporal sequences, Cognitive Computation 2 (2010) 265–271.
[13] C.W.J. Granger, Investigating causal relations by econometric models and cross-spectral methods, Econometrica 37 (1969) 424–438.
[14] H.J. Hamilton, K. Karimi, The timers ii algorithm for the discovery of causality, in: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 744–750.
[15] D. Heckerman, A Bayesian approach to learning causal networks, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1995, pp. 285–295.
[16] P. Holme, J. Saramäki, Temporal Networks, CoRR abs/1108.1780, 2011.
[17] H. Ishii, Q. Ma, M. Yoshikawa, An incremental method for causal network construction, in: Proceedings of the 11th International Conference on Web-age Information Management, WAIM'10, SpringerVerlag, Berlin, Heidelberg, 2010, pp. 495–506.
[18] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[19] R. Krishna, C.T. Li, V. Buchanan-Wollaston, A temporal precedence based clustering method for gene expression microarray data, BMC Bioinformatics 11 (2010) 68.

[20] O. Kwon, K.J. Li, Causality join query processing for data streams via a spatiotemporal sliding window, Journal of Universal Computer Science 15 (2009) 2287–2310.
[21] W. Lam, F. Bacchus, Using new data to refine a Bayesian network, in: Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence, UAI'94, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1994, pp. 383–390.
[22] G. Li, T.Y. Leong, Active learning for causal bayesian network structure with non-symmetrical entropy, in: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 290–301.
[23] Y. Liu, A. Niculescu-Mizil, A.C. Lozano, Y. Lu, Learning temporal causal graphs for relational time-series analysis, in: ICML, Omnipress, 2010, pp. 687–694.
[24] S. Mani, C.F. Aliferis, A.R. Statnikov, Bayesian algorithms for causal data mining, Journal of Machine Learning Research 6 (2010) 121–136.
[25] C. Meek, Causal inference and causal explanation with background knowledge, in: Proceedings of the Eleventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-95), Morgan Kaufman, San Francisco, CA, 1995, pp. 403–410.
[26] S. Meganck, P. Leray, B. Manderick, Learning causal bayesian networks from observations and experiments: a decision theoretic approach, in: Proceedings of the Third International Conference on Modeling Decisions for Artificial Intelligence, MDAI'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 58–69.
[27] A. Meliou, W. Gatterbauer, K.F. Moore, D. Suciu, Why so? or why no? Functional causality for explaining query answers, in: MUD, 2010, pp. 3–17.
[28] J. Pearl, Causality: Models, second ed., Reasoning and Inference, Cambridge University Press, New York, NY, USA, 2009.
[29] J. Pearl, T. Verma, A theory of inferred causation, in: KR, 1991, pp. 441–452.
[30] K. Popper, The Logic of Scientific Discovery, Routledge, reprint edition, 1992.
[31] I. Tsamardinos, L.E. Brown, C.F. Aliferis, The max-min hill-climbing bayesian network structure learning algorithm, Machine Learning 65 (2006) 31–78.