Advance Access publication on 24 February 2017
doi:10.1093/comjnl/bxw098

Causal Prediction of Top-*k* Event Types Over Real-Time Event Streams

Saurav Acharya[1], Byung Suk Lee[1*] and Paul Hines[2]

[1]*Department of Computer Science, University of Vermont, Burlington, VT, USA*
[2]*School of Engineering, University of Vermont, Burlington, VT, USA*
[*]*Corresponding author: bslee@cs.uvm.edu*

This paper addresses the problem of causally predicting the top-*k* most likely next events over real-time event streams. Existing approaches have limitations—(i) they model causality in an acyclic causal network structure and search it to find the top-*k* next events, which does not work with real world event streams as they frequently manifest cyclic causality, and (ii) they prune out possible non-causal links from a causal network too aggressively and end up omitting many less frequent yet important causal links. We overcome these limitations using a novel *event precedence model (EPM)* and a *run-time causal inference* mechanism. The EPM constructs a Markov chain incrementally over event streams, where an edge between two events signifies a temporal precedence relationship between them, which is a necessary condition for causality. Then, the run-time causal inference mechanism performs causality tests on the EPM during query processing, and temporal precedence relationships that fail the causality test in the presence of other events are removed. Two query processing algorithms are presented. One performs *exhaustive search* on the model and the other performs more efficient *reduced search with early termination*. Experiments using two real data sets (cascading blackouts in power systems and web page views) verify efficacy and efficiency of the proposed probabilistic top-*k* prediction algorithms.

Keywords: prediction; top-k query; causal network; event stream

Received 13 April 2016; revised 5 November 2016; editorial decision 9 November 2016
Handling editor: Peter Wood

1. INTRODUCTION

Causal prediction (e.g. [1–3]) is emerging as an essential field for real-time monitoring, planning and decision support in diverse applications such as stock market, electric power grid, sensor network, cyber security and world wide web. There is a need for active systems that continuously monitor the event streams from these applications to allow for the prediction of future effect events in real time. Specifically, given a sequence of potentially causal events, many applications would benefit from good algorithms to predict the next most likely (namely, top *k*) effect events. The potentially huge answer space, however, and the unknown dynamics as well as the streaming nature of data make such top-*k* prediction a challenging task.

Consider the following two scenarios as motivating examples:

Example 1. *Web page click stream*: Consider web-based online systems. A majority of them display the same content for everyone. However, the user experience can be more productive with a dynamic system where content is displayed based on real-time prediction of users' most likely activities, given historical data. One can use the results (i.e. the web pages/links most likely to be visited next) to display the most relevant links, content and advertisements at each step of the user activity.

Example 2. *Electric power grid*: Consider an electric power grid. When components of a power grid fail, as a result of a storm, malfunction or cyber-attack, a cascading sequence of subsequent component failures may result, which may lead to a very large blackouts (e.g. [4]). Thus, a timely prediction of the components that are most likely to fail next, given a list of a few components that have failed, may enable operators to take mitigating actions (like shutting down sections of the power grid) before a large-scale blackout occurs. Cascading blackouts typically progress slowly (minutes to tens of minutes) in the initial stages; a few seconds delay to compute and implement emergency controls is generally sufficient.

In this paper, we meet the challenges for continuously predicting the *top-k* most probable next effects in real-time streams. Specifically, we focus on three central research problems.

(1) *Continuous causal prediction*: To the best of our knowledge, there are no existing top-*k* causal query processing mechanisms that are sufficiently efficient to support time-critical applications such as those in examples 1 and 2. Moreover, the previous works on the causal prediction are based on inefficient exhaustive search (ES) over a large search space of causal network (e.g. [5, 6]), which is not suitable for real-time streams.
(2) *Cyclic causality*: The traditional causal network, which models causality in a directed acyclic graph, does not support cyclic causality such as $A \rightarrow B \rightarrow C \rightarrow A$ or $A \leftrightarrow B$ (e.g. [7, 8]). The event streams from many applications, however, exhibit cyclic relationships. For example, a visitor to a news web site may visit the home page, proceed to read an article, and then return to the home page, creating a cyclic relationship between these vertices in the graph.
(3) *Causal information loss*: The causal Markov condition, often considered an essential property of traditional causal networks, holds conservative assumptions in the causal inference process. As a result, it removes many infrequent causal relationships from the causal network [8–11]. Specifically, the causal Markov condition calls for the removal of any suspicious and weak relationships which could potentially be independent in the presence of one or more events. Often this approach backfires by removing less frequent but important causal relationships [8]. We call this limitation the *causal information loss*.

To address these problems, we proceed as follows in this paper. First, we propose an *event precedence model* (EPM) that captures temporal precedence relationship between every two event types into a first order absorbing Markov chain. The resulting graphical structure is called event precedence network (EPN), where an edge signifies the temporal relationship between two events. The inclusion of *all* temporal precedence—hence likely causal—relationships helps to avoid causal information loss. Since EPN encodes all cyclic as well as non-cyclic precedence relationships from event streams, the less frequent but important potential causal relationships are not discarded. Note that EPN is a generative model of the observed event stream, which is built over a set of predefined event types instead of event instances.

Second, we propose a *run-time causal inference* method to support cyclic causal inference. The cyclic causality has to be resolved in real-time as soon as the cause event whose effects are to be predicted is observed in the event stream. In run-time causal inference, the edges of EPN are examined by causal tests (i.e. marginal and conditional independence [CI] tests) on the fly to determine causality.

Third, we present two query processing algorithms—the ES algorithm and the Reduced Search Early Termination (RSET) algorithm—to continuously predict top-*k* event types with the highest scores based on the inferred causal relationships. The ES algorithm formalizes an outward breadth-first search of EPN, performed to calculate the scores of event types as they are traversed, while identifying a 'causal search order' to enable fail-safe score calculations before the run-time causal inference. The RSET algorithm is built upon the ES algorithm, and reduces the search space and terminates the search early whenever possible. As a result, it decreases the runtime with only marginal reduction in prediction accuracy.

Fourth, we present experiments conducted to evaluate the accuracy and runtime performance of the proposed *ES* and *RSET* algorithms using two real data sets. In each evaluation, there are two objectives. The first objective is to compare the run-time causal inference mechanism of the proposed algorithms (i.e. ES, RSET) against the state-of-the-art traditional causal inference mechanism called the Fast Causal Network Inference (FCNI) algorithm [12]. The FCNI algorithm is essentially inapplicable to our problem due to its inability to handle cyclic causality and runtime causal inference, but is the best available in the state of the art. The second objective is to compare the query processing mechanisms between the ES algorithm and the RSET algorithm.

The contributions of this paper are summarized as follows:

(1) It presents an EPM to represent the temporal precedence relationships between event types and proposes an algorithm to construct an EPN incrementally over event streams. The network construction is fast, and the resulting network is cyclic, which is critical to supporting cyclic causality during causal inference.
(2) It introduces a runtime causal inference mechanism to infer the causal relationships in real time, and proposes two query processing algorithms: ES and RSET, to continuously predict the top-*k* next effects over event streams. Novelty of the algorithms include (i) performing runtime causal tests, (ii) finding causal search order during the EPN search, and (iii) reducing search space and allowing for early termination for computational efficiency.
(3) It empirically demonstrates the advantages of the proposed runtime causal inference mechanism and the query processing algorithms in terms of the prediction accuracy and the runtime.

The remainder of the paper is organized as follows. Section 2 discusses the related work, and Section 3 presents some preliminary concepts. Sections 4 and 5 describe the EPM and the query processing model, respectively. Section 6 evaluates the proposed query processing algorithms. Section 7 concludes the paper and suggests future work.

2. RELATED WORK

This section discusses the limitations of conventional causal inference techniques and related work on predictions, and points out the unique contributions of our work.

There are two approaches for constructing a traditional causal network. The first approach, search and score-based (e.g. [13–16]), performs greedy search (usually hill climbing) over all possible causal networks of the data to select the network with the highest score. This approach, however, has two limitations. First, the computational complexity increases exponentially as the number of variables in the causal network increases. Second, the problem of equivalence classes [17], where two or more network structures represent the same probability distribution, makes the causal direction between nodes quite random and therefore unreliable. The second approach, constraint-based (e.g. [1, 11, 18, 19]), performs CI tests between variables to construct a causal network. This approach does not have the problem of equivalence classes, but performs a prohibitively large number of conditional independent tests. The state-of-the-art FCNI by the authors of this paper [12] provided a faster algorithm for constructing a traditional constraint-based causal network over event streams. (Thus, we consider the FCNI algorithm as the representative of the traditional causal network approach in this paper.) The FCNI algorithm learns temporal precedence relationships from the event stream and only performs causal inference between those event types that exhibit temporal precedence relationship, thereby reducing the number of CI tests. (This idea is employed in this paper as well.) However, like others, FCNI assumed *acyclic* causality in the data.

There has been some work (e.g. [20–22]) to support *cyclic* Bayesian network which aims to handle the cyclic causality in Bayesian networks. This work, however, still carries the drawbacks inherent in the Bayesian network approach—that is, the ambiguity of equivalence classes and the inability to meet the requirement of a causal network that the parent node in the network should always represent the direct cause—and hence is not useful in our work.

The existing body of work on 'causal prediction' only addresses inference of the likelihood of occurrence of an effect variable given a cause variable (e.g. [23–28]), while the prediction of top-k effects requires finding the most likely k effects among all possible effect variables. Therefore, the only way to find the top-k next effects is to construct a traditional causal network, which ignores cyclic causality and suffers from causal information loss, over event streams and then infer the top-k effects of the cause exhaustively (e.g. [5, 6, 29]). To the best of our knowledge, there is no solution that supports cyclic causality, mitigates the causal information loss, and performs only necessary partial search to find the top-k effects of the given causes over event streams.

Web click stream prediction has been a popular research topic lately. Markov models and their variations have been found well-used for this problem (e.g. [30–35]). One fundamental problem of these models, however, is that a Markov model alone cannot explain causality. In contrast, our work considers both—specifically, in a two-step approach, that is, first starting with a lower order Markov chain model and then augmenting it with *causality* tests. Indeed, in many critical applications (like the electric power grid monitoring) where tolerance for false positives is low, we need such a stronger test for prediction by considering not only temporal precedence but also statistical dependency—that is, *causal* prediction.

Association rule mining algorithms (e.g. [36–38]) are extensively used for prediction and recommendation. However, association does not necessarily imply causation (e.g. [39–44]), and therefore they are not useful to our problem due to their exclusion of the fundamental concept of causality. That is, two variables that are associated require stronger conditions, such as temporality and strength, to be considered causally related.

A few works on top-k query processing in the Internet domain, such as over social-tagging networks [45] and over web 2.0 stream [46], were published. Unlike our work, however, these works do not address causal prediction in an event-based environment at all.

3. PRELIMINARIES

In this section, we introduce the concepts that are central to the techniques explained in the paper.

3.1. Event streams

An event stream is a discrete, indefinitely long sequence of event instances. An *event instance* (or event) refers to a time-stamped action which may have an effect. Each event instance is created by one event owner. Two events are related to each other if they share common attributes such as event owner, location and time. These attributes are called *common relational attributes* (CRAs). In examples 1 and 2, the CRAs are the session id and the blackout id, respectively. A prototype for creating events is called an *event type*.

An event has the following schema: $\langle timestamp, type, CRA, attribute\text{-}set\rangle$. That is, an event has the timestamp at which it was created, the event type it belongs to, the CRA value, and a set of additional attributes called the attribute-set.

For simplicity, we refer to an event as e_{ij} where i is the value of the CRA and j (=1, 2, 3,…) is its event type id (E_j).

EXAMPLE 3. Figure 1 shows an illustrative example of events in a user click event stream of example 1. The first field in each line (e.g. e_{21}) denotes the actual event instance shown in the remainder of the line (e.g. ⟨05/05/11 1:12 pm, 1, 2, [200 s, ...]⟩). The session id serves as the CRA and the webpage categories (e.g. frontpage, news, weather, sports, entertainment, tech, local) are the event types. For instance, in the event instance e_{32}, 2 is the event type and 3 is the CRA. Note that the event type is represented by a numerical equivalent of the original event type (e.g. frontpage $= E_1$, news $= E_2$, weather $= E_3$, sports $= E_4$, entertainment $= E_5$, tech $= E_6$, local $= E_7$).

We use a window, called *partitioned window* [12], to collect the events from the stream for a user-specified observation period T. To group related events in the window, these events are partitioned by the CRA and then arranged in the temporal order within individual partitions. Figure 2 shows a partitioned window for the event stream described in Fig. 1. Once the observation period expires, the window shifts to the next batch of events. The last event of one window overlaps the first event of the next window in order to ensure consistency in event precedence modeling across two consecutive windows.

DEFINITION 3.1 (Partition). *A partition W_i in a partitioned window is defined as a set of observed events sharing the same CRA value i and arranged in the temporal order over a time period T, that is*

$$W_i = \{e_{ij}(t) | t \leq T, i \in \mathbf{A}, j \in [1, N]\}$$

where t is the timestamp, $\mathbf{A}$ is the set of all possible CRA values, j is the event type id and N is the number of event types.

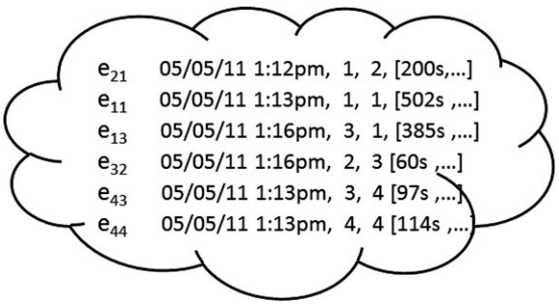

FIGURE 1. Sample of event instances in a stream from example 1.

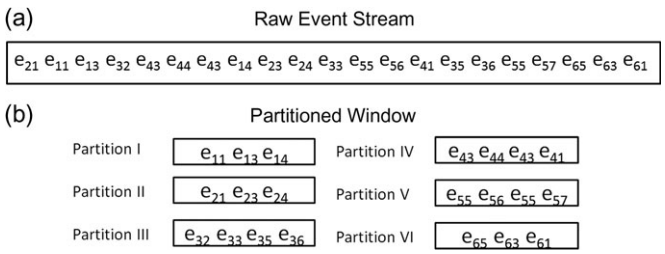

FIGURE 2. Event stream. (e_{ij}s are abbreviations of actual event instances such as shown in Figure 1.)

The events that are being predicted are *effect events* while the events that are used for prediction are *cause events*. We denote the cause event type and the effect event type as $C_i \in \mathbf{E}$ and $T_j \in \mathbf{E}$, respectively, where i and j are the positions of the events in each sequence. Note that C_i and E_i are not necessarily the same, and nor are T_j and E_j.

Table 1 summarizes the key notations used in this paper.

3.2. Causal networks

A causal network (or causal *Bayesian* network) (e.g. [1, 11, 13–16, 19]) is a directed acyclic graph $G = (V, \Xi)$ to encode causality, where V is the set of nodes (representing event types) and Ξ is the set of edges between nodes. For each directed edge, the parent node denotes the cause, and the child node denotes the effect.

The joint probability distribution of a set of N event types $\mathbf{E} \equiv \{E_1, \ldots, E_N\}$ in a causal network is specified as

$$P(\mathbf{E}) = \prod_{i=1}^{N} P(E_i | \mathbf{Pa}_i)$$

where $\mathbf{Pa}_i$ is the set of the parent nodes of event type E_i.

TABLE 1. Definitions of key symbols.

Symbols	Definitions
N_p	Number of partitions
N_{ei}	Number of event instances in the ith partition
N_e	Total number of event instances in all partitions
E_i	Event type with id i
e_{ij}	Event instance of type j and CRA i
N	Number of event types
C_i	Cause event type at position i
T_i	Effect event type at position i
O	Causal search order
S_i	Event type at the ith position in O
C_δ	The most recent cause event type
$\mathbf{E}$	Set of N event types in the data $\{E_1, E_2, \ldots, E_N\}$
R_k	Ranked list of the *top-k* event types
$N_{\text{instances}}$	Number of event instances
N_{CRA}	Number of common relational attributes
$f(E_i, E_j)$	Number of observations of the instances of type E_i followed by the instances of type E_j

Consider the event stream of Fig. 2. The causal relationships among the event types in the stream may be modeled as a causal network like the one shown in Fig. 3.

3.3. CI tests

CI test establishes causality between two random variables X and Y in the presence of a set of random variables, C. One of the popular metrics for CI, is *conditional mutual information* (CMI) (e.g. [18, 47]).

$$\mathrm{CMI}(X, Y|\mathbf{C}) = \sum_{x\in X}\sum_{y\in Y}\sum_{c\in\mathbf{C}} P(x, y, c)\log_2\frac{P(x, y|c)}{P(x|c)P(y|c)}$$

where P is the probability mass function calculated from the frequencies of variables. CMI gives the strength of dependency between variables in a measurable quantity, which helps to identify the weak (or spurious) causal relationships.

In the traditional CMI, two variables X and Y are said to be independent if $\mathrm{CMI}(X, Y|\mathbf{C}) = 0$, and dependent otherwise. This criterion itself offers no distinction between weak and strong dependencies. With a higher value of $\mathrm{CMI}(X, Y|\mathbf{C})$, the dependency between X and Y should be considered stronger. Thus, weak dependencies are pruned out using a threshold CMI value, below which the evidence is considered 'too weak'. To do so, we relate CMI with the G^2 test statistics [1, 48] as below:

$$G^2(X, Y|\mathbf{C}) = 2 \cdot N_s \cdot \log_e 2 \cdot \mathrm{CMI}(X, Y|\mathbf{C})$$

where N_s is the number of samples (i.e. event instances).

Under the independence assumption, G^2 follows the χ^2 distribution [49] with the degree of freedom df equal to $(n_x - 1)(n_y - 1)\prod_{s\in S} n_s$, where n_x, n_y and n_s are the number of possible distinct values of X, Y and S, respectively. So, we perform the test of independence between X and Y given $\mathbf{C}$ by using the calculated G^2 test statistics as the χ^2 test statistics in a χ^2 distribution, which provides the threshold based on df and significance level α, to validate the result. We set α as the typically accepted value of 95% [50].

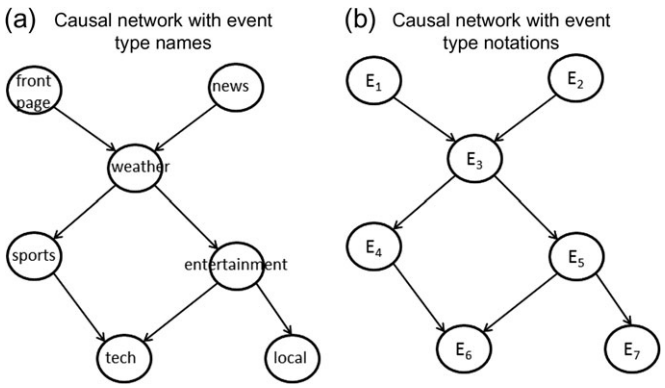

FIGURE 3. Causal network. (The event type names in the subfigure a are from the web page click stream in Figure 2, and the event type notations in the subfigure b are symbols used to represent the real event type names.)

We define a Boolean function *IsIndependent* $(X, Y, \mathbf{C})$ to test the CI between two variables X and Y given a set of condition variables $\mathbf{C}$ using the G^2 test statistics. It returns true if these two variables are conditionally independent for any subset of $\mathbf{C}$; otherwise, it returns false. Specifically, the *IsIndependent* function performs a series of CI tests, one for each of the $2^{|\mathbf{C}|} - 1$ distinct subsets of $\mathbf{C}$. The goal of each CI test is to check if X and Y are *dependent* in the presence of the given subset of $\mathbf{C}$. To control false positives, a single failure in any of these tests returns that X and Y are not dependent, that is, they are independent. The statistical power of the *IsIndependent* test is the product of the statistical powers of individual conditional independent tests, which could be calculated provided with the population sizes of the individual subsets of $\mathbf{C}$.

The unbounded and continuous nature of event streams of interest makes it infeasible to store all of the historical data. Therefore, we use an incremental approach such that when a new batch of events is processed, we only update the record of the frequency of observations without storing the old events.

4. EVENT PRECEDENCE MODEL

In this section, we introduce the proposed incremental mechanism to model the precedence relationships between events in a network structure.

4.1. Model

To overcome the problems described in Section 1, we propose an *EPM*. Founded upon the fact that temporal precedence is a required condition for causality, EPM models the temporal precedence relationship between events as a Markov chain. The resulting graphical structure is called *EPN*. Specifically, EPM takes the partitioned window (collected from the event stream) as an input and incrementally builds a model to reflect all precedence relationships found in it. Once a new batch of events arrives, the EPN is updated with the information from the new partitioned window. Such an adaptive approach is essential for a streaming environment with continuous and unbounded data. To avoid any information loss, the evidence of every precedence relationship is preserved.

We have made the following decisions for in the EPM:

- Causal relationships between events of the same type are not considered and, therefore, such precedence relationships are ignored. Otherwise, with our causal modeling done at the event type level, there would be 'self-causation' introduced in the model, which might present a serious flaw in causal inference. It is particularly true in such circumstances as in the electric

power grid application, where the same component cannot fail twice when cascading failures occur (see details about the data set in Section 6.1.2).

- The cause and effect events should share the same CRA value. As described in Section 3.1, the events are grouped into partitions based on their *CRA* values (e.g. session id in example 1, blackout id in example 2, respectively). In other words, two events are not related to if they have different CRA values.

The proposed EPM is a *first-order absorbing* Markov chain [51] which models both direct and indirect dependencies through a sequence network structure. The chain is 'absorbing' to allow every state to transition to an absorbing (a.k.a. terminating) state—in other words, to accommodate in an EPN those events that are not causing other events (which do happen in real applications). Since in a first order Markov chain an observation is independent of all previous observations except the most recent one, the probability of occurrence of an effect event given past cause events is given as follows:

$$P(T_0|C_0, C_1,\ldots, C_\delta) = P(T_0|C_\delta).$$

$P(T_0|C_\delta)$ can be rewritten as below:

$$P(T_0|C_\delta) = \frac{P(T_0, C_\delta)}{P(C_\delta)}$$

By approximating the probabilities in the numerator and the denominator, $P(T_0|C_\delta)$ can be estimated as shown below:

$$P(T_0|C_\delta) \approx \frac{\frac{f(C_\delta, T_0)}{N}}{\frac{f(C_\delta)}{N}}$$

where N is the number of event instances observed so far and $f(C_\delta, T_0)$ denotes the number of observations in which instances of the type C_δ precedes instances of the type T_0. This equation is then simplified to

$$P(T_0|C_\delta) \approx \frac{f(C_\delta, T_0)}{f(C_\delta)}$$

and, by replacing $f(C_\delta)$ by the sum of $f(C_\delta, E_j)$ over all children nodes E_j's, we obtain

$$P(T_0|C_\delta) \approx \frac{f(C_\delta, T_0)}{\sum_{E_j \in children(C_\delta)} f(C_\delta, E_j)} \quad (1)$$

In summary, *EPM* allows us to automatically build a tractable probabilistic graphical model from the events, discovering the existing dependencies among the event types in the event stream. These dependencies are represented by a graph, as illustrated in Fig. 4, where conditional probabilities are stored at the condition node. Storing conditional probabilities in the condition nodes facilitates computing the probability of an effect given the occurrence of a cause effect.

4.2. Algorithm

Algorithm 1 outlines the EPN construction algorithm. It has two steps: *observation* and *graph generation*. These steps are discussed below:

(1) *Observation:* This step observes adjacent neighbor events in each partition of the window to learn the precedence relationships and updates the frequency matrix. Note that, based on the assumptions stated earlier, the precedence relationships should be between events in the same partition and between events of different types. For every pair of adjacent events that meet these criteria, it increment by 1 the frequency matrix's count element between the ordered pair of their event types, denoted as $f(E_i, E_j)$ where E_i and E_j are the event types of the two adjacent events, respectively. The frequency matrix is updated incrementally for each new partition of events.

(2) *Graph generation:* This step starts with an edgeless graph $G = (V, \Xi)$ where V is the set of nodes (event types) and Ξ is an empty set of edges. Then, for any evidence of precedence relationship between event types E_i and E_j (i.e. f(E_i, E_j) > 0), an edge is added between the two nodes that represent these event types. Note that the graph supports a cyclic loop of edges via two or more nodes; thus, the graphical model offers the flexibility to incorporate all possible types of relationships, unlike in the traditional systems where only directed edges are supported. In

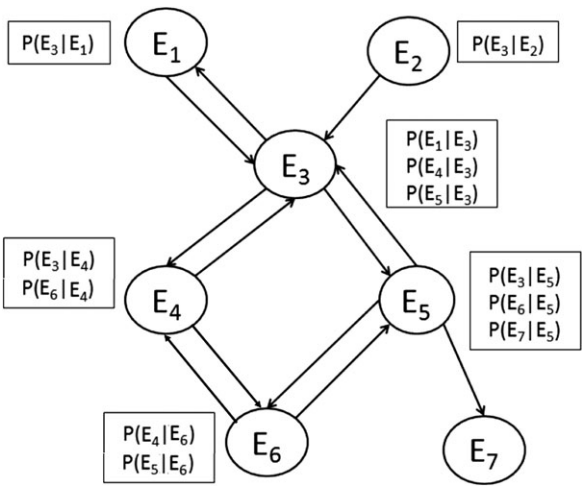

FIGURE 4. Illustration of EPN construction from the event stream in Fig. 2.

addition, for every edge added in the graph, the probability of an event type given its parent event type is calculated using equation 1. The calculated probability is then stored in the parent node.

The running time complexity of the algorithm is polynomial with the total number of events that have arrived thus far and the number of event types. First, the observation step counts every pair of consecutive events in every partition of the window. Clearly, for each partition, the number of the counts is always one less than the number of events in it. If N_e and N_p are the number of events and the number of partitions, respectively, then the running time complexity of this step is given as $\mathrm{O}\left(\sum_{i=1}^{N_p}(N_{ei}-1)\right)=\mathrm{O}(N_e)$, where N_{ei} is the number of events in the ith partition. Second, the graph generation phase checks for the evidence of the precedence relationships between every pair of event types. In the worst case, the EPN is completely connected (including cyclic edges) and has $N(N-1)$ edges. So, the running time complexity of this step is $\mathrm{O}(N^2)$. Hence, the total running time of the algorithm is given as $\mathrm{O}(N_e+N^2)$.

Algorithm 1 *EPM construction.*

Require: A partitioned window P from a batch of new events.
{*Observation*: }
1: **for** each partition $W_k \in P$ where k is the CRA value **do**
2: **for** each pair of consecutive events (of type E_i and E_j, respectively, such that $i \neq j$) in W_k **do**
3: $f(E_i, E_j)$++, i.e. increase the observed frequency by 1;
4: **end for**
5: **end for**
{*GraphGeneration*: }
6: Construct an edgeless network $G = (V, \Xi)$;
7: **for** each pair of event types, E_i and E_j such that $i \neq j$, **do**
8: **if** $f(E_i, E_j) > 0$ **then**
9: $\Xi := \{\Xi \cup \{E_i \rightarrow E_j\}\}$ i.e. add an edge $E_i \rightarrow E_j$;
10: $P(E_j|E_i) := \dfrac{f(E_i, E_j)}{\sum_{E_k \in children(E_i)} f(E_i, E_k)}$;
11: **else**
12: $P(E_j|E_i) := 0$;
13: **end if**
14: **if** $f(E_j, E_i) > 0$ **then**
15: $\Xi := \{\Xi \cup \{E_j \rightarrow E_i\}\}$ i.e. add an edge $E_j \rightarrow E_i$;
16: $P(E_i|E_j) := \dfrac{f(E_j, E_i)}{\sum_{E_k \in children(E_j)} f(E_j, E_k)}$;
17: **else**
18: $P(E_i|E_j) := 0$;
19: **end if**
20: **end for**

5. TOP-K PREDICTIVE QUERY PROCESSING

In this section, we first discuss the predictive causal query processing model and then present the two top-k continuous predictive query processing algorithms—*ES* algorithm and the more efficient *RSET* algorithm.

5.1. Predictive query processing model

The predictive query processing problem can be formulated as a search problem to find the possible effects of a given set of observed events in a causal network. The traditional causal network, however, is not equipped to run a causal inference query in the face of causal information loss and lack of support for cyclic causality. To address this issue, we propose to infer causality from the EPN 'on the fly' during query processing, since every causal relationship subsumes a temporal precedence relationship.

In our work, the predictive query is a standing query running continuously, and the ranked result list may change when a new event is observed. This query processing is done by exploring the EPN, which represents all precedence relationships between event types. Since a cause event always precedes its effect events, an outward breadth first search from a cause event node in the EPN is required to predict its effect events. If a node is revisited, as EPN is cyclic, it is ignored. We call the event type, from which EPN exploration starts, the *effect observation point* (EOP). For instance, in Fig. 4, consider the two event types E_3 and E_4. E_3 is the effect of E_4 when E_4 is the EOP whereas E_4 is the effect of E_3 when E_3 is the EOP, as illustrated in Fig. 5.

Two issues, however, make EPN insufficient to answer a causal predictive query. First, two variables that have a precedence relationship are not necessarily causally related, that is, unless they prove to be statistically dependent as well. Second, two variables that are causally related in the absence of other variables may not be in the presence of others. For example, rain and wet ground are causally related, as rain causes ground to become wet. However, they are not, in the presence of a roof over the ground (which is a conditional variable), as the roof keeps rain from causing the ground wet. To resolve these two issues, we perform CI tests (described in Section 3.3) on the edges of the EPN during query

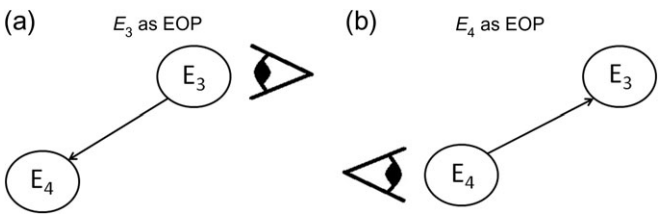

FIGURE 5. Views from the EOPs for Fig. 4.

processing. The edge between two event types that fail these tests is determined non-causal, and hence ignored.

The ranking score of the predicted effect event type E_i is calculated as $P(E_i|C_\delta)$ given its EOP C_δ. An EPN node stores the conditional probability of every child node given the current node as the parent node. Scores across a chain of event types, $E_g \rightarrow E_p \rightarrow E_i$ (where E_p is a parent of E_i and E_g is a parent of E_p), in EPN is calculated using the multiplicative property of conditional probability $P(E_i|E_g) = P(E_i|E_p) \cdot P(E_p|E_g)$.

5.2. ES algorithm

5.2.1. *Approach*

The most straightforward approach to the top-k prediction problem is to search for all possible effects exhaustively during the runtime causal inference over EPN and then sort them in non-increasing order of the score to determine the k effects with the top scores. This *ES* approach offers a robust strategy for exploring the EPN to infer effects, and, as mentioned earlier, an outward breadth first search is performed over the EPN for runtime causal inference.

The score calculation of the effects, however, is not straightforward. To apply multiplicative property of conditional probability described in Section 5.1, the scores of the parents of an event type should be known before its score can be calculated, but this is not always possible as demonstrated in Fig. 6. Therefore, we employ a search strategy that determines the *causal search order* before exploring the EPN for run-time causal inference. It gives us an order of event types when traversing the EPN such that the probabilities of parent nodes are always known before calculating the probabilities of their children nodes.

DEFINITION 5.1 (*Causal Search Order*). *The causal search order O is an ordered set of event types $\{S_1, S_2, \ldots, S_N\}$ observed during the outward breadth first search of the EPN such that S_{i+j} is never an ancestor of S_i, where $j > 0$ and $i + j \leq N$.*

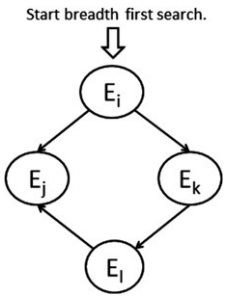

FIGURE 6. Illustration of the need for a causal search order. (Score of Ej is $P(E_j\ {}_jEi) + P(E_j\ {}_jEl)\ P(El_jE_k)\ P(Ek_jE_i)$. Note that El is explored after E_j (and E_k) in breadth-first search and, therefore, $P(El_jEi)$ is not known when E_j 's score is calculated.)

EXAMPLE 4. Let us illustrate the *causal search order* considering the EPN shown in Fig. 4. See Fig. 7 for the illustration. As described earlier, we run outward breadth first search in EPN from the EOP. Suppose E_3 is the EOP. Initially, E_3 is added to O and is explored. Then, the children of E_3 are added, so O becomes $\{E_3, E_1, E_4, E_5\}$. Then, since E_3 has already been explored, the next unexplored node in O, E_1, is explored. However, no new nodes are added to O as E_1 has no child. Then, we consider the next unexplored node (E_4) in O and add its unexplored child node E_6 to O. Now, the next unexplored node in O is E_5. So, the children of E_5 are added to O, which then becomes $\{E_3, E_1, E_4, E_5, E_6, E_7\}$. The recently added unexplored node E_6 has no unexplored children and E_7 has no

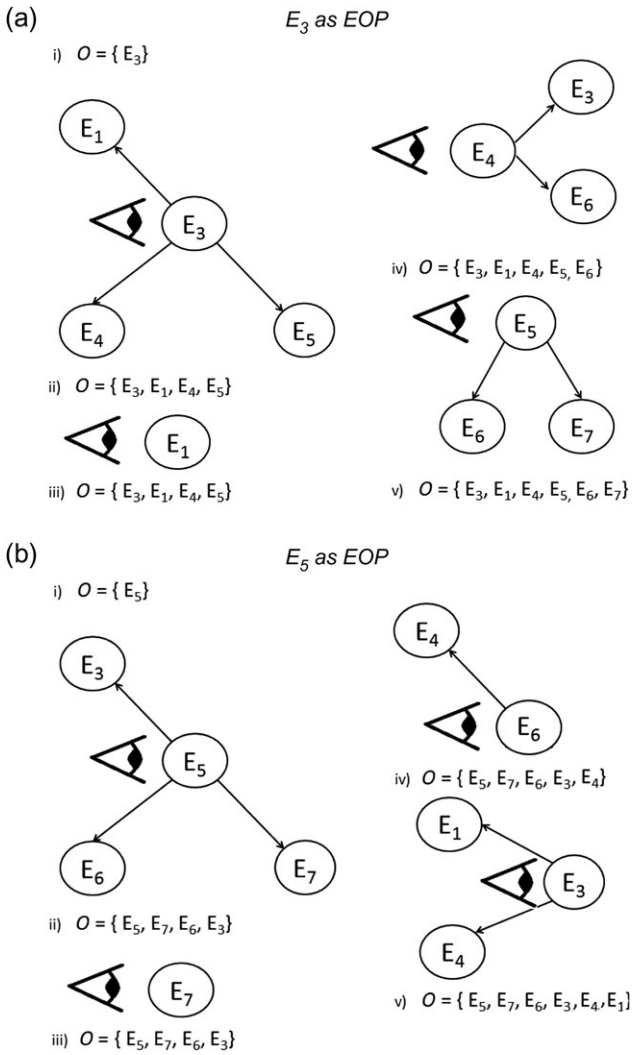

FIGURE 7. Illustration of the steps to determine causal search order for the EPN of Fig. 4.

children and, therefore, no new nodes are added to O. So, the final causal search order O is $\{E_3, E_1, E_4, E_5, E_6, E_7\}$. These steps are shown in Fig. 7(a). Similarly, when EOP is E_5, the causal search order O may be determined to be $\{E_5, E_7, E_6, E_3, E_4, E_1\}$, as shown in Fig. 7(b).

5.2.2. *Algorithm*

Algorithm 2 outlines the ES algorithm. It has a two-pass strategy for exploring EPN to infer effects. In the first pass, breadth-first search is performed over the EPN to determine the causal search order. In the second pass, EPN is explored in this search order for runtime causal inference. The input to the algorithm includes the EPN G, the most recently observed cause event type, i.e. EOP, C_δ, and the size k of the result. The main steps of the algorithm are given as follows:

(1) First, an outward breadth first search over the EPN is run from the EOP to determine the *causal search order* O (Line 2).
(2) Second, *marginal independence tests*[1] are performed between each pair of event types that have an edge in the EPN, and the edge is removed from the EPN if the two event types are judged independent, which is an indication of weak causal relationship (Lines 3–7). (See Section 3.3 for the function *IsIndependent*.) Note that this step helps to reduce the number of tests required in the next step.
(3) Third, the EPN is traversed to find the effect events of each unexplored node E_j according to the order of event types in O (Line 8). We perform *CI tests* on edges between E_j and each of its parents and, then, the edge is excluded if the two event types are judged conditionally independent, which is an indication of weak causal relationship (Lines 9–13). Note that in our work only the parents of E_j can have causal effects on E_j and, therefore, we perform the tests only against them. Then, the score of the node E_j is calculated from the remaining edges (i.e. those that are judged to be dependent) and is stored in the buffer together with the node (Lines 16–17).
(4) Finally, all event types thus explored are sorted in non-increasing order of the score, and then the event types with the top-k scores are returned (Line 19).

EXAMPLE 5. Let us illustrate the ES algorithm considering the EPN shown in Fig. 4. Suppose C_δ is E_3 and k is 2.

(1) The causal search order O, from the EOP (i.e. E_3), is determined as $\{E_3, E_1, E_4, E_5, E_6, E_7\}$.

[1]A marginal independence test disregards the effect of other event types; in other words, it is equivalent to a CI test with an empty condition set.

Algorithm 2 *ES algorithm.*

Require: EPN $G = (V, \Xi)$; EOP C_δ; result size k.
1: Create empty buffer B_T (to store the effect event types and their scores);
2: Determine *causal search order*, O, with the outward breadth first search in G from the EOP C_δ;
3: **for** every edge $E_i \rightarrow E_j \in \Xi$ **do**
4: **if** *IsIndependent* $(E_i, E_j, \varnothing)$ returns true **then**
5: $\Xi := \Xi - \{E_i \rightarrow E_j\}$; {Exclude the edge}
6: **end if**
7: **end for**
8: **for** every node $E_j \in (O - C_\delta)$ **then**
9: $S_{parents} :=$ parents of E_j;
10: **for** every node $E_i \in S_{parents}$ **do**
11: **if** *IsIndependent* $(E_i, E_j, S_{parents} - \{E_i\})$ returns true **then**
12: $\Xi := \Xi - \{E_i \rightarrow E_j\}$;{Exclude the edge}
13: $S_{parents} := S_{parents} - \{E_i\}$;
14: **end if**
15: **end for**
16: $P(E_j|C_\delta) := \sum_{E_p \in S_{parents}} P(E_j|E_p) P(E_p|C_\delta)$;
17: Insert the pair $(E_j, P(E_j|C_\delta))$ into B_T;
18: **end for**
19: Sort the nodes in B_T in non-increasing order of score and return the top-k results from B_T;

(2) The marginal independence tests are performed on each edge in EPN. For simplicity in illustration, we assume that these tests fail to exclude any edge.
(3) Now, the score of each event type in O is calculated and stored into the buffer B_T:
  (a) The score of the first unexplored event type E_1 is calculated and updated in B_T as follows:
    (i) Determine the parents of E_1, that is, $S_{parents} := \{E_3\}$.
    (ii) Perform CI test between E_3, (the single parent of E_1) and E_1. Suppose the CI test succeeds and thus the edge $E_1 \rightarrow E_3$ is considered a causal link and is not excluded.
    (iii) Calculate the score of E_1 as $P(E_1|C_\delta) \equiv P(E_1|E_3)$, which we assume equals 0.33.
    (iv) Update B_T as $\{(E_1, 0.33)\}$.
  (b) The score of the next unexplored event type E_4 is calculated similarly as above, and let us assume that the score is 0.50. Then, B_T is updated to $\{(E_1, 0.33), (E_4, 0.50)\}$.
  (c) Following the same step as above, let us say B_T is updated to $\{(E_1, 0.33), (E_4, 0.50), (E_5, 0.16)\}$ for the next event type E_5, to $\{(E_1, 0.33), (E_4, 0.50), (E_5, 0.16), (E_6, 0.143)\}$ for the event

type E_6, and to $\{(E_1, 0.33), (E_4, 0.50), (E_5, 0.16), (E_6, 0.143), (E_7, 0)\}$ for the event type E_7. Note that the zero score for E_7 (i.e. $P(E_7|C_\delta) = 0$) means that the CI test between E_7 and E_5 has failed and as a result the edge $E_5 \rightarrow E_7$ has been excluded from the EPN.

(4) B_T is sorted in non-increasing order of the score to $\{(E_4, 0.50), (E_1, 0.33), (E_5, 0.16), (E_6, 0.143), (E_7, 0)\}$. Then, the top two, $(E_4, 0.50)$ and $(E_1, 0.33)$, are selected from B_T and returned.

5.3. RSET algorithm

5.3.1. *Approach*

While the ES algorithm is robust, it evidently scales poorly as the number of event types, hence the network size, increases. This section describes the alternative, *RSET* algorithm we designed to overcome or alleviate the problem, with the goal of reducing the running time with little or no reduction in the prediction accuracy. Specifically, the strategies discussed below are employed.

To reduce the query execution time, we first reduce the search space in EPN by exploring only descendants of the nodes *currently in the top-k*. The nodes that are not in the top-k or their descendants have lower scores, due to the multiplicative property of conditional probability (in Section 5.1), and therefore are disqualified from being top-k candidates. Second, we use a priority-based breadth-first search in the EPN with an *early termination* criterion such that the query execution is stopped as soon as it is certain that the top-k results have been found. This search always chooses the unexplored descendant node with the highest score to explore its children. The early termination criterion is met when there is no change in the list of event types in the top-k, that is, when there is no more descendant node whose score can be greater than those in the current top-k. This strategy effectively reduces the EPN to be only partially explored. For this reason, even though the causal inference is done at runtime, it incurs only a small overhead.

Reduced search may affect the achieved accuracy. The prediction error would be higher when the value of k is smaller or when the graph density of EPN is higher, because then more nodes might be ignored. In practice, however, the impact is insignificant as explained below in the algorithm.

5.3.2. *Algorithm*

Algorithm 3 outlines the RSET algorithm and can be described as follows:

(1) First, two empty buffers, B_C and B_k, are created (Line 1), and the *EOP*, C_δ, with 1 as its score is added to both buffers (Line 2). Note that C_δ has the probability of 1 because the event type has already been observed. B_C is to store the event types explored during query processing, and B_k is to store the top-k effect event types computed.

(2) Second, marginal independence tests are performed (Lines 3–7), in the same way as in the ES algorithm (algorithm 2).

(3) Then, the algorithm traverses the EPN in the priority-based breadth-first search order starting with the EOP while performing CI tests on edges selected to reduce the search space. The key ideas implemented are as follows. First, the EPN is traversed for the

Algorithm 3 *RSET algorithm.*

Require EPN $G = (V, \Xi)$; EOP C_δ; the result size k
1: Create two empty buffers $B_C = \phi$ and $B_k = \phi$;
2: $B_C \coloneqq B_C \cup \{(C_\delta, 1)\}$;
3: **for** every edge $E_i \rightarrow E_j \in \Xi$ **do**
4: **if** *IsIndependent* (E_i, E_j, ϕ) returns true **do**
5: $\Xi \coloneqq \Xi - \{E_i \rightarrow E_j\}$; {Exclude the edge}
6: **end if**
7: **end for**
8: **for** each unvisited node $E_c \in (\{C_\delta\} \cup$ set of event types in B_k) with the highest score **do**
9: Mark E_c as visited;
10: $S_{children}$:= children of E_c;
11: **for** each node $E_j \in S_{children}$ **do**
12: Set of parents $S_{parents} \coloneqq$ set of parents of $E_j \cap$ set of event types in B_C;
13: **for** each node $E_i \in S_{parents}$ **do**
14: **if** *IsIndependent*$(E_i, E_j, S_{parents} - \{E_i\})$ is true **then**
15: $\Xi \coloneqq \Xi - \{E_i \rightarrow E_j\}$; {Exclude the edge}
16: $S_{parents} \coloneqq S_{parents} - \{E_i\}$;
17: **end if**
18: **end for**
19: $P(E_j|C_\delta) \coloneqq \sum_{E_p \in S_{parents}} P(E_j|E_p) P(E_p|C_\delta)$;
20: $B_C \coloneqq B_C \cup \{(E_j, P(E_j|C_\delta))\}$;
21: **if** $|B_k| \leq k$ **then**
22: $B_k \coloneqq B_k \cup \{(E_j, P(E_j|C_\delta))\}$;
23: Sort the event types in B_k in non-increasing order of their scores;
24: **else**
25: Find the entry with the lowest score, (E_{lowest}, P_{lowest}), in B_k.
26: **if** $P(E_j|C_\delta) >$ the lowest value in B_k **then**
27: $B_k \coloneqq (B_k - \{(E_{lowest}, P_{lowest})\}) \cup \{(E_j, P(E_j|C_\delta))\}$;
28: Sort the event types in B_k in non-increasing order of their scores;
29: **end if**
30: **end if**
31: **end for**
32: **end for**

EOP or each unvisited node in the buffer B_k (i.e. top-k) (Line 8). In other words, for any child event type E_c that is not in B_k, E_c and its descendants are ignored, thus reducing the search space. Note, again, that the probabilities of the children of E_c are much lower than that of E_c due to the multiplicative property of conditional probability. We then perform CI test on edges between each child E_j of E_c and each of its parents already visited, and the edge is excluded if judged conditionally independent (Lines 12–18). The score of the node E_j is then calculated from the remaining edges (Line 19) and is stored in B_c and B_k (Lines 20–30). If the buffer B_k overflows, then the node with the lowest score, E_{lowest} is removed from the buffer (Lines 25–29), which further reduces the search space. The buffer B_k is always sorted in non-increasing order of the score after a new event type is added to it (Line 23 and Line 28), and so in traversing the EPN the priority is always given to an *unvisited* node with the highest score.

Note that, although the search space is reduced by ignoring those nodes in B_c that are not in B_k, its impact on the prediction accuracy is insignificant because children of the ignored nodes have even lower scores and therefore have no chance of being in the top-k result.

In addition, note that the algorithm terminates early when there is no change in the list of event types in B_k (see Line 8 of the algorithm) after exploring their children because it means that there is no more event type beyond the current level of exploration in the EPN that has higher score than those already in B_k.

Computational complexity of the RSET algorithm is dominated by the number of CI tests, as is for the ES algorithm, and for both algorithms the number is exponential in the worst case [12]. Hence, in the worst case both algorithms have exponential computational complexity. In practice, however, the RSET algorithm reduces the computational complexity significantly by virtue of its reduced search space and early termination strategies.

EXAMPLE 6. Let us illustrate the RSET algorithm considering the EPN shown in Fig. 4. Suppose C_δ is E_3 and k is 2. We omit the causal search ordering and marginal independence test steps:

(1) Two empty buffers B_C and B_k are created to store all the event types explored so far and to store the current top-k predicted event types, respectively.
(2) The search starts with the EOP (E_3) and updates B_C to $\{(E_3, 1)\}$.
(3) Each *unvisited* event type is explored as follows. For simplicity of illustration, we assume that the CI tests return false and hence the edges are not excluded.
   (a) The first unvisited event type, E_3, is marked as visited and its children, $S_{children}$ (=$\{E_1, E_4, E_5\}$), are explored.
      (i) The score of the first unexplored child, E_1, is calculated and added to the two buffers as follows:
         (A) Determine the parents of E_1; $S_{parents}$ i set to $\{E_3\}$.
         (B) Perform CI test of the edge between E_1 and E_3 given $S_{parents}$.
         (C) Calculate the score of E_1 as $P(E_1|C_\delta)=P(E_1|E_3)\,P(E_3|C_\delta) = 0.33$.
         (D) Update B_C to $\{(E_3, 1), (E_1, 0.33)\}$.
         (E) Update and sort B_k to $\{(E_1, 0.33)\}$.
      (ii) The same steps as above are followed for the next unexplored child E_4. The two buffers B_C and B_k are updated to $\{(E_3, 1), (E_1, 0.33), (E_4, 0.50)\}$ and $\{(E_4, 0.50), (E_1, 0.33)\}$, respectively.
      (iii) B_C and B_k are updated to $\{(E_3, 1), (E_1, 0.33), (E_4, 0.50), (E_5, 0.16)\}$ and $\{(E_4, 0.50), (E_1, 0.33)\}$, respectively, for the next unexplored child E_5.
   (b) Now, the next unvisited event type, E_4 in B_k, is marked as visited and its children are explored. However, there is no child of E_4, i.e. $S_{children}$ is empty. Therefore, there is no computation done.
   (c) The same result is seen for the next event type, E_1, as well. As it has no child (i.e. empty $S_{children}$), there is no computation done.
(4) The top-k result in B_k is obtained as $\{(E_4, 0.50), (E_1, 0.33)\}$.

For the same C_δ, the RSET algorithm considered only four event types—E_3, E_1, E_4, E_5—and the ES algorithm considered all event types—E_3, E_1, E_4, E_5, E_6, E_7. Note that in this example, RSET produced the same result (i.e. $B_k = \{(E_4, 0.50), (E_1, 0.33)\}$) as ES. This is typical unless the value of k is significantly large. It shows the merit of the early terminating reduced search approach of the RSET algorithm against the ES approach of the ES algorithm.

6. PERFORMANCE EVALUATION

We conducted experiments to evaluate the runtime causal inference model and the top-k query processing mechanism in the proposed *RSET algorithm* and the *ES algorithm*. One evaluation was with respect to the accuracy of the top-k results, and the other evaluation was with respect to the runtime. Section 6.1 describes the experiment setup, including the evaluation measures, data sets and the platform used, and Section 6.2 presents the experiment results.

6.1. Experiment setup

6.1.1. *Performance measures*

The measures of evaluating top-k query processing are accuracy and runtime.

Accuracy measure

Suppose R_k is the ranked list of the top-k effects predicted given an EOP C_δ observed in the test sequence. Then, the accuracy can be measured by checking the next event type observed in the test sequence, E_o ($o \in [1, N]$), against the predicted ranked list R_k. If E_o exists in R_k, then we say the prediction is correct (or 'hit'), and otherwise incorrect (or 'miss').

We used two alternative methods to calculate the accuracy: *hit-or-miss* and *weighted*. The hit-or-miss method gives any hit 100% accuracy and any miss 0% accuracy, whereas the weighted method rates the accuracy of a hit according to the rank of E_o in R_k. Either method is suitable depending on the application requirements. The specific formulas can be summarized as follows:

Hit-or-miss accuracy: Let n_{hits} and n_{misses} be the number of hits and the number of misses, respectively, out of n_{tests} tests. Then, the hit-or-miss accuracy of the result, α_{h-m}, is calculated as follows:

$$\alpha_{h-m} = \frac{n_{hits}}{n_{tests}} = \frac{n_{hits}}{n_{hits} + n_{misses}} \tag{2}$$

Weighted accuracy: Suppose $P(E_o)$ is the score of E_o in R_k. As discussed earlier, the rank is based on the score; we normalize the score such that the prediction accuracy decreases gradually with the decrease in the rank of E_o in R_k. That is, in the case of a hit, the accuracy is given as $\frac{P(E_o)}{\max\{P(E_j) \mid E_j \in R_k\}}$, where the denominator is the highest probability among all event types in R_k. (Note that this measure gives the top event type the accuracy of 100%.) In the case of a miss, the accuracy of E_o is 0%.

So, given n_{hits} and n_{misses}, the weighted accuracy of the result, $\alpha_{weighted}$, is calculated as follows:

$$\alpha_{weighted} = \frac{\sum_{i=1}^{n_{tests}} \frac{P(E_{o_i})}{\max\{P(E_{j_i}) | E_{j_i} \in R_{k_i}\}}}{n_{tests}}$$

where E_{o_i} is the ith observed event type in the test sequence.

Since the accuracy of a miss is 0%, we can consider only the hits in the numerator:

$$\alpha_{weighted} = \frac{\sum_{h=1}^{n_{hits}} \frac{P(E_{oh})}{\max\{P(E_{j_h}) | E_{j_h} \in R_{k_h}\}}}{n_{hits} + n_{misses}} \tag{3}$$

where E_{o_h} is the hth observed event type in the test sequence that has a result hit.

Runtime measure

The runtime was the CPU time taken during query processing. Note that the EPN construction is not part of the query processing mechanism and, therefore, we did not include it to measure runtime. In the query processing with the RSET algorithm and the ES algorithm, there is an overhead of runtime causal inference, and so it was included in the runtime. In contrast, the query processing with the traditional causal inference (i.e. FCNI algorithm) does not include the causal network construction time in the runtime because the causal inference is performed only once (during the causal network construction) prior to query processing.

In our work, latency is the interval between the arrival of a new event and the identification of its top-k effect events. However, as the time for EPN update is insignificant (see the polynomial runtime in Section 4.2) compared to the time for query processing (which is exponential), latency is essentially the query processing time.

6.1.2. *Data sets*

Experiments were conducted using two real-world data sets (profile summary in Table 2) to evaluate the proposed algorithms.

Electric power grid data set

This data set contains simulated temporal sequences of cascading electric power grid component outages, such as those that can lead to very large blackouts (e.g. [52]). The sequences were generated using a model of the Polish power network, which is described in [53]. Each sequence represents the order in which the components failed, as well as the time of the failure. Each grid component was considered an event type, whereas a component failure was an event instance. This data set includes 4492 cascade sequences and 565 distinct event types.

In the original data set, each file, representing one blackout, has a list of the components that failed in that blackout. The original schema of the power grid data is as follows: $\langle$*event indicator*, *timestamp*, *component id*$\rangle$. The event indicator can be one of 0, 1 and -1, which refer to an initiating event, a

TABLE 2. Profiles of the data sets.

Data set	N	$N_{\text{instances}}$	CRA	N_{CRA}
Web page click	17	4 698 795	session id	989 818
Electric power grid	565	94 339	blackout id	4492

N, number of event types; $N_{\text{instances}}$, number of event instances; N_{CRA}, number of different CRA values.

dependent event and a stop event, respectively. There is always at least one *initiating event* at the beginning of each component failure sequence (with 0 as its starting time). Since these events are always at the beginning of the sequence, there is no inward edge toward them in the EPN. A *dependent event* is the result of the initiating event or another dependent event. A blackout sequence always has at least one dependent event. We treated both an initiating event and a dependent event in the same way. On the other hand, a *stop event* denotes the end of the blackout and is not a real event. Therefore, we ignored stop events. The *timestamp* and the *component id* are, respectively, the starting time of an event and the attribute that uniquely identifies a grid component.

To create an event stream, we modified the schema and mixed the data from the files in random order while preserving the temporal order of the component failures in each blackout. The modified schema, ⟨*timestamp*, *component id*, *blackout id*, *event indicator*⟩, has an additional tag *blackout id* to identify the blackout to which the component failure belongs to. So, the blackout id is the CRA in the power grid data set.

Web page click stream data set

This data set consists of click-stream data of 989 818 sequences obtained from the University of California, Irvine's machine learning repository [54]. Each sequence reflects the browsing activities, arranged in temporal order, in one user session. The data set gives a random sample of the length of visits of users browsing the msnbc.com web site on the whole day of 28 September 1999. The length of the visit is an estimate of the total number of clicks or pages seen by each user and is based on the 'Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com'. A webpage category is an event type, and a webpage visit is an event instance. The session id of the visit is the CRA for its event instance.

The number of distinct event types is 17. That is, a sequence can have web activities related to 17 different webpage categories. These event types (i.e. webpage categories) are frontpage, news, technology, local, opinion, on-air, miscellaneous, weather, MSN-news, health, living, business, MSN-sports, sports, summary, BBS and travel. The total number of event instances (i.e. page visits) is 4 698 795.

To create an event stream, we randomly mixed the events while preserving the temporal order of the events for each session. The schema of the events is ⟨*timestamp*, *webpage category*, *session id* ⟩.

6.1.3. *Experiment platform*

The experiments were conducted on 2.3 GHz Intel Core i5 machine with 4GB of memory, running Windows 7. The algorithms were implemented in Java 1.7.0.

6.2. Experiment results

We conducted two sets of experiments to evaluate the RSET and ES algorithms for top-k predictive query processing. One set of experiments was to evaluate the prediction accuracy, and the other set of experiments was to evaluate the runtime. There were two objectives in each set of experiments. The first objective was to compare the *causal inference mechanism* used in query processing—that is, between the runtime causal inference mechanism of RSET and ES algorithms and the traditional causal inference mechanism of the FCNI algorithm. For better fairness, we incorporated the RSET algorithm (as opposed to ES) into the FCNI algorithm for query processing because RSET is more efficient than ES. The second objective was to compare the query processing mechanism between the RSET algorithm and the ES algorithm. In both sets of experiments, we also studied the effect of varying the value of k on the proposed algorithms.

For each real data set, we used 70% for training and 30% for testing the proposed algorithms. The division of web page click stream data set was done by session id, and the division of electric power grid data set was done by blackout id. From the event stream in training data set, an EPN was constructed and input to the RSET and ES algorithms and a causal network was constructed and input to the FCNI algorithm. The window observation period was set to 10 msec for the web page click stream data set and to 10 sec for the electric power grid data set. Testing data simulated an event stream. As soon as a new event arrived, it was added to the partitioned window and, then, the top-k prediction query execution was triggered in response to the most recent event at position δ (called the *EOP index*) in the sequence of cause events in the same partition. Note that the RSET and ES algorithms perform query processing over an EPN whereas the FCNI algorithm does so over a causal network. Upon the arrival of a new event, the measurements of prediction accuracy and runtime were repeated and the calculated average accuracy and average runtime were reported.

6.2.1. *Accuracy*

Figures 8 and 9 show the hit-or-miss accuracy and the weighted accuracy, respectively, for the web page click stream data set, and Figs. 10 and 11 show those for the electric power grid data set. In these figures, the accuracies of the RSET algorithm, the ES algorithm and the FCNI algorithm are shown for varying value of the EOP index (δ) over the sequence of events in the condition set. Note that the EOP index δ is the position of the most recent event in the sequence of cause events in the same partition. In addition, Figs. 12 and 13 show the average hit-or-miss and weighted accuracies for different values of k in the web page click stream data set and the electric power grid data set, respectively.

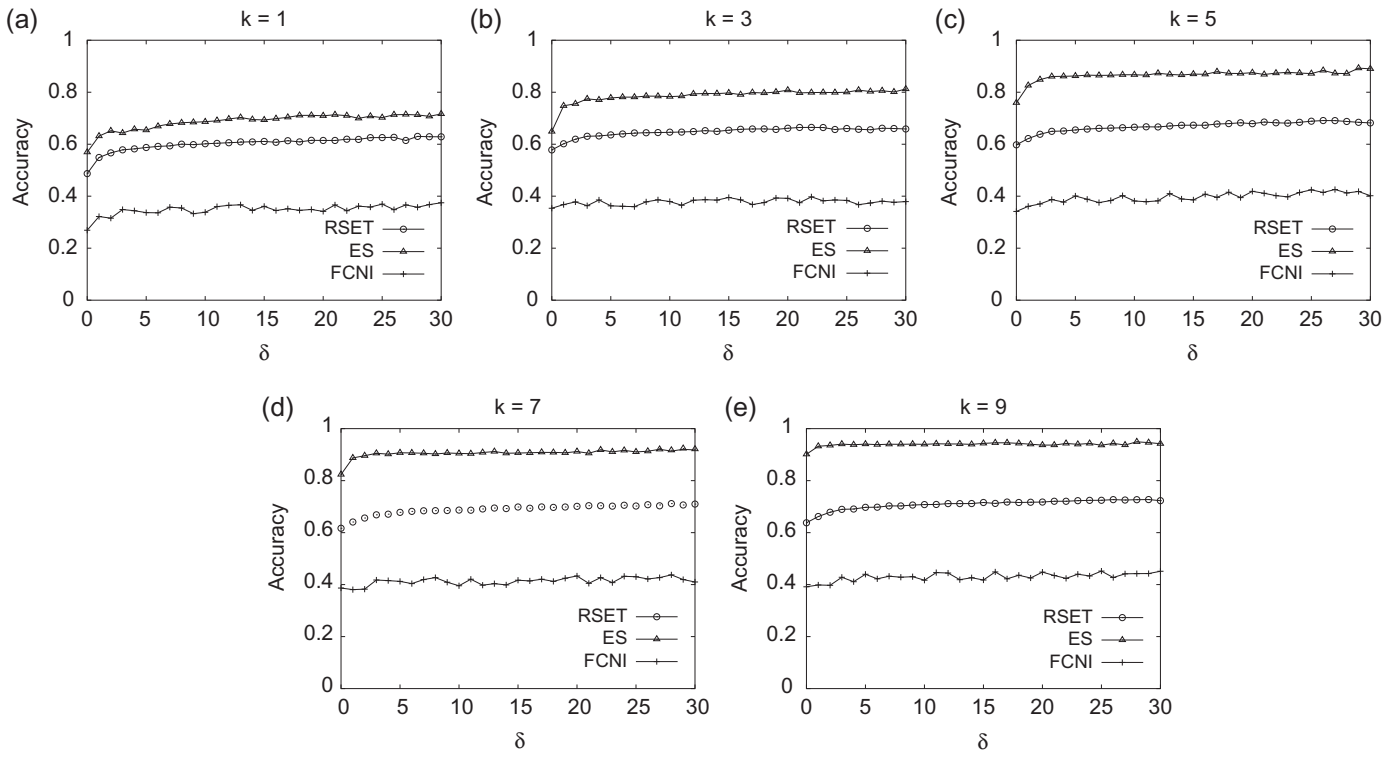

FIGURE 8. Hit-or-miss accuracies of the RSET, ES and FCNI algorithms for the web page click stream data set.

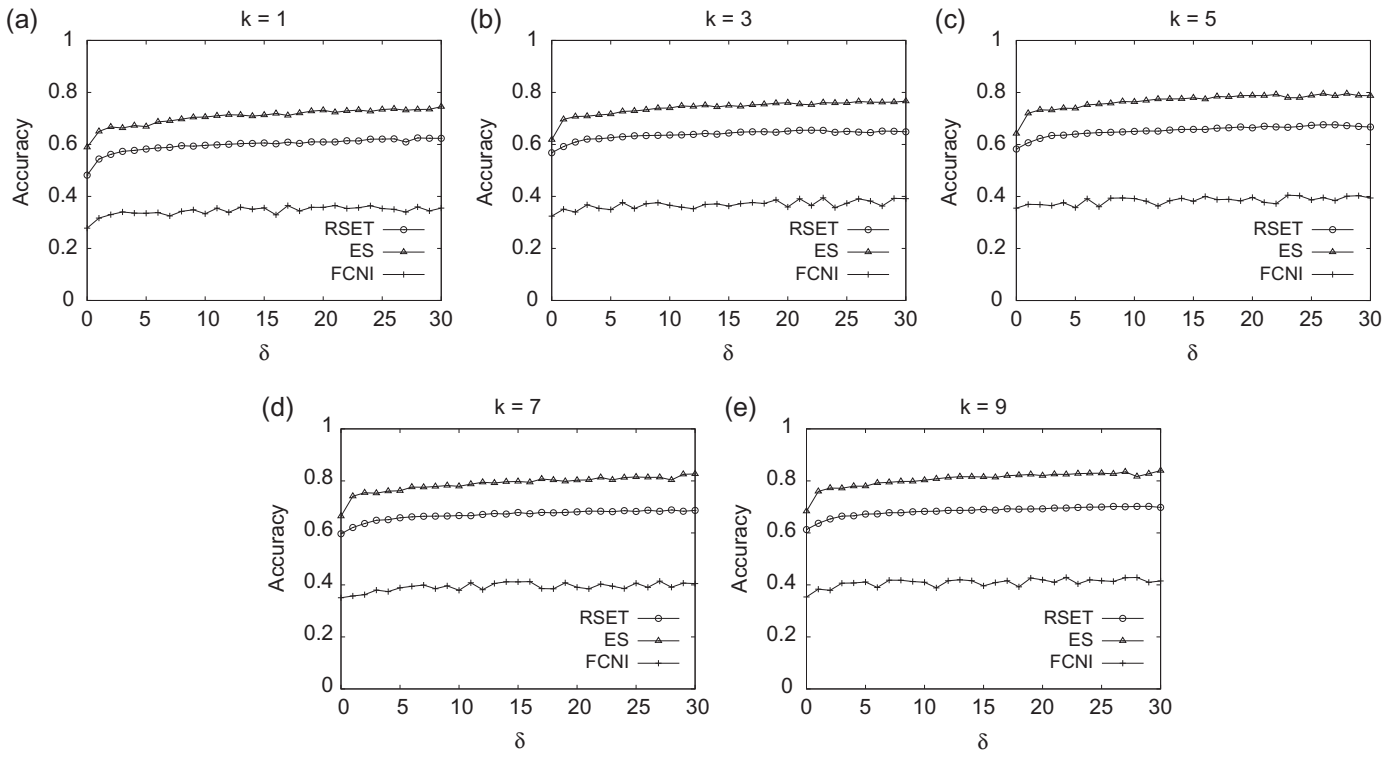

FIGURE 9. Weighted accuracies of the RSET, ES and FCNI algorithms for the web page click stream data set.

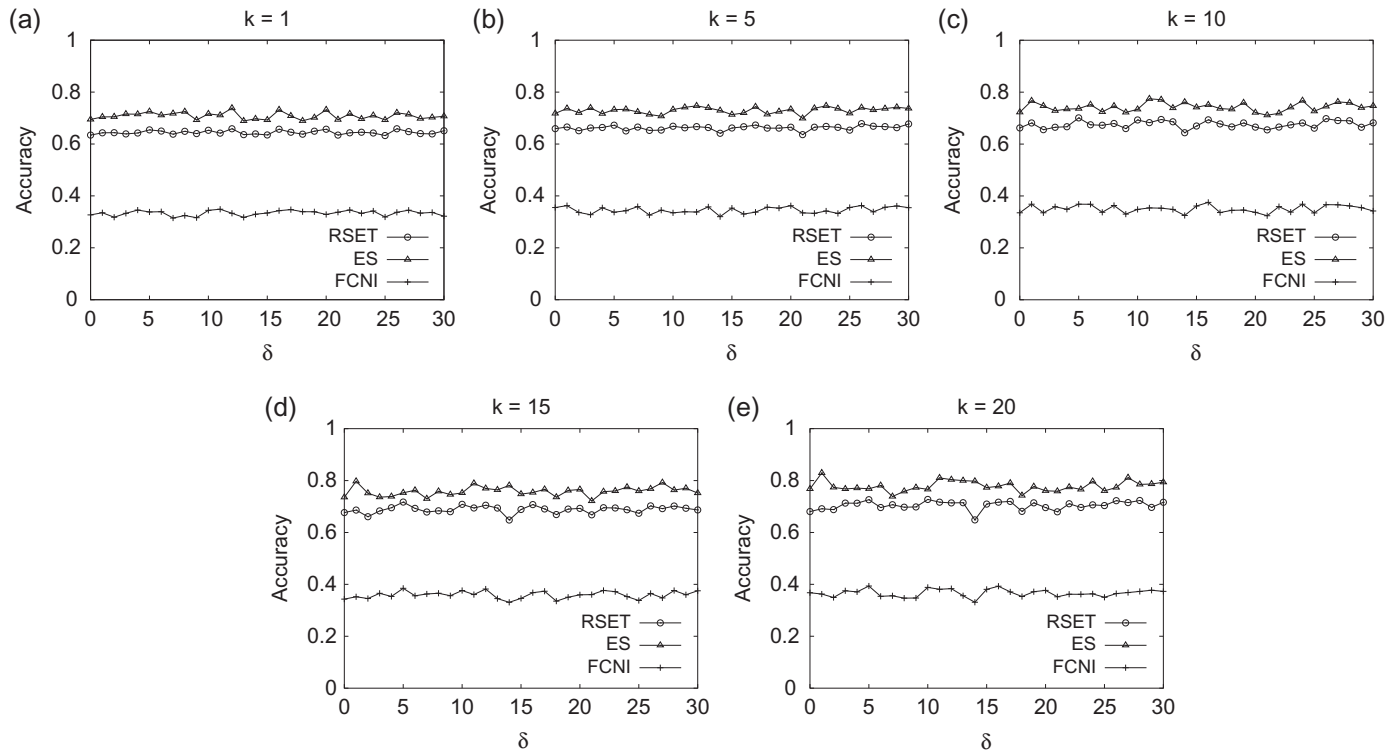

FIGURE 10. Hit-or-miss accuracies of the RSET, ES and FCNI algorithms for the electric power grid data set.

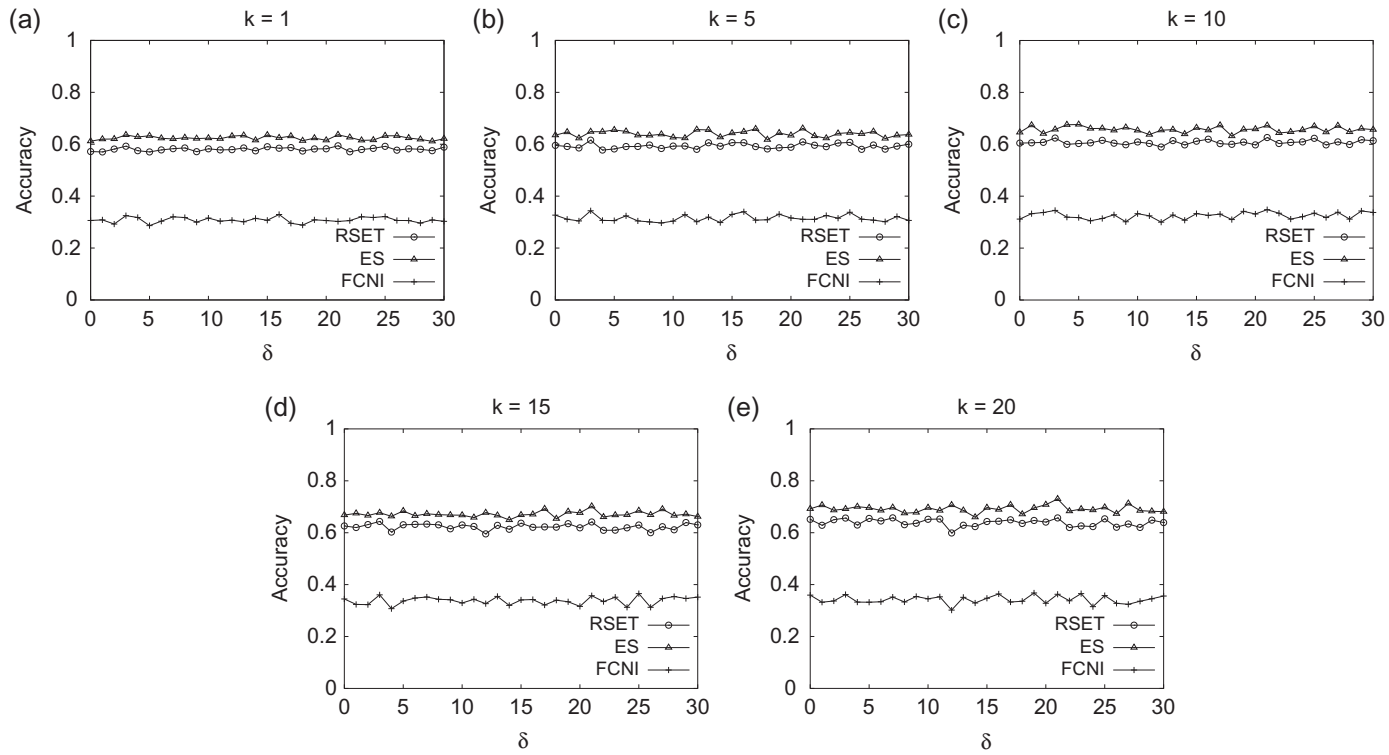

FIGURE 11. Weighted accuracies of the RSET, ES and FCNI algorithms for the electric power grid data set.

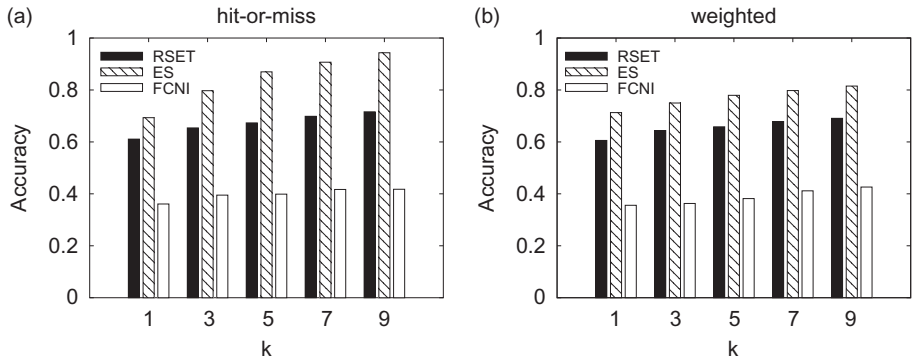

FIGURE 12. Average accuracies of the RSET, ES and FCNI algorithms w.r.t. k for the web page click stream data set.

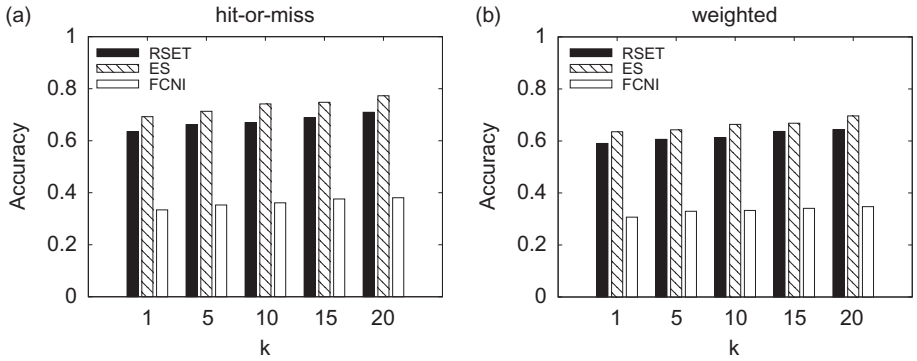

FIGURE 13. Average accuracies of the RSET, ES and FCNI algorithms w.r.t. k for the electric power grid data set.

Note that the hit-or-miss accuracy is never lower than the weighted accuracy. This always holds in all cases because in the hit-or-miss accuracy a hit always receives the score of 100% while in the weighted accuracy a hit receives a score lower than 100% unless it is the observed event type (from the test data set) that has the highest score in the ranked list. Note as well that, the two accuracy measures give the same value when k is 1 because then the size of the result ranked list is one and hence equation 3 reduces to equation 2.

Now, we discuss below our observations from comparing the accuracies of the three algorithms.

Comparison of the causal inference mechanism

All results showed that the prediction accuracies of the proposed algorithms (both ES and RSET) were significantly higher than that of the FCNI algorithm at every EOP index (δ). This difference in accuracy came from their causal models. That is, the traditional causal model of the FCNI algorithm was limited due to its lack of support for cyclic causality and due to the loss of causal information that the prediction accuracy was compromised significantly compared with that of the RSET algorithm. (Recall that, for fairness, in the FCNI algorithm we used the same query processing mechanism used in the RSET algorithm.) In comparison, the *run-time* causal inference of the RSET and ES algorithms readily supported cyclic causality and reduced causal information loss, thereby achieving higher prediction accuracies.

The results also showed that, for the same reason (i.e. causal models), the prediction accuracies of the proposed ES and RSET algorithms were always higher than that of the FCNI algorithm for different number k of predicted events. Besides the accuracy of all three algorithms increased with the increase of k. The reason is that a larger number k of predicted event types generally contributes more to the overall accuracy calculated from the resulting set of event types (see equations 2 and 3).

Comparison of the query processing mechanism

The results showed that the prediction accuracy of the ES algorithm was always higher than that of the RSET algorithm, which is evident given that the ES algorithm performs an ES whereas the RSET algorithm performs only a partial search.

It was discussed earlier why the accuracies of all three algorithms increase as the value of k increases. Here, we add further explanation on the ES algorithm and the RSET algorithm based on their search mechanisms for query processing.

In the RSET algorithm, the search space increases as k increases, therefore reducing the probability of missing out the correct top-k events. In the ES algorithm, the ES space is constant, but, still, larger k means lower probability of missing the correct top-k events as a result of the search.

6.2.2. *Runtime*

Figures 14 and 15 show the runtime for the web page click stream data set and the electric power grid data set, respectively. In these figures, the runtime of the three algorithms are compared for different values of the EOP index (δ) over the sequence of events in the condition set. In addition, Fig. 16 shows the runtime for different values of k in the web page click stream and the electric power grid data sets.

Comparison of the causal inference mechanism

The results showed that the runtimes of the RSET and ES algorithms were longer than that of the FCNI algorithm at every EOP index for every value of k. As discussed in Section 5.3, the RSET and ES algorithms have an overhead of the runtime causal inference during query processing while FCNI algorithm does not because it uses a pre-built causal network for prediction. Therefore, the runtimes for the ES and RSET algorithms are always longer than that of the FCNI algorithm.

Interestingly, the runtimes of the three algorithms were longer in the electric power grid data set than the web page click stream data set. This is because of the larger number N of event types in the electric power grid data set.

Comparison of the query processing mechanism

First, as expected, the runtime of the RSET algorithm was always shorter than that of the ES algorithm. The main reason lies in the difference in their search strategies (i.e. reduced in RSET as opposed to exhaustive in ES) as discussed in Section 5. Besides, there is an additional overhead in the ES algorithm to calculate the causal search order.

Second, the runtime reduction of the RSET algorithm from that of the ES algorithm was much larger for in the electric power grid data set than for the web page click stream data set. This demonstrates that the RSET algorithm scales up better than the ES algorithm for data sets with a larger number of event types, that is, larger EPN. The larger EPN for the electric power grid data set results in a larger search space (which is typical over event streams) for the ES algorithm and, thus, requires longer runtime for query processing. The RSET algorithm, on the other hand, is capable of significantly reducing the search space and terminating the query processing early even in the larger search space.

Third, the runtime of the ES algorithm did not change with an increase in k. The ES algorithm always runs an ES, irrespective of the value of k, and uses k only to filter out the top-k event types out of N event types at the end, which has insignificant effect on the overall runtime. On the other hand,

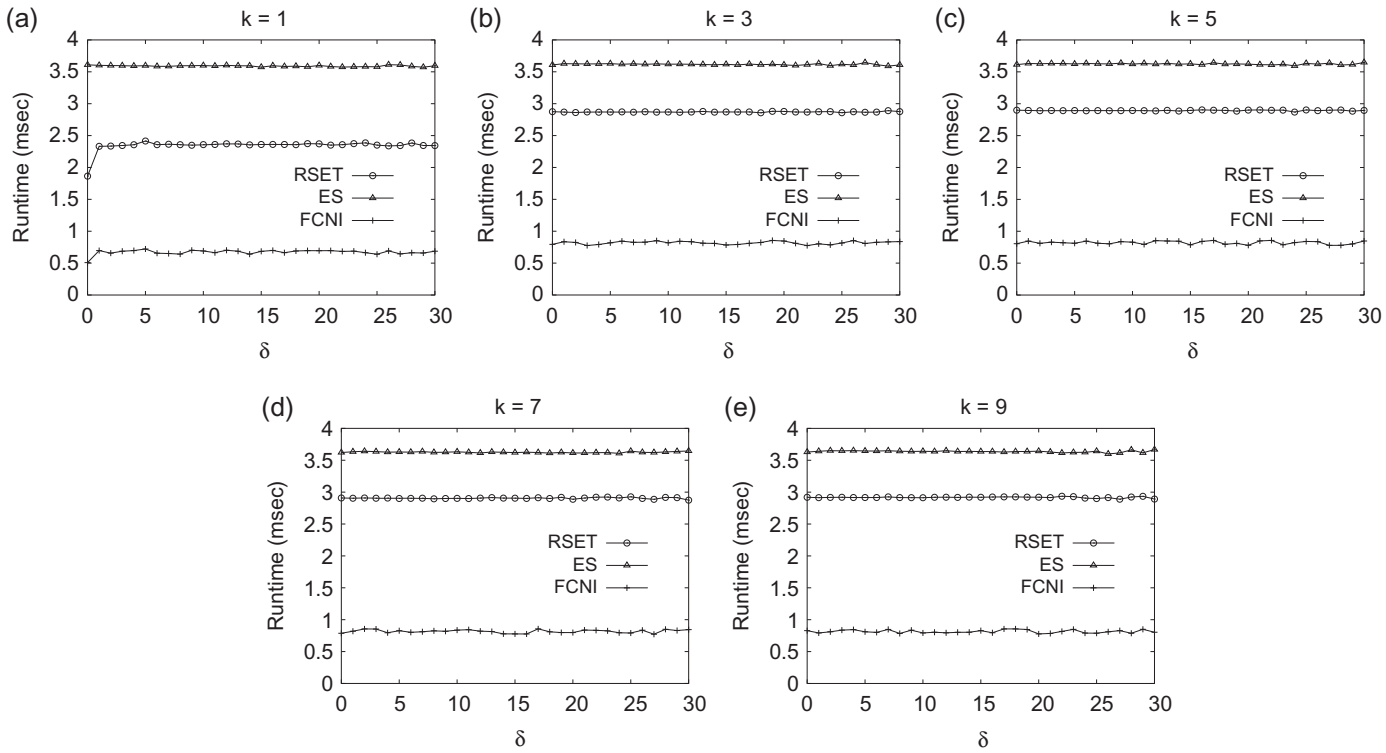

FIGURE 14. Runtime of the RSET, ES and FCNI algorithms for the web page click stream data set.

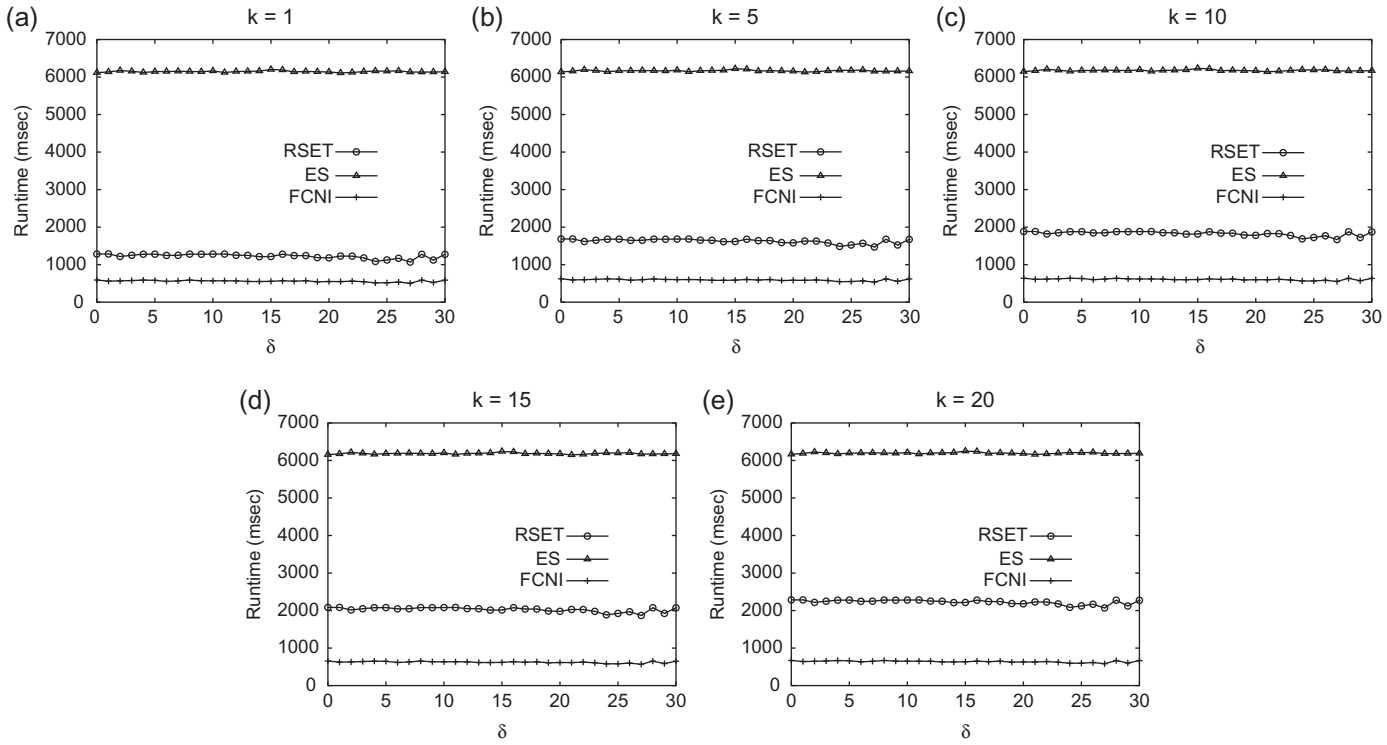

FIGURE 15. Runtime of the RSET, ES and FCNI algorithms for the electric power grid data set.

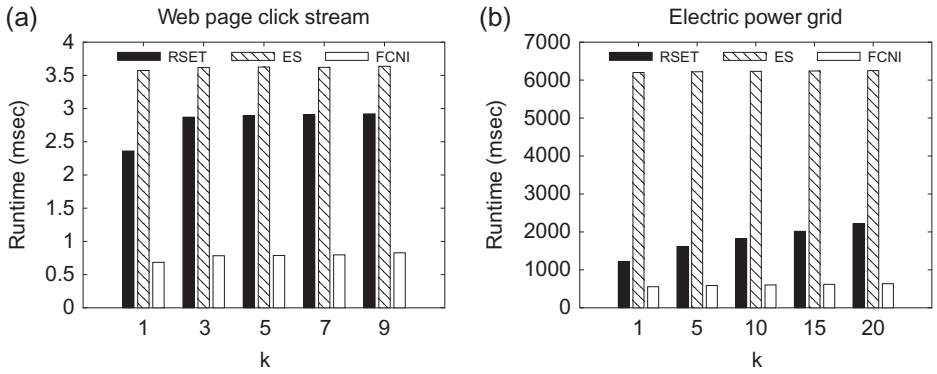

FIGURE 16. Runtime of the RSET, ES and FCNI algorithms w.r.t. *k*.

the runtime of the RSET algorithm does increase with an increase in *k*. The search space covered by the RSET algorithm increases with *k*, which leads to the increased runtime.

6.2.3. Discussion of experiment results

The proposed runtime causal inference mechanism, in the RSET and ES algorithms, handled cyclic causality and avoided causal information loss, and thus improved prediction accuracy significantly. The FCNI algorithm, on the other hand, performed worse than both the RSET and ES algorithms as it could not handle cyclic causality. The ratios of the number of cycles over the number of edges in the EPN were 0.69 for the electric power grid data set and 0.85 for the web page click stream data set. Intuitively, the accuracy of the FCNI algorithm would suffer increasingly more as the number of cycles increases. Thus, despite its reduced runtime, the FCNI algorithm is judged unsuitable for top-*k* predictive query processing over event streams.

The RSET algorithm achieved comparable accuracy as the ES algorithm, and it was much faster than the ES algorithm. Additionally, the running time of the RSET algorithm showed scaling better than the ES algorithm with an increase in the

network size. We thus conclude that the RSET algorithm is more suitable for *real-time* continuous top-k query processing over event streams; whereas the ES algorithm is more suitable when the time is of lesser importance. This is clearly evident for real-time applications with hundreds (or possibly thousands) of event types such as the electric power grid data set because then the pruning effect of reduced search and early termination becomes increasingly more significant.

Between the two data sets, the runtime for the electric power grid data set was much longer than the runtime for the web page click stream data set. This is because of the difference in the numbers of event types in these two data sets. That is, the much larger number of event types in the electric power grid data set leads to much more CI tests during the runtime causal inference, thus resulting in slower query execution. In our work, the runtime measurements were made on a low-end laptop. The use of a more powerful computational setup (e.g. parallel processing) would further reduce the runtime.

7. CONCLUSION AND FUTURE WORK

This paper addressed the problem of continuously predicting top-k effect events over event streams. We proposed a novel *run-time* causal inference mechanism to support *cyclic* causal relationships and to overcome causal information loss. Then, we proposed two query processing algorithms, the *Reduced Search with Early Termination (RSET)* algorithm and the *ES* algorithm, which use runtime causal inference to predict top-k effects continuously. Through experiments, we demonstrated that the proposed approach overcomes the two main limitations of the traditional causal inference approach—acyclic causality and causal information loss. We showed in two problem domains that the proposed RSET and ES algorithms significantly improved the causal inference power.

We plan to address a number of issues for the future work. First, in this paper, we assumed that events in a stream are always in the correct temporal order. If, however, events arrive out of order, erroneous relationships can be introduced in the EPN, thereby degrading the accuracy of prediction. One idea to deal with such an out-of-order stream is to allow for ambiguity in the edge direction by introducing, for example, undirected edges and then allowing the algorithm to resolve edge directions at query processing time. Second, in this paper, for the sake of computational efficiency, we only supported *direct* causality, therefore only one-level of prediction, under the assumption that an event is the most likely effect of its immediately preceding event. Extended from this, supporting *indirect* causality between events, thus multiple levels of causal prediction through a chain of intermediate events, would be appealing. Since the mechanism to compute the propagation of probabilities through the EPN is already in place (see Examples 5 and 6), this extension is not difficult conceptually. However, the computational cost, which increases exponentially with the number of prediction levels, is likely to be a challenge. Third, in this paper, we assumed that event types are *provided* by domain experts, but it may not be always possible. Some applications may require that the EPN constructor automatically identify and define event types from event streams. Fourth, in this paper, we focused on building a single generative prediction model to enable real-time causal modeling and causal prediction at event type level. It will be an interesting extension, especially for the web page click stream application, to personalize the prediction by modeling the causality pertaining to different users in multiple models.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful comments and recommendations. Their comments were very helpful to improving the quality of the original manuscript.

REFERENCES

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

[2] Glymour, C. (2003) Learning, prediction and causal Bayes nets. *Trends Cogn. Sci.*, **7**, 43–48.

[3] Guyon, I., Aliferis, C.F., Cooper, G.F., Elisseeff, A., Pellet, J.-P., Spirtes, P. and Statnikov, A.R. (2008) Design and Analysis of the Causation and Prediction Challenge. In *IEEE World Cong. Computational Intelligence Causation and Prediction Challenge*, Hong Kong, June, pp. 1–33. JMLR.

[4] Vaiman, M., Bell, K., Chen, Y., Chowdhury, B., Dobson, I., Hines, P., Papic, M., Miller, S. and Zhang, P. (2012) Risk assessment of cascading outages: methodologies and challenges. *IEEE Trans. Power Syst.*, **27**, 631–641.

[5] Akdere, M., Çetintemel, U. and Upfal, E. (2010) Database-support for continuous prediction queries over streaming data. *Proc. VLDB Endow.*, **3**, 1291–1301.

[6] Zhang, N.L. and Poole, D. (1996) Exploiting causal independence in Bayesian network inference. *J. Artif. Intell. Res.*, **5**, 301–328.

[7] Friedman, N., Linial, M., Nachman, I. and Pe'er, D. (2000) Using Bayesian Networks to Analyze Expression Data. In *Proc. 4th Ann. Int. Conf. Computational Molecular Biology*, New York, NY, USA, August RECOMB '00, pp. 127–135. ACM.

[8] Rottman, B.M. and Hastie, R. (2014) Reasoning about causal relationships: inferences on causal networks. *Psychol. Bull.*, **140**, 109–139.

[9] Pearl, J. (1995) Causal diagrams for empirical research. *Biometrika*, **82**, 669–688.

[10] Pearl, J. (1998) Graphs, causality, and structural equation models. *Sociol. Methods Res.*, **27**, 226–284.

[11] Spirtes, P., Glymour, C.N. and Scheines, R. (1991) From probability to causality. *Philos. Stud. Int. J. Philos. Anal. Tradit.*, **64**, 1–36.

[12] Acharya, S. and Lee, B. (2013) Fast Causal Network Inference Over Event Streams. In *Data Warehousing and Knowledge Discovery*, Berlin, Heidelberg, August, Lecture Notes in Computer Science, **8057**, pp. 222–235. Springer, Berlin, Heidelberg.

[13] Ellis, B. and Wong, W.H. (2008) Learning causal Bayesian network structures from experimental data. *J. Am. Stat. Assoc.*, **103**, 778–789.

[14] Heckerman, D. (1995) A Bayesian Approach to Learning Causal Networks. In *Proc. 11th Conf. Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, August UAI'95, pp. 285–295. Morgan Kaufmann Publishers Inc.

[15] Li, G. and Leong, T.-Y. (2009) Active Learning for Causal Bayesian Network Structure with Non-Symmetrical Entropy. In *Proc. 13th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining*, Berlin, Heidelberg, July PAKDD '09, pp. 290–301. Springer.

[16] Meganck, S., Leray, P. and Manderick, B. (2006) Learning Causal Bayesian Networks from Observations and Experiments: A Decision Theoretic Approach. *Proc. 3rd Int. Conf. Modeling Decisions for Artificial Intelligence*, Berlin, Heidelberg, April MDAI'06, pp. 58–69. Springer.

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

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

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

[20] Elloumi, M. and Zomaya, A.Y. (2013) *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, John Wiley & Sons, NJ.

[21] Kłopotek, M.A. (2006) Cyclic Bayesian network–Markov process approach. *Stud. Inform.*, **1**, 7.

[22] Tulupyev, A.L. and Nikolenko, S.I. (2005) Directed Cycles in Bayesian Belief Networks: Probabilistic Semantics and Consistency Checking Complexity. In *Proc. 4th Mexican Int. Conf. Advances in Artificial Intelligence*, Berlin, Heidelberg, November, pp. 214–223. Springer.

[23] Cheng, J. and Druzdzel, M.J. (2000) AIS-BN: an adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *J. Artif. Intell. Res. (JAIR)*, **13**, 155–188.

[24] Cheng, J. and Druzdzel, M.J. (2001) Confidence Inference in Bayesian Networks. In *Proc. 17th Conf. Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, August UAI'01, pp. 75–82. Morgan Kaufmann Publishers Inc.

[25] MacKay, D.J.C. (1999) Introduction to Monte Carlo methods. In *Learning in Graphical Models*, 175–204. MIT Press, Cambridge, MA, USA.

[26] Moral, P., Doucet, A. and Jasra, A. (2012) An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Stat. Comput.*, **22**, 1009–1020.

[27] Shachter, R.D. (1990) Evidence Absorption and Propagation Through Evidence Reversals. *Proc. 5th Ann. Conf. Uncertainty in Artificial Intelligence*, Amsterdam, The Netherlands, August UAI '89, pp. 173–190. North-Holland Publishing Co.

[28] Zhang, N.L. and Poole, D. (1994) A Simple Approach to Bayesian Network Computations. *Proc. 10th Can. Conf. Artificial Intelligence*, Banff, Alberta, May CCAA '94, pp. 171–178. Morgan Kaufmann.

[29] Henrion, M. (1991) Search-Based Methods to Bound Diagnostic Probabilities in Very Large Belief Nets. *Proc. 7th Conf. Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, July UAI'91, pp. 142–150. Morgan Kaufmann Publishers Inc.

[30] Bucklin, R.E. and Sismeiro, C (2003) A model of web site browsing behavior estimated on clickstream data. *J. Mark. Res.*, **40**, 249–267.

[31] Bollen, J., Van de Sompel, H., Hagberg, A., Bettencourt, L., Chute, R., Rodriguez, M.A. and Balakireva, L. (2009) Clickstream data yields high-resolution maps of science. *PLoS One*, **4**, e4803.

[32] Frias-Martinez, E. and Karamcheti, V. (2002) A Prediction Model for User Access Sequences. *Proc. 4th Int. Workshop on Web Mining for Usage Patterns and User Profiles*, Edmonton, Canada, July. ACM.

[33] Gündüz, S. and Özsu, M.T. (2003) A Web Page Prediction Model Based on Click-Stream Tree Representation of User Behavior. *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Washington, D.C., August, pp. 535–540. ACM.

[34] Lu, L., Dunham, M. and Meng, Y. (2005) Mining Significant Usage Patterns from Clickstream Data. *Proc. 7th Int. Workshop on Knowledge Discovery on the Web*, Chicago, IL, August, pp. 1–17. Springer.

[35] Montgomery, A.L., Li, S., Srinivasan, K. and Liechty, J.C. (2004) Modeling online browsing and path analysis using clickstream data. *Market. Sci.*, **23**, 579–595.

[36] Jiang, X.-R. and Gruenwald, L. (2005) Microarray gene expression data association rules mining based on BSC-tree and FIS-tree. *Data Knowl. Eng.*, **53**, 3–29.

[37] Rudin, C., Letham, B., Salleb-Aouissi, A., Kogan, E. and Madigan, D. (2011) Sequential Event Prediction with Association Rules. *Proc. 24th Ann. Conf. Learning Theory*, Budapest, Hungary, June COLT '11, pp. 615–634. Journal of Machine Learning Research.

[38] Veloso, A.A., Almeida, H.M., Gonçalves, M.A. and Meira, W. Jr. (2008) Learning to Rank at Query-time Using Association Rules. *Proc. 31st Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, July SIGIR '08, pp. 267–274. ACM.

[39] Bowes, J., Neufeld, E., Greer, J. and Cooke, J. (2000) A comparison of association rule discovery and bayesian network causal inference algorithms to discover relationships in discrete data. In Hamilton, H. Advances in Artificial Intelligence, *Lecture Notes in Computer Science, 1822*, 326–336. Springer, Berlin Heidelberg.

[40] Lin, T.Y., Xie, Y., Wasilewska, A. and Liau, C.-J. (2008) *Data Mining: Foundations and Practice*, Springer, Berlin, Heidelberg. Studies in Computational Intelligence, **118**.

[41] Mazlack, L.J. (2004) Mining Causality from Imperfect Data. *Proc. 6th Int. FLINS Conf. Applied Computational Intelligence Proceedings*, Belgium, September Applied Computational Intelligence, pp. 155–160. World Scientific Publishing Company.

[42] Mohammad, Y. and Nishida, T. (2010) Mining causal relationships in multidimensional time series. In Szczerbicki, E. and Nguyen, N. *Smart Information and Knowledge Management*, 309–338. Springer, Berlin Heidelberg. Studies in Computational Intelligence, **260**.

[43] Silverstein, C., Brin, S., Motwani, R. and Ullman, J. (2000) Scalable techniques for mining causal structures. *Data Min. Knowl. Discov.*, **4**, 163–192.

[44] Young, S.S. and Karr, A. (2011) Deming, data and observational studies. *Significance*, **8**, 116–120.

[45] Haghani, P., Michel, S. and Aberer, K. (2010) The Gist of Everything New: Personalized Top-k Processing over Web 2.0 Streams. *Proc. 19th ACM Int. Conf. Information and Knowledge Management*, New York, NY, USA, October CIKM '10, pp. 489–498. ACM.

[46] Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X. and Weikum, G. (2008) Efficient Top-k Querying over Social-tagging Networks. *Proc. 31st Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, July SIGIR '08, pp. 523–530. ACM.

[47] de Campos, L.M. (2006) A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *J. Mach. Learn. Res.*, **7**, 2149–2187.

[48] Bishop, Y.M., Fienberg, S.E. and Holland, P.W. (1975) *Discrete Multivariate Analysis: Theory and Practice*, MIT Press, Cambridge, MA.

[49] Kullback, S. (1968) *Information Theory and Statistics* (2nd edn). Dover Publication, NY.

[50] Zar, J. (1984) *Biostatistical Analysis* (2nd edn). Prentice Hall International, New Jersey.

[51] Kemeny, J. and Snell, J. (1969) Finite Markov chains*, repr edition University Series in Undergraduate Mathematics*, VanNostrand, New York.

[52] US-Canada Power System Outage Task Force. (2004) Final Report on the August 14, 2003 Blackout in the United States and Canada. Technical Report. US Department of Energy, Washington, D.C.

[53] Eppstein, M. and Hines, P. (2012) A 'Random Chemistry' algorithm for identifying collections of multiple contingencies that initiate cascading failure. *IEEE Trans. Power Syst.*, **27**, 1698–1705.

[54] Heckerman, D. (1999). UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data.