Graph Dependency Construction Based on Interval-based Event Dependencies Detection in Data Streams

Marc Plantevit Céline Robardet Vasile-Marian Scuturici

Abstract

Pattern mining over data streams is critical to a variety of applications such as understanding and predicting weather phenomena or outdoor surveillance. Most of the current techniques attempt to discover relationships between time-point events but are not practical for discovering dependencies over interval-based events. In this work, we present a new approach to mine dependencies between streams of interval-based events that links two events if they occur in a similar manner, one being often followed by the other one after a certain time interval in the data. The proposed method is robust to temporal variability of events and determines the most appropriate time intervals whose validity is assessed by a χ^2 test. As several intervals may redundantly describe the same dependency, the approach retrieves only the most specific intervals with respect to a dominance relationship over temporal dependencies, and thus avoids the classical problem of pattern flooding in data mining. The TEDDY algorithm, TEmporal Dependency DiscoverY, prunes the search space while guaranteeing the discovery of all valid and significant temporal dependencies. We present empirical results on simulated and real-life data to show the scalability and the robustness of our approach. The dependency relationships define a graph that supports intelligent analysis as illustrated by two case studies: Outdoor surveillance of a building via video camera and motion sensors, and assistance for road deicing operations based on the humidity and temperature measurements at the urban scale. These applications demonstrate the efficiency and the effectiveness of our approach.

1 Introduction

Recent breakthroughs in sensor technology have given users the ability to monitor many events in real time producing multiple heterogeneous data streams. This novel context has been a source of motivation for the development of many data stream management and analysis techniques [1], and the extension of classical pattern mining techniques (e.g., frequent itemsets [10, 19, 26], multidimensional data [23], sequences [11, 12, 34], multidimensional sequences [42] and

graphs [2, 17]) to tackle the new challenges faced in this context [27]: Handling infinite sequences of events occurring at a steady pace to provide actionable insights to end-users. Since the mining step has to be faster than the data acquisition process, it is not possible to store data streams in their entirety and then perform various scans over them. For this reason, data stream mining has been widely recognized as an important research area with many applications, such as the comprehension and prediction of weather phenomena, anomaly detection in outdoor surveillance, or mining health monitoring streams, to mention just a few.

The constant evolution in hardware and software technology has made it possible for companies to generate very large volumes of information from various data sources. As an example, smart environments equipped with different kinds of sensors (e.g., cameras, badge readers, motion sensors, automatic doors), that act as data sources generating several data streams at high speed. In this paper we address the original problem of identifying temporal dependencies between streams of interval-based events (i.e., events having a duration and described as a collection of disjoint intervals). Two events are linked if the intervals of one is repeatedly followed by the appearance of the intervals of the other one, in a certain time delay. Such dependencies constitute key actionable insights for timely challenges such as smart environment monitoring, partial failure detection, or object tracking between various cameras. The discovered dependencies can be used to construct a graph that supports intelligent data analysis.

Considering events described by time intervals makes it possible to improve existing time-point based approaches by (1) better handling events that are rare but occur for a long period of time; (2) being more robust to the temporal variability of events; (3) allowing the discovery of sophisticated relations based on Allen's algebra [5].

Our approach to detect interval-based event dependencies is illustrated in Figure 1. First, data streams are generated from the monitoring of a *smart environment*. The data streams produced by each sensor are transformed into *sequences of intervals* that correspond to a sensor state. Our method aims to identify pairs of sensor states that are in *temporal dependence*. Such a dependence occurs when the sequences of intervals can be translated so that their interval set intersection is sufficiently large not to be due to chance. Interval sequences are translated using a time-delay interval that is automatically determined to maximize the proportion of time where the two event intervals intersect. Several possible time-delay intervals are generated and a Pearson's χ^2 *test* of independence is used to determine whether or not the time-delay gives rise to statistically dependent sequences. As several intervals may redundantly describe the same dependency, the approach retrieves only the most specific ones with respect to a *dominance test*. Thus, it avoids the classical problem of pattern flooding in data mining. Discovering all valid and significant temporal dependencies is challenging since, for every couple of events, all possible time-delay intervals have to be considered. Therefore, we propose an efficient algorithm TEDDY, TEmporal Dependency DiscoverY, that benefits from several properties in order to prune the search space while guaranteeing the complete-

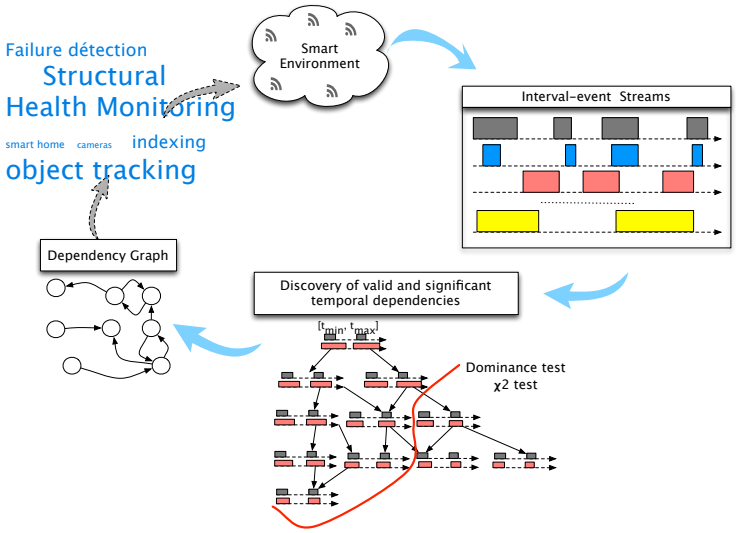


Figure 1: Overview TEDDY approach.

ness of the extraction. The extracted dependencies can be directly exploited by the end-users, or be used to construct a *dependency graph*. Such a graph can then be analyzed to understand the temporal relationships between sensors. We illustrate this intelligent data management on two case-studies based on real-world sensor networks:

Outdoor surveillance: Video camera and motion sensors can be used to monitor events that occur around buildings. The camera are often equipped with infrared LEDs, used to illuminate the scene at night. This technology produces heat which attracts all sorts of insects, especially spiders. This often leads to the appearance of spider's webs that partially obstruct the view of the camera and blur the captured images. This phenomenon, which may happen very quickly, is even more important when it rains, when water droplets hanging from the web cause serious obstruction of the camera view. We show, in Section 5.1, how temporal dependencies can be used to automatically detect cameras that are prone to this phenomenon.

Road deicing operation assistance: Every winter, deicing or snow-clearing of roads is an important issue that impacts on the local economy, the public finances and the environment. Deicing of roads must therefore be well-organized in order to limit its negative environmental, technical and health impacts. Road operators rely on weather forecasting. However, weather alerts are on the scale of an entire urban area whereas topographic (e.g., hill) and urban (e.g. parks, buildings) disparities can cause differences

in temperature and freezing. Hence, many roads are processed without this being necessary and some slick roads are not deiced. To enhance deicing operation management, the Lyon urban area (Grand Lyon, France) has equipped the roads with sensors that measure the humidity and the temperature in real-time. These sensors are embedded in the bitumen and monitor short-lived atmospheric phenomena. In Section 5.2, we use our approach to identify the temporal dependencies among freezing events and show how they can be used to implement road deicing operations.

Contributions and Roadmap

To summarize, the main contributions of this paper are:

- The introduction of the problem of temporal dependency discovery between sequences of interval-based events (see Section 2). We formally define the notion of confidence and assess its validity based on a χ^2 test. We present a dominance relationship on temporal dependencies that makes it possible to control the redundancy among the patterns.
- The design of an efficient algorithm, presented in Section 3, that benefits from various properties to prune the search space while guaranteeing the completeness of the extraction.
- An evaluation of the efficiency of the algorithm on synthetic data (Section 4).
- The investigation of two case-studies in Sections 5.1 and 5.2 that illustrate the applicability of the mined temporal dependencies to generate a graph that support intelligent data analysis on real data streams from two smart real-world environments equipped with sensors.

2 Temporal dependencies

Data streams are sequences of time-point events, $S =< (a,t) >$, that is to say sequences of couples made of a nominal symbol $a \in \mathcal{A}$, and a time stamp $t \in T_s$, with T_s the discrete time of observation. For example in Figure 2, $\mathcal{A} = \{open, close\}$ and the time-point events are $< (open,1), (close,2), \cdots, (close,9) >$. But, in many application domains, it is the time interval between time-point events that conveys the most valuable information. For example, the time intervals during which a door is open may be in temporal dependency with the detection of a moving object by a camera. Therefore, it can be interesting to examine the intervals associated to these events. A point-based event sequence S is turned into as many *interval-based event sequences* as there are symbols $a \in \mathcal{A}$. The resulting interval-based sequences are denoted by capital letter, e.g. the event a is associated to the interval-based sequence A, and is defined by:

$$A =< [t_i, t_{i+1}) \mid t_i, t_{i+1} \in T_s > \text{ where } \forall t \in ([t_i, t_{i+1}) \cap T_s)\,,\ (a,t) \in S$$

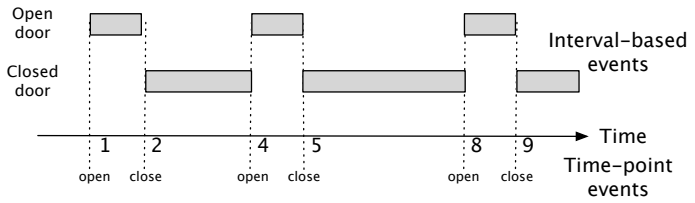


Figure 2: *open* and *close* events generated by a door sensor and their related set of intervals.

Following the example in Figure 2, the interval set associated to the event *open* is *Open door* $=< [1,2),\ [4,5),\ [8,9) >$ and the one associated to *close* is *Closed door* $=< [2,4),\ [5,8) >$. The significance of an interval-based event, called *event* hereafter, is evaluated by the sum of the lengths of its intervals:

$$\mathbf{len}(A) = \sum_{[t_i,t_{i+1})\in A} (t_{i+1} - t_i)$$

In Figure 2, $\mathbf{len}(\textit{Open door}) = 3$. The dependency of two events A and B is evaluated on the basis of the intersection of their intervals: $\mathbf{len}(A \cap B) = \mathbf{len}(< [t_i, t_{i+1}) \cap [t_j, t_{j+1}) >)$ with $[t_i, t_{i+1}) \in A$ and $[t_j, t_{j+1}) \in B$).

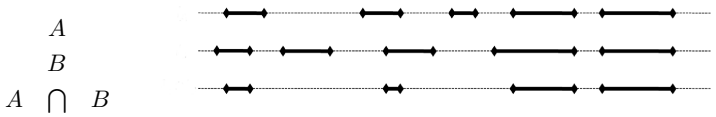


Figure 3: Example of intersection of interval sets.

Figure 3 provides an example of the intersection of two events. However, two events A and B can be in temporal dependency $A \rightarrow B$ while not being synchronous. It happens when B is time-delayed with respect to A. To capture such dependencies the intervals of B may undergo some transformations so as to better coincide with the intervals of A. Two types of transformations can be applied: (1) B can be shifted so as to maximize its intersection with A, and (2) B can be slightly extended so as to make the temporal dependency measure more robust to the inherent variability of the data. Shifting B by β consists of translating each interval of B by β time units: $B^{[\beta,\beta]} = \{[t_j - \beta, t_{j+1} - \beta) \mid [t_j, t_{j+1}) \in B\}$. To slightly extend the intervals of B, the second bound of each interval of B is translated by only β time units, with $\alpha \geq \beta \geq 0$. It results in the following new interval set: $B^{[\alpha,\beta]} = \{[t_j - \alpha, t_{j+1} - \beta) \mid [t_j, t_{j+1}) \in B\}$. Notice that the intervals of $B^{[\alpha,\beta]}$ may intersect. In that case, intersecting intervals are merged. Figure 4 illustrates some interval set shifts.

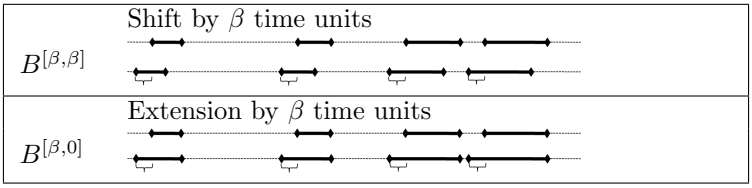


Figure 4: Example of interval set shifts.

2.1 Temporal dependency assessment

The temporal dependency of A over B is evaluated by the proportion of time where the two events simultaneously occur over the length of A. This confidence measure is formally defined below.

Definition 1 (Confidence of event dependency) *Considering two events A and B as well as a shifted interval $[\alpha, \beta]$, the strength of a $[\alpha, \beta]$-temporal dependency between A and B, denoted $A \xrightarrow{[\alpha,\beta]} B$, is evaluated by:*

$$\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B) = \frac{\mathbf{len}(A \cap B^{[\alpha,\beta]})}{\mathbf{len}(A)}$$

We can observe that $\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)$ is equal to 1 iff each interval of A is included in an interval of $B^{[\alpha,\beta]}$. Some values of $[\alpha, \beta]$ convey some specific semantics of the confidence measure. For instance, $A \xrightarrow{[0,0]} B$ highlights a *simultaneous* dependency between A and B. $A \xrightarrow{[\beta,\beta]} B$ means that events A and B are in a relation *after exactly* β time stamps, and $A \xrightarrow{[\beta,0]} B$ means that A is in the relation *after at most* β time stamps with event B. In Figure 5, $\mathbf{conf}(A \xrightarrow{[0,0]} B) = \frac{2+3+5+3}{2+4+5+3} = \frac{13}{14}$ and $\mathbf{conf}(A \xrightarrow{[1,2]} B) = \frac{2+2+4+2}{2+4+5+3} = \frac{10}{14}$.

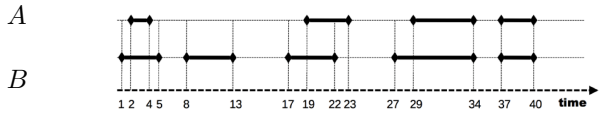


Figure 5: Example of confidence measure evaluation.

To statistically assess the value of $\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)$, we propose to perform a Pearson's chi-squared test of independence [41]. The test determines whether or not the occurrences of A and $B^{[\alpha,\beta]}$ are statistically independent over the period of observation T defined by $T = [t_{begin}, t_{end})$ with

$$\begin{aligned} t_{begin} &= \min\{\min_{[t_i,t_{i+1})\in A} t_i, \min_{[t_j,t_{j+1})\in B} t_j\} \text{ and} \\ t_{end} &= \max\{\max_{[t_i,t_{i+1})\in A} t_{i+1}, \max_{[t_j,t_{j+1})\in B} t_{j+1}\} \end{aligned}$$

A given time point of T might belong or not to an interval of A. These two possible outcomes are denoted $\mathbf{A}$ and $\overline{\mathbf{A}}$. Table 1 is the contingency table O that crosses the observed outcomes of A and $B^{[\alpha,\beta]}$. The null hypothesis states

	$\mathbf{B}^{[\alpha,\beta]}$	$\overline{\mathbf{B}^{[\alpha,\beta]}}$
$\mathbf{A}$	$\mathbf{len}(A \cap B^{[\alpha,\beta]})$	$\mathbf{len}(A) - \mathbf{len}(A \cap B^{[\alpha,\beta]})$
$\overline{\mathbf{A}}$	$\mathbf{len}(B^{[\alpha,\beta]}) - \mathbf{len}(A \cap B^{[\alpha,\beta]})$	$\mathbf{len}(T) - \mathbf{len}(A) - \mathbf{len}(B^{[\alpha,\beta]}) + \mathbf{len}(A \cap B^{[\alpha,\beta]})$

Table 1: Matrix O partitions the interval T into the four possible outcomes of A and B.

that the occurrences of the outcomes $\mathbf{A}$ and $\mathbf{B}^{[\alpha,\beta]}$ are statistically independent. If we suppose that $\mathbf{A}$ occurs uniformly over T, there are $\frac{\mathbf{len}(A)}{\mathbf{len}(T)}$ chances that event $B^{[\alpha,\beta]}$ occurs at the same time. As $\mathbf{B}^{[\alpha,\beta]}$ occurs during $\mathbf{len}(B^{[\alpha,\beta]})$ time stamps, the expected number that $\mathbf{B}^{[\alpha,\beta]}$ occurs simultaneously with $\mathbf{A}$ under the null hypothesis is $\frac{\mathbf{len}(B^{[\alpha,\beta]}) \times \mathbf{len}(A)}{\mathbf{len}(T)}$. The three others outcomes under the null hypothesis are constructed according to the same principle. All these expected outcomes E are given in table 2.

	$\mathbf{B}^{[\alpha,\beta]}$	$\overline{\mathbf{B}^{[\alpha,\beta]}}$
$\mathbf{A}$	$\frac{\mathbf{len}(B^{[\alpha,\beta]}) \times \mathbf{len}(A)}{\mathbf{len}(T)}$	$\frac{\left(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]})\right) \times \mathbf{len}(A)}{\mathbf{len}(T)}$
$\overline{\mathbf{A}}$	$\frac{\mathbf{len}(B^{[\alpha,\beta]}) \times (\mathbf{len}(T) - \mathbf{len}(A))}{\mathbf{len}(T)}$	$\frac{\left(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]})\right) \times (T - \mathbf{len}(A))}{\mathbf{len}(T)}$

Table 2: Matrix E partitions the interval T into the four possible outcomes of A and $B^{[\alpha,\beta]}$ under the null hypothesis.

The value of the statistical test is

$$\begin{aligned} X^2 &= \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \\ &= \frac{\mathbf{len}(T)\left(\mathbf{len}(T)\,\mathbf{len}\left(A \cap B^{[\alpha,\beta]}\right) - \mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})\right)^2}{\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})(\mathbf{len}(T) - \mathbf{len}(A))(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]}))} \end{aligned} \quad (1)$$

The null distribution of the statistic is approximated by the χ^2 distribution with 1 degree of freedom, and for a significance level of 5%, the critical value is equal to $\chi^2_{0.05} = 3.84$. Consequently, X^2 has to be greater than 3.84 to establish that the interval set intersection is sufficiently large not to be due to chance. From equation (1) we derive the following quadratic equation in $\mathbf{len}\left(A \cap B^{[\alpha,\beta]}\right)$:

$$\begin{aligned} &\left(\mathbf{len}(T)\,\mathbf{len}\left(A \cap B^{[\alpha,\beta]}\right) - \mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})\right)^2 \geq \\ &\frac{3.84}{\mathbf{len}(T)}\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})(\mathbf{len}(T) - \mathbf{len}(A))(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]})) \end{aligned}$$

which is satisfied iff $0 \leq \mathbf{len}\left(A \cap B^{[\alpha,\beta]}\right) \leq \cap_1$ or $\mathbf{len}(T) \geq \mathbf{len}\left(A \cap B^{[\alpha,\beta]}\right) \geq \cap_2$, $\cap_1$ and $\cap_2$ being the roots[1] of this equation.

Intersection values that range between 0 and $\cap_1$ are much smaller than the one expected under the null hypothesis. Such values can be used to detect anomalies, but, in the following we focus on the intersection values that are unexpectedly high. Therefore, we conclude that a temporal dependency $A \xrightarrow{[\alpha,\beta]} B$ is valid iff

$$\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B) \geq \frac{\cap_2}{\mathbf{len}(A)} \tag{2}$$

As the χ^2 test only works well when the dataset is large enough, we use the conventional rule of thumb [41] that enforces all the expected numbers (cells in Table 2) to be greater than 5. Otherwise, the dependency is not considered.

2.2 Significant temporal dependencies selection

For two events in temporal dependency, a huge number of shifting intervals $[\alpha, \beta]$ may exist that result in valid temporal dependencies. These intervals may describe distinct temporal dependencies (e.g., different paths may exist between two motion captors producing as many temporal dependencies), but they can also be redundant, depicting the same phenomenon several times. Redundancy between shifting intervals is mainly caused by the following property:

Property 1 (Confidence monotonicity) *Let A and B be two events and $[\alpha_1, \beta_1]$, $[\alpha_2, \beta_2]$ be two shifting intervals. If $[\alpha_1, \beta_1] \subseteq [\alpha_2, \beta_2]$, then* $\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B) \leq \mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)$.

Proof 1 *$[\alpha_1, \beta_1] \subseteq [\alpha_2, \beta_2]$ implies that for each interval $[t_i, t_{i+1})$ in B, $[t_i - \alpha_1, t_{i+1} - \beta_1) \subseteq [t_i - \alpha_2, t_{i+1} - \beta_2)$ and $\mathbf{len}(B^{[\alpha_1,\beta_1]}) \leq \mathbf{len}(B^{[\alpha_2,\beta_2]})$. As a result $\mathbf{len}(B^{[\alpha_1,\beta_1]} \cap A) \leq \mathbf{len}(B^{[\alpha_2,\beta_2]} \cap A)$ and $\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B) \leq \mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)$.*

□

To be useful in real data streams, we wish our mining process to automatically discover the shifted intervals that best describe the temporal dependencies of two given events while avoiding the pattern flooding that may result from

[1]

$$\cap_1 = \frac{\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]}) - \sqrt{\frac{3.84}{\mathbf{len}(T)}\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})\left(\mathbf{len}(T) - \mathbf{len}(A)\right)\left(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]})\right)}}{\mathbf{len}(T)}$$

$$\cap_2 = \frac{\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]}) + \sqrt{\frac{3.84}{\mathbf{len}(T)}\mathbf{len}(A)\mathbf{len}(B^{[\alpha,\beta]})\left(\mathbf{len}(T) - \mathbf{len}(A)\right)\left(\mathbf{len}(T) - \mathbf{len}(B^{[\alpha,\beta]})\right)}}{\mathbf{len}(T)}$$

the computation of all valid temporal dependencies. Among all the shifting intervals included in $[t_{min}, t_{max}]$ that lead to valid temporal dependencies, those that are of interest should have (1) a high confidence value and (2) be as specific as possible with respect to the inclusion relation. This leads to the following definition of the *dominance* relationship on the set of temporal dependencies.

Definition 2 (Dominance relationship) *Let* $d_{[\alpha_1,\beta_1]} = A \xrightarrow{[\alpha_1,\beta_1]} B$ *and* $d_{[\alpha_2,\beta_2]} = A \xrightarrow{[\alpha_2,\beta_2]} B$ *be two temporal dependencies between* A *and* B*. We say that* $d_{[\alpha_1,\beta_1]}$ *dominates* $d_{[\alpha_2,\beta_2]}$*,* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha_2,\beta_2]}$*, iff* $[\alpha_1,\beta_1] \subseteq [\alpha_2,\beta_2]$ *and*

$$1 - \frac{\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)} < 1 - \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})} \tag{3}$$

The rationale behind this definition is that when $[\alpha_1, \beta_1]$ dominates $[\alpha_2, \beta_2]$, the loss of the confidence measure of $[\alpha_1, \beta_1]$ is less than the reduction of its interval set length and thus $B^{[\alpha_2,\beta_2]\setminus[\alpha_1,\beta_1]} \cap A$ is almost empty. Indeed, if the reduction of the interval length of $B^{[\alpha,\beta]}$ is uniformly distributed over $[t_{begin}, t_{end}]$, then the length of its intersection with A will be reduced by the same proportion:

$$\frac{\mathbf{len}(A \cap B^{[\alpha_1,\beta_1]})}{\mathbf{len}(A \cap B^{[\alpha_2,\beta_2]})} \approx \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})}$$

However, if the reduction of the interval set length of $B^{[\alpha,\beta]}$ mainly occurs when A is not active, then the length of its intersection with A decreases less than its active interval set length:

$$1 - \frac{\mathbf{len}(A \cap B^{[\alpha_1,\beta_1]})}{\mathbf{len}(A \cap B^{[\alpha_2,\beta_2]})} < 1 - \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})}$$

This dominance relationship makes it possible to refine an interval while controlling the loss of the confidence measure. If an interval reduction leads to a significant loss, then the refinement process has to be stopped, since the portion of A not covered by the interval will not be subsequently either. Therefore, significant temporal dependencies are the most specific temporal dependencies that dominate all their supersets:

Definition 3 (Significant temporal dependencies) *For two events* A *and* B*, let* Σ *be the set of temporal dependencies* $d_{[\alpha,\beta]} = A \xrightarrow{[\alpha,\beta]} B$ *such that (i)* $d_{[\alpha,\beta]}$ *dominates all of its supersets, and (ii) every superset of* $d_{[\alpha,\beta]}$ *dominates its supersets as well:*

$$\begin{aligned}\Sigma = \{d_{[\alpha_1,\beta_1]} \mid \quad & \forall \quad [\alpha_2,\beta_2] \text{ such that } [\alpha_1,\beta_1] \subseteq [\alpha_2,\beta_2],\ d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha_2,\beta_2]} \\ \text{and} \quad & \forall \quad [\alpha_3,\beta_3] \text{ such that } [\alpha_2,\beta_2] \subseteq [\alpha_3,\beta_3],\ d_{[\alpha_2,\beta_2]} \preceq d_{[\alpha_3,\beta_3]}\}\end{aligned}$$

Temporal dependencies that belong to the positive border of $(\Sigma, \preceq)$ *are said to be significant.*

Property 2 (Σ-belonging monotonicity) *Let* $[\alpha_1, \beta_1] \subseteq [\alpha_2, \beta_2]$. *If* $d_{[\alpha_1,\beta_1]}$ *belongs to* Σ *then* $d_{[\alpha_2,\beta_2]} \in \Sigma$.

Proof 2 *As* $d_{[\alpha_1,\beta_1]} \in \Sigma$, *from Definition 3 we have* $\forall [\alpha_3, \beta_3]$ *such that* $[\alpha_2, \beta_2] \subseteq [\alpha_3, \beta_3], d_{[\alpha_2,\beta_2]} \preceq d_{[\alpha_3,\beta_3]}$. *Furthermore, as* $\forall [\alpha_4, \beta_4]$ *such that* $[\alpha_3, \beta_3] \subseteq [\alpha_4, \beta_4]$, *we have* $[\alpha_1, \beta_1] \subseteq [\alpha_3, \beta_3]$ *with* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha_3,\beta_3]}$ *and* $d_{[\alpha_3,\beta_3]} \preceq d_{[\alpha_4,\beta_4]}$. *Therefore, as*

1. $\forall [\alpha_3, \beta_3]$ *such that* $[\alpha_2, \beta_2] \subseteq [\alpha_3, \beta_3]$, $d_{[\alpha_2,\beta_2]} \preceq d_{[\alpha_3,\beta_3]}$
2. $\forall [\alpha_4, \beta_4]$ *such that* $[\alpha_3, \beta_3] \subseteq [\alpha_4, \beta_4]$ $d_{[\alpha_3,\beta_3]} \preceq d_{[\alpha_4,\beta_4]}$

and we can conclude that $d_{[\alpha_2,\beta_2]} \in \Sigma$.

□

Property 3 (Dominance transitivity) *Let* $d_{[t_i,t_j]}$ *designate the temporal dependency* $A \xrightarrow{[t_i,t_j]} B$. *For all intervals* $[\alpha, \beta]$ *such that* $[\alpha_1, \beta_1] \subseteq [\alpha, \beta] \subseteq [\alpha_2, \beta_2]$, *if* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha,\beta]}$ *and* $d_{[\alpha,\beta]} \preceq d_{[\alpha_2,\beta_2]}$ *then* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha_2,\beta_2]}$.

Proof 3 *As* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha,\beta]}$, *from Definition 2 we have* $\frac{\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)} > \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})}$. *By multiplying this equation by* $\frac{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)}$, *we get*

$$\left(\frac{\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)} > \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})}\right) \frac{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)}$$
$$\equiv \frac{\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)} > \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})} \frac{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)}$$

Similarly, considering $d_{[\alpha,\beta]} \preceq d_{[\alpha_2,\beta_2]}$ *and multiplying it by* $\frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})}$, *we get*

$$\left(\frac{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)} > \frac{\mathbf{len}(B^{[\alpha,\beta]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})}\right) \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})}$$
$$\equiv \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha,\beta]})} \frac{\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)} > \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})}$$

Therefore, we have $\frac{\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B)}{\mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)} > \frac{\mathbf{len}(B^{[\alpha_1,\beta_1]})}{\mathbf{len}(B^{[\alpha_2,\beta_2]})}$ *and we can conclude that* $d_{[\alpha_1,\beta_1]} \preceq d_{[\alpha_2,\beta_2]}$.

□

3 Efficient Temporal Dependencies Discovery

Discovering temporal dependencies is time-consuming for large volumes of data. Positing that it is not useful to look for temporal dependencies with large time lag, we restrict the search of shifting intervals to the intervals $[\alpha, \beta]$ included in $[t_{min}, t_{max}]$, where t_{min} and t_{max} are parameters set by the end-user. They specify the time range in which the shifting intervals are taken. Indeed, a naive algorithm, that looks for dependencies between two events A and B, will explore all possible time shift intervals included in $[t_{min}, t_{max}]$ (the number of such intervals is in $\Theta((t_{max} - t_{min})^2)$). For each interval, it will compute its confidence value, which can be done in $\Theta(\#I)$, where $\#I$ is the number of intervals of A or B. Such an algorithm has to be executed with a relatively high frequency over data stream batches. Therefore, it is a key issue to improve its efficiency to make it suitable for the context of multiple data streams. To do that we propose TEDDY, TEmporal Dependency DiscoverY, an algorithm that (1) takes advantage of the monotonic characteristic of the confidence measure, as stated in property 1, to avoid considering time shift intervals that are guaranteed not to be valid; (2) exploits an upper bound on the confidence measure, whose complexity is $O(1)$, to reduce the computation required for the confidence evaluation; (3) explores the search space using a level-wise approach in order to discover significant temporal dependencies while computing the confidence value of each interval at the most once.

TEDDY is sketched in Algorithm 1. For every pair of events, it explores the temporal dependencies in a breadth-first approach. The inclusion operation over time shift intervals defines a semi-lattice, illustrated in Figure 6, where intervals at given depth d have the same length: $t_{max} - t_{min} - d$. This semi-lattice is traveled level by level. At each iteration of the *while* loop, Cand_d contains the d-depth shift interval candidates. Line 7, Prom_d is computed as the restriction of Cand_d to the dependencies whose confidence value is greater than a lower bound. If a temporal dependency from Prom_d dominates its two ancestors, then it is a promising dominant candidate and thus belongs to Σ_d (line 8). As such, it is added to the *Border* set whereas its ancestors are removed, as they are no longer the most specific intervals of *Border*. Line 9, $d+1$-depth candidates are generated if their d-depth ancestors belongs to Σ_d. Line 12 processes *Border* only to extract valid and significant temporal dependencies.

The four most important steps of this algorithm, the candidate generation, the pruning based on confidence and dominance, and the identification of valid and significant dependencies, are detailed in the following subsections.

3.1 Candidate time shifts generation

As stated in property 1, the confidence measure increases monotonically with time shift interval inclusion. In addition, property 2 stipulates that Σ-belonging is also a monotonic property. So, to prune the search space made of temporal dependencies that are not valid or not significant, the interval semilattice is traversed from the largest interval down to the singletons. If a time shift interval is

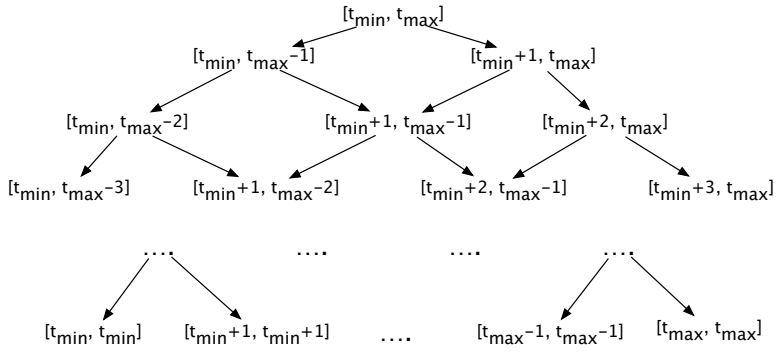


Figure 6: Interval shift join semi-lattice with respect to $\subseteq$.

not valid or does not dominate one of its direct ancestors, then none of the intervals included in it can correspond to a valid significant temporal dependency. As each interval at depth $d+1$ is included in at most two intervals at depth d, we generate an $d+1$-depth candidate by intersecting d-depth promising time shifts. Algorithm 2 presents the candidate generation procedure. The first test (line 2) checks that the bottom of the semilattice has not been reached. It processes L, the list of promising d-depth intervals ordered by their first endpoint. If the first (resp. last) interval, lines 3-5 (resp. lines 15-17), is $[t_{min}, t_{max} - d]$ (resp. $[t_{min} + d, t_{max}]$), then $[t_{min}, t_{max} - (d+1)]$ (resp. $[t_{min} + (d+1), t_{max}]$) is a $d+1$-depth candidate as its only one ancestor belongs to L. The loop (lines 7-14) generates all other temporal dependencies whose two direct ancestors are in L by intersecting their time shift intervals.

3.2 Pruning-based on confidence measure

In order to avoid the computation of the confidence values of unpromising dependencies, we consider the following property that bounds the difference of confidence between two time shift intervals:

Property 4 (Bounds on confidence) *Let A and B be two events, and $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$ be two shifted intervals. We have:*

$$|\mathbf{conf}(A \xrightarrow{[\alpha_1,\beta_1]} B) - \mathbf{conf}(A \xrightarrow{[\alpha_2,\beta_2]} B)| \leq \frac{(|\alpha_1 - \alpha_2| + |\beta_1 - \beta_2|) \times \#B}{\mathbf{len}(A)}$$

where $\#B$ represents the number of intervals in B.

Proof 4 *By shifting an interval $[t_j - \alpha_1, t_{j+1} - \beta_1] \in B^{[\alpha_1,\beta_1]}$ with $[\alpha_2 - \alpha_1, \beta_2 - \beta_1]$, the length of the resulting interval may gain or lose a maximum of $(|\alpha_1 - \alpha_2| + |\beta_1 - \beta_2|)$ time units. By multiplying this quantity by the number of intervals in B, the result follows.*

□

Algorithm 1 `TEDDY`

Require: IS a set of interval-based events, $[t_{begin}, t_{end})$, and $[t_{min}, t_{max}]$.
Ensure: All significant temporal dependencies over IS.

```
for all s_i ∈ IS do
  for all s_j ∈ IS do
    Border ← ∅
    Cand_0 ← [t_min, t_max]
    d ← 0
    while Cand_d ≠ ∅ do
      Prom_d ←Pruning_based_on_confidence(Cand_d)
      [Σ_d, Border] ←Pruning_based_on_dominance(Prom_d, Border)
      Cand_{d+1} ←Candidate_generation(Σ_d)
      d ← d + 1
    end while
    Significant_{s_i,s_j} ←Compute_valid_and_significant_TD(Border)
  end for
end for
return ⋃_{s_i,s_j} Significant_{s_i,s_j}
```

Furthermore, as stated by equation (2) page 8, valid temporal dependencies have a confidence value greater than

$$\text{MinConfidence}\left(L(\alpha,\beta)\right) \equiv \frac{\lambda L(\alpha,\beta) + \sqrt{\frac{3.84}{T}\lambda(T-\lambda)L(\alpha,\beta)(T-L(\alpha,\beta))}}{\lambda T}$$

where $L(\alpha,\beta) = \mathbf{len}(B^{[\alpha,\beta]})$ and $\lambda = \mathbf{len}(A)$. Property 5 provides a lower bound on $\text{MinConfidence}\left(L(\alpha,\beta)\right)$:

Property 5 (Lower bound on MinConfidence $(L(\alpha,\beta))$)

$$\textit{MinConfidence}\left(L(\alpha,\beta)\right) \geq \min\left(1, \textit{MinConfidence}\left(L(0,0)\right)\right)$$

Proof 5 *$L(\alpha,\beta)\left(T - L(\alpha,\beta)\right)$ is a quadratic function which vanishes at $L(\alpha,\beta) = 0$ and $L(\alpha,\beta) = T$. Therefore, MinConfidence $(L(\alpha,\beta))$ first increases and then decreases over $[0,T]$ with MinConfidence $(0) = 0$ and MinConfidence $(T) = 1$. Let $x_1 < T$ be such that MinConfidence $(x_1) = 1$. We can observe that MinConfidence (x) increases over $[0,x_1]$ (see Figure 7). As $L(\alpha,\beta) \geq L(\alpha,\alpha) = L(0,0)$, we can lower bound MinConfidence $(L(\alpha,\beta))$ by MinConfidence $(L(0,0))$, when it is below 1 – the maximum confidence value– or by 1 otherwise:*

$$\textit{MinConfidence}\left(L(\alpha,\beta)\right) \geq \min\left(1, \textit{MinConfidence}\left(L(0,0)\right)\right)$$

□

$\mathbf{conf}(A \xrightarrow{[\alpha,\beta]} B)$ is upper bounded by 1, therefore if MinConfidence > 1, there is no valid temporal dependency. Algorithm 3 details the evaluation of

Algorithm 2 `Candidate_generation`

Require: L, the list of promising d-depth intervals, ordered by their first endpoint.
Ensure: Cand, the list of $d+1$-depth candidate intervals.

```
1:  Cand← ∅
2:  if t_max − t_min > d then
3:    if (L[0] = [t_min, t_max − d]) then
4:      Cand← Cand ∪ [t_min, t_max − (d + 1)]
5:    end if
6:    i ← 0
7:    while i < #L − 1 do
8:      [α, β] ← L[i]
9:      [γ, δ] ← L[i + 1]
10:     if (α = γ − 1) and (β = δ − 1) then
11:       Cand← Cand ∪ [γ, β]
12:     end if
13:     i ← i + 1
14:   end while
15:   if (L[#L − 1] = [t_min + d, t_max]) then
16:     Cand← Cand ∪ [t_min + d + 1, t_max]
17:   end if
18: end if
19: return Cand
```

the confidence measure. The confidence value of the first candidate is computed (line 4). Then, the confidence value of the following candidates is estimated based on Property 4 (line 7). If the upper-bound (`lastConf + maxGain`) of the confidence value of a candidate is lower than MinConfidence $(L(0,0))$ (`boundMinConfidence`, estimated thanks to property 5), then the candidate cannot be valid. Otherwise, its exact confidence is evaluated (line 10) and, if it is greater than `boundMinConfidence` (line 11), the candidate is considered as a promising valid temporal dependency. Notice that the confidence measure is stored for future needs (line 12). This confidence value is used as a new reference for further `maxGain` evaluations, since `maxGain` tends to decrease when evaluated on distant intervals in Cand.

3.3 Pruning-based on dominance relationships

The `Pruning_based_on_dominance` function consists simply of evaluating whether each promising candidate satisfies equation (3) for its direct ancestors. In fact, property 3 states that if a temporal dependency dominates its direct ancestors, then it also dominates all its ancestors, and thus belongs to Σ. It is also added to the *Border* set whereas its ancestors are removed, as they are not the most specific temporal dependencies of Σ anymore.

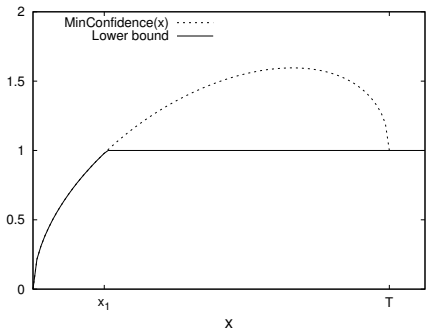


Figure 7: Illustration of MinConfidence function.

3.4 Identification of valid and significant dependencies

The last step of TEDDY is to consider the temporal dependencies of *Border* to ensure they are valid, that is, they truly satisfy equation (2). Algorithm 4 states that if a temporal dependency is valid (line 7) and more specific than any other dependencies of R (line 8), then it is added to R (line 9) and temporal dependencies that are more general are removed from R (line 10). If the dependency is not valid (line 12), its direct ancestors are recursively considered in lines 14 and 17.

If R is implemented as an interval tree, evaluating that a temporal dependency is the most specific among the n elements of R can be done in $O(\log(n))$. Finding all the dependencies of R that are more general than $d_{[\alpha,\beta]}$ can be done in $O(\min(n, k\log(n)))$ where k is the number of dependencies in the output list [14].

4 Performance Study

In this section, we report experimental results to illustrate the performance of our approach. We provide some experiments using a multi-camera test bed that makes it possible to specify the number of generated events and their frequencies. We also evaluate the efficiency of TEDDY algorithm on real-world data generated by smart environments and sensor networks. All experiments were performed on a 8 GB RAM computer with a octo-core processor clocked at 3 GHz, running Windows 7. The TEDDY algorithm is implemented in standard C++.

We analyze the experimental results with regard to the following questions:

- *What is the efficiency of TEDDY with regard to dataset characteristics that may affect its execution time?* At the beginning of section 3, we mention that a baseline algorithm, which explores all possible time intervals included in $[t_{min}, t_{max}]$, would have a time complexity function that

Algorithm 3 `Pruning_based_on_confidence`

Require: Cand, an ordered list of candidate intervals, $\#B$, the number of intervals in B and $\mathbf{len}(A)$.

Ensure: Prom, the set of promising valid temporal dependencies and their confidence value.

```
1: Prom← ∅
2: k ← 0
3: [α, β] ←Cand[k]
4: lastConf← conf(A --[α,β]--> B)
5: while k < #Cand do
6:    [α_k, β_k] ←Cand[k]
7:    maxGain← (|α − α_k| + |β − β_k|) × #B / len(A)
8:    if (lastConf + maxGain ≥ boundMinConfidence then
9:       [α, β] ←Cand[k]
10:      lastConf← conf(A --[α,β]--> B)
11:      if (lastConf ≥ boundMinConfidence then
12:         Cand[k].confidence← lastConf
13:         Prom← Prom ∪ Cand[k]
14:      end if
15:   end if
16:   k ← k + 1
17: end while
18: return Prom
```

is quadratic with regard to $t_{max} - t_{min}$ and linear with the number of intervals in the active interval sets. Therefore, it is interesting to know whether TEDDY outperforms the baseline algorithm and to compare their empirical complexities. Notice that both algorithms have the same complexity in the worst case, but that TEDDY uses pruning techniques that should increase its efficiency in practice. As the number of intervals in active interval sets increases with batch size, in the following, we study the behavior of TEDDY with regard to t_{max} and batch size.

- *How effective are TEDDY's pruning properties?* We carry out a detailed study of the impact of each pruning technique on TEDDY's performance.
- *Does TEDDY scale?* We want to investigate the scalability property of TEDDY's execution time.
- *Is TEDDY robust when data are noisy?* We analyze TEDDY's ability to discover temporal dependencies in noisy data.
- *How does TEDDY compare to state-of-the-art approach?* We compare the temporal dependencies found by TEDDYto the Dynamic Time Warping measure.

Algorithm 4 `Compute_valid_and_significant_TD`

Require: *Border*
Ensure: R the set of valid and significant TD.
1: $R \leftarrow \emptyset$
2: **for all** $[\alpha, \beta] \in Border$ **do**
3: `Confident_and_most_specific`$([\alpha, \beta], R)$
4: **end for**
5: **return** R

Function `Confident_and_most_specific`$([\alpha, \beta], R)$

6: **if** $depth([\alpha, \beta]) \geq 0$ **then**
7: **if** $([\alpha, \beta]$.confidence $\geq$ MinConfidence(α, β) **then**
8: **if** is_most_specific$([\alpha, \beta],\ R)$ **then**
9: insert$([\alpha, \beta], R)$
10: $R \leftarrow R \backslash$ set_of_most_general$([\alpha, \beta], R)$
11: **end if**
12: **else**
13: **if** $\alpha - 1 \geq t_{min}$ **then**
14: `Confident_and_most_specific`$([\alpha - 1, \beta], R)$
15: **end if**
16: **if** $\beta + 1 \leq t_{max}$ **then**
17: `Confident_and_most_specific`$([\alpha, \beta + 1], R)$
18: **end if**
19: **end if**
20: **end if**

4.1 Dataset description

Three types of datasets are used in the experiments: A multi-camera testbed that provides synthetic data used for efficiency and robustness to noise evaluation, and two real datasets that are also qualitatively studied in the case-studies (see Section 5).

4.1.1 Multi-camera testbed datasets

To generate the synthetic datasets, we built a simulator of a sensor surveillance network. It consists of the simulation of 8 video cameras that record what is going on in a rectangular space. Each camera captures the images of an elliptical area of this space as described in Figure 8. The simulation consists of moving objects along eight predefined rectilinear paths. To control the number of events occurring per unit of time, objects are generated according to a Poisson distribution. The area covered by each camera is divided into subareas that correspond to as many sequences. In total, there are 216 sequences that produce events in our experiments. A sequence contains an interval-based event *"object detected"* during every time interval an object is located in the associated subarea.

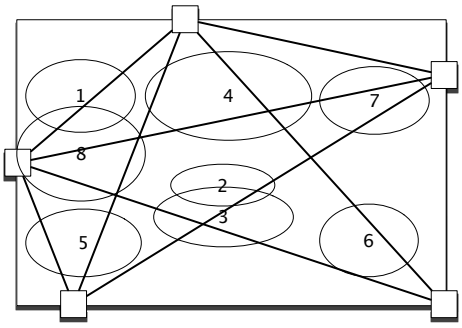


Figure 8: Synthetic testbed: A rectangular space is equipped with 8 video cameras. Objects are moving along 8 rectilinear trajectories.

We generate four different datasets, which differ according to the average number of events produced per minute and sequence. Quantitative characteristics of these datasets are given in Table 3.

Dataset	# Data sources	# Events	Duration	Avg events
SYNT02	216	85,806	3 hours	2
SYNT04	216	173,645	3 hours	4
SYNT08	216	352,553	3 hours	8
SYNT16	216	696,677	3 hours	18
Foxstream	1604	33,278,036	one week	2.1
Grizzly	8	136,603	3 months	8 every 5min

Table 3: Dataset characteristics. The last column shows the average number of events per minute and per sequence.

4.1.2 Real-world smart environment Foxstream

Foxstream[2] is a real-world dataset that depicts one week of activity of a smart environment composed by two video-cameras, two thermographic cameras and four motion sensors. These devices are used for outdoor surveillance of a building. The area captured by each camera is partitioned into 400 rectangular subareas that correspond to as many data streams. The detection of movement in each of these subareas is carried out as follows. For each subarea, we compute a background image by averaging the last 50 frames. Every second, the new image is compared with the background. A pixel is considered to have changed if its difference from the background corresponding pixel is greater than a given threshold (15 in our case). If the subarea has at least 75% of its pixels that have changed, an event *"motion detected"* is produced. We emphasize that motion

[2]Provided by the company `http://www.foxstream.com/`

detection in video is still a challenging problem. Therefore, this pre-processing may produce erroneous data: For instance, spurious motions can be detected due to changes in weather or lighting (rain, shadow). During the one-week period, the cameras generated about 33 million events, whereas the motion sensors produce around 38 000 events. Data streams are split into batches of one hour length ($T = 3600s$). Figure 9 (left) reports the distribution of these events through out a week of measurement. We can observe that the number of events is greater the weekdays, especially in the morning, around the lunch break and late afternoon.

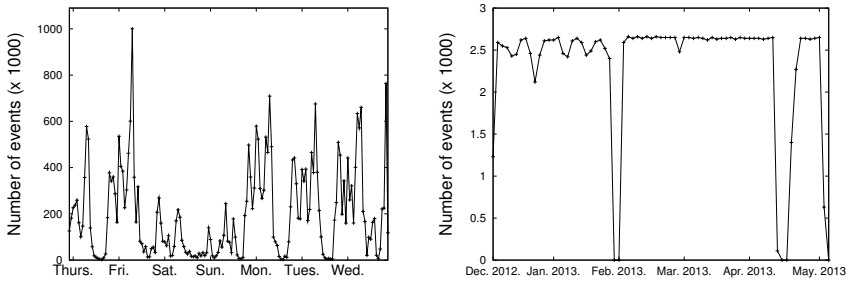


Figure 9: Number of events: Foxstream ($T = 1$ hour) and Grizzly ($T = 2$ days).

4.1.3 Real-world sensor network Grizzly

The GrizzLY (a.k.a. Grand LYon) project aims at implementing a strategy of weather forecasting at a small spatial scale for the Lyon urban area. It consists of the deployment of HiKoB[3] wireless sensors to monitor roads to improve snow removal and salting in the city. Eight sensors are spread over the urban area and located in neighborhoods known to be deicing sensitive. Each sensor provides real time information on in-pavement temperature combined with outdoor air temperature and relative humidity. The way we combine these measures into freezing and not freezing events is described in Section 5.2. Figure 9 (right) shows the number of events along time. In February and late April, the sensor network has encountered some partial failures and we decided to not consider these time periods in our experiments.

4.2 Experiment setup

We study the behavior of TEDDY with respect to various parameters: The frequency of events (for synthetic data), the period of observation T and t_{max}. In all the experiments, t_{min} is set to 0. Additionally, we examine the impact of the constraints that define valid and significant temporal dependencies (the χ^2 assessment of the confidence measure and the non-dominated constraint) on

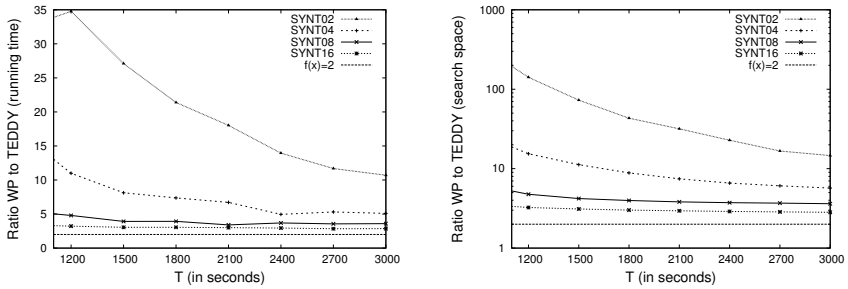
[3]http://www.hikob.com/

the search space size as well as on the execution time of TEDDY. To this end, the four following configurations of Algorithm 1 are studied:

1. **WP** (without pruning): Lines 7 and 8 are removed and all possible temporal dependencies are considered.
2. **Chi2** (χ^2-based pruning): Line 8 is removed and only the constraint on the confidence measure is used to reduce the search space.
3. **Gradient** (dominance-based pruning): Line 7 is removed and only the dominance constraint makes it possible to discard unpromising dependencies.
4. **TEDDY**: Both pruning constraints are fully exploited as presented in Algorithm 1.

4.3 Evaluation of the pruning efficiency

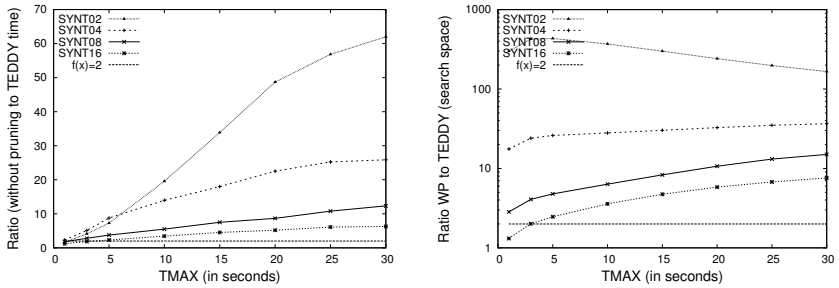
There is no other algorithm that computes temporal dependencies using the same constraints as in our proposal. Therefore, we first study the performance of TEDDY in comparison with the baseline algorithm. This algorithm considers all possible temporal dependencies and removes, in post-processing, the non valid or non significant dependencies. For these experiments, we do not take the execution time required by this post-processing step into account.



(a) (b)

Figure 10: Comparison of TEDDY and WP w.r.t. T ($t_{max} = 10s$): Execution time ratio (a); Search space size ratio (b).

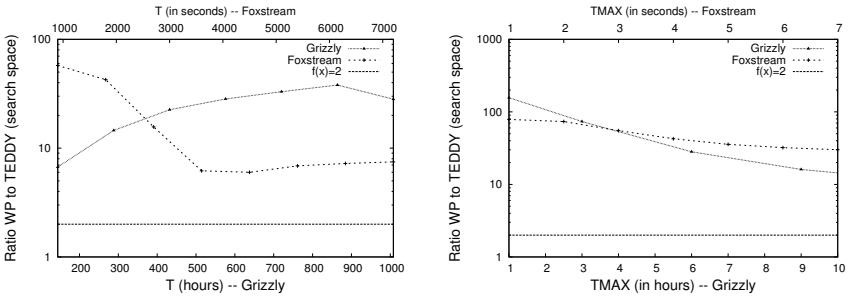
Figures 10 and 11 depict the behavior of TEDDY on synthetic data when T and t_{max} vary. In each figure, the running time and search space size ratios of WP to TEDDY are given. Each value is averaged over all the sequences of the same size. In most cases, TEDDY is at least twice as fast as WP (see line $f(x) = 2$). The ratio of the execution time increases with t_{max} since the number of possible intervals is quadratic in $t_{max} - t_{min}$ and TEDDY is able to prune a large part of them early on. On the contrary, when T increases, the ratio tends to decrease since the number of intervals $\#A$ of each event A tends to



(a) (b)

Figure 11: Comparison of TEDDY and WP w.r.t. t_{max} ($T = 900s$): Execution time ratio (a); Search space size ratio (b).

increase and TEDDY is not able to prune the search space as much. Indeed, MAXGAIN increases linearly with $\#A$ and the condition at line 8 of Algorithm 3 tends to be always true which implies that the time interval $[t_{min}, t_{max}]$ cannot be pruned. Furthermore, from Figures 10 and 11, we can also notice that the denser the datasets, the lower the ratios are.



(a) (b)

Figure 12: Search space size ratio of WP and TEDDY w.r.t. T (a) and t_{max} (b) (default values for Grizzly: $T = 7$ days and $t_{max} = 1$ hours; For Foxstream: $T = 1800s$ and $t_{max} = 5s$). Foxstream axis on top; Grizzly axis on bottom.

Figure 12 evaluates the search space size ratio of WP to TEDDY on real datasets. We notice that TEDDY's behavior on Foxstream is similar to that observed on SYNT16, the two datatsets having a large number of events. On the Grizzly dataset, the behavior is different: When T increases, the ratio tends to increase too because the MAXGAIN property is more effective. This is due to the fact that weather phenomena are lasting much longer than the events observed with video cameras. Furthermore, the variations are smoother.
Figures 13 and 14 show the proportion of the search space explored by TEDDY. Among the pruned candidates, we make a distinction between those removed thanks to the chi2-based constraint and those discarded by the gradient-based

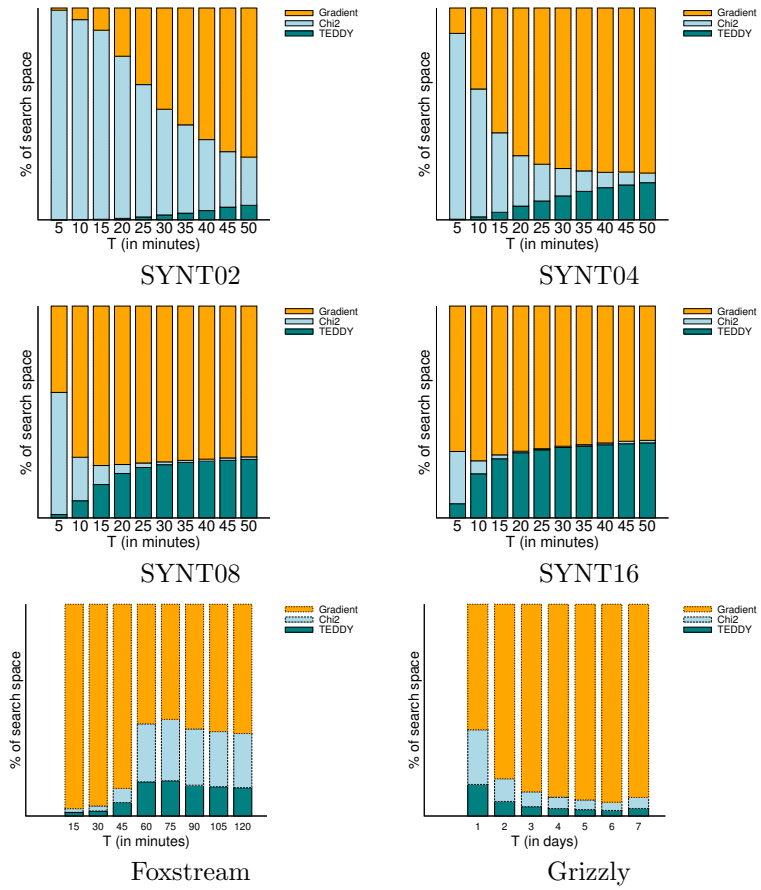


Figure 13: Impact of each constraint on search space size with regard to T: Number of candidates pruned by Chi2 and Gradient, and number of candidates considered by TEDDY.

constraint. These quantities are evaluated with respect to T (Figure 13) and t_{max} (Figure 14). A first observation is that the number of candidates avoided thanks to the two constraints is much higher than the number of dependencies considered by TEDDY, except when the dataset is very dense and t_{max} very small. The gradient constraint is even more effective when the dataset density increases or the values of T and t_{max} grow. Indeed, while T increases, the number of candidates avoided due to the gradient-based constraint increases or remains stable whatever the dataset density. This pruning criterion becomes even more effective when t_{max} increases. The larger the length of a pruned interval, the greater the size of the pruned search space. Indeed, if an interval $[\alpha, \beta]$ does not dominate one of its direct ancestors, it is pruned by the gradient-based constraint as are $\frac{(\beta-\alpha)\times(\beta-\alpha+1)}{2} - 1$ other candidates, that is to say all

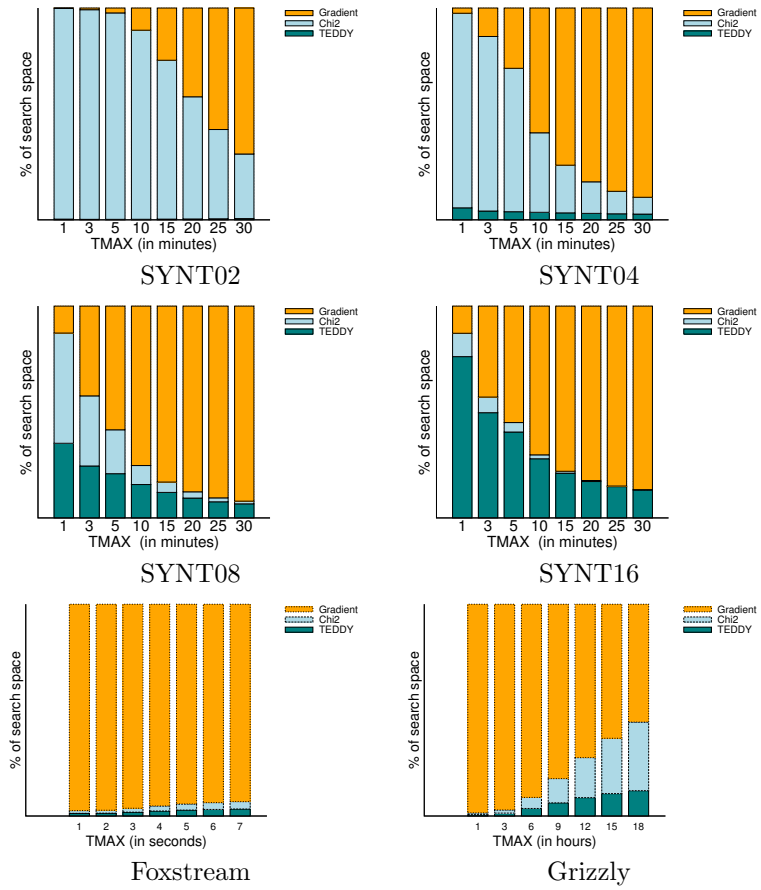


Figure 14: Impact of each constraint on search space size with regard to t_{max}: Number of candidates pruned by Chi2 and Gradient, and number of candidates considered by TEDDY.

the dependencies that are below $[\alpha, \beta]$ in the semi-lattice. Beside, chi2-based pruning tends to be less efficient when t_{max} and/or T increase. As explained above, this is due to maxGain that increases with the time interval length and the number of intervals associated to an event. On the real datasets, TEDDY is as efficient as on the synthetic ones. The search space size explored by TEDDY is very small, and this pruning can be due to one or the other constraint.

4.4 Execution time and number of extracted dependencies when parameters vary

Figure 15 reports the execution time of TEDDY and the average number of dependencies discovered per sequence pair when T varies. Figure 16 also studies

the performance of TEDDY but according to the value of t_{max}. In a manner consistent with what has been observed from Figures 13 and 14, TEDDY benefits from the two pruning techniques in obtaining very good time performance. Notice that, for each dataset the execution time is always much lower than T length (at least 200 times). Therefore the temporal dependencies computation is faster than the data acquisition process, which is a prerequisite for data stream mining techniques. For dense datasets (SYNT08 and SYNT16), the number of extracted dependencies increases with T and t_{max}. For SYNT02 and SYNT04, the number of dependencies decreases with T: Some values may become statistically insignificant on a large sequence of data, when the density of events is low. The number of dependencies also decreases slightly with t_{max}. This is due to the gradient-based constraint: A dependency satisfies this constraint if it dominates all its ancestors. Thus, as the number of ancestors increases with t_{max}, the constraint may become unsatisfied, especially when the dataset density is small.

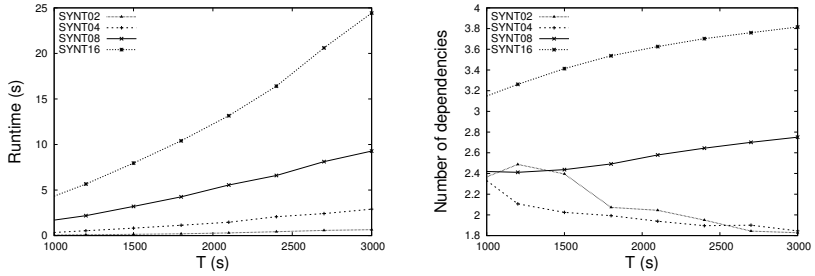


Figure 15: Total running time (left) and average number of discovered dependencies (right) per sequence pair with respect to T ($t_{max} = 10$).

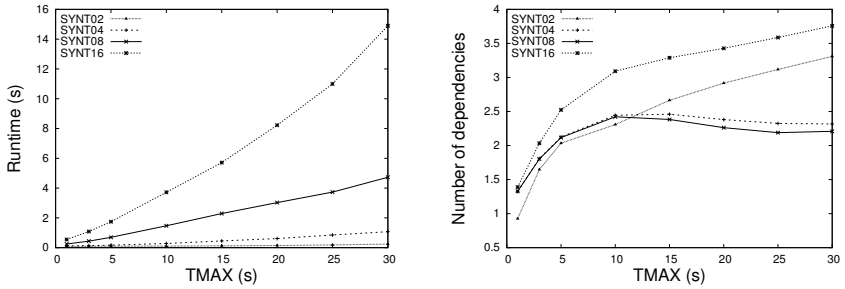


Figure 16: Running time (left) and number of discovered dependencies (right) with respect to t_{max} ($T = 900s$).

Figures 17 and 18 reports the running time of TEDDY and the average number of dependencies per sequence pair with respect to T (a) and t_{max} (b) for the Foxstream and Grizzly datasets, respectively. These results confirm the

observations made above: TEDDY's execution time increases according to t_{max} and T but remains negligible compared to the real batch duration. Therefore, temporal dependency detection can be carried out without risk of batch overflow. Furthermore, it is so fast that there is still time for more sophisticated analysis in addition to temporal dependency discovery. The same observations can be

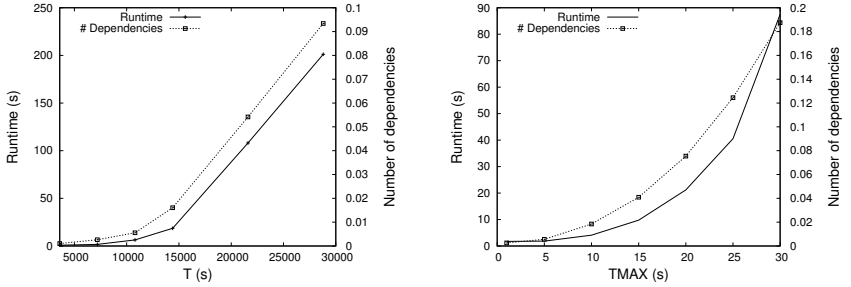


Figure 17: Total running time and average number of discovered dependencies per sequence pair on Foxstream with respect to T (in seconds) (a) and t_{max} (in seconds) (b) (default values $T = 7200$ s and $t_{max} = 5$ s).

made on the Grizzly data (see Figure 18).

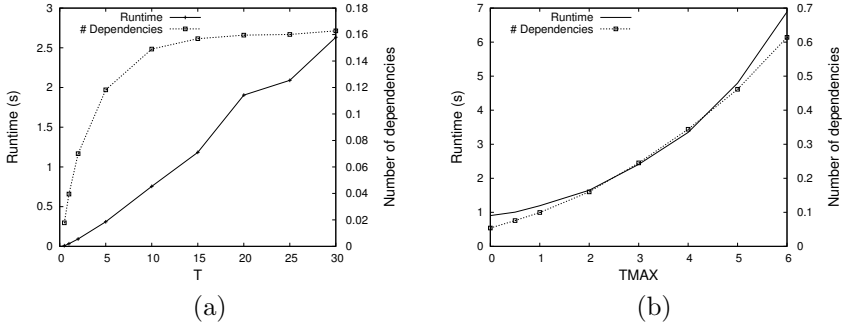


Figure 18: Total running time and average number of discovered dependencies per sequence pair on Grizzly with respect to T (in days) (a) and t_{max} (in hours) (b) (default values $T = 21$ days and $t_{max} = 2$ hours).

These first series of experiments are very conclusive and we can argue that our approach scales up well with regard to the t_{max} and T parameters.

4.5 Robustness to noise

This empirical study also aims to investigate TEDDY's robustness against noise. We assume that synthetic datasets described in Table 3 are noiseless, having been generated following a specific scenario in a testbed. We introduce

a uniform random noise in each dataset by adding $x\%$ of spurious events, $x \in \{1, 3, 5, 10, 20, 30\}$. We suppose that the set of valid and significant dependencies of the original dataset are those expected and constitute the relevant dependencies. TEDDY's robustness is thus evaluated by computing the recall and the precision of this set of dependencies. To assess whether two temporal dependencies $A \xrightarrow{[\alpha^1,\beta^1]} B$ and $C \xrightarrow{[\alpha^2,\beta^2]} D$ depict the same phenomenon, we consider the two following cases:

- exact matching: All the temporal dependency parameters are equal ($A = C$, $B = D$, $\alpha^1 = \alpha^2$, $\beta^1 = \beta^2$);
- relaxed matching: The two temporal dependencies are between the same events ($A = C$, $B = D$).

Finally, we report the F_1 score, a trade-off between precision and recall which reaches its best value at 1 and worst score at 0:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

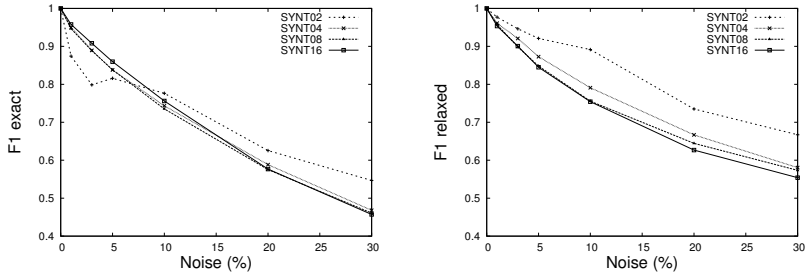


Figure 19: Robustness of TEDDY on synthetic dataset with respect to percentage of noise ($T = 900$, $t_{max} = 5$).

Figure 19 shows the F1 score computed for exact matching (left) and relaxed matching (right) for the different datasets. It demonstrates that TEDDY is rather robust to noise as the harmonic mean of precision and recall remains satisfactory even when the percentage of noise is high. Two things are worth noticing. Firstly, exact matching gives worse results. In fact, the shifted intervals are less well identified by TEDDY when the percentage of noise increases. Secondly, the higher the number of events per minute, the less robust against noise TEDDY becomes. If the proportion of events is small, then the added spurious events are too sparse to produce additional temporal dependencies and the F1 score remains high.

4.6 Comparison to DTW

It is unrealistic to compare our approach with other classical pattern mining based approaches that aim to discover regularities among a collection of se-

quences. Indeed, these approaches rely on the assumption that the sequences of events are described with the same alphabet which is not true in our case. We discuss these approaches in more detail in the related work section. TEDDY is more related to methods that make pairwise comparisons, especially the methods that aim to compute a distance between two time-series. Actually, temporal dependencies can be seen as a kind of distance/similarity measure between two sequences of interval-based events. Therefore, in this section we compare the temporal dependencies discovered by TEDDY with the Dynamic Time Warping (DTW) method that is commonly known as the most effective time series distance measure [49, 29].

To explore the differences between TEDDY and DTW, we apply the DTW algorithm on the two real-world datasets. Each stream is transformed into a numerical time series, where every state is associated to a natural number. We compute the DTW similarities between every pair of streams, and we select the K most similar pairs. Using TEDDY, we also select the K most confident temporal dependencies. TEDDY is used with $T_{min} = 0$, $T_{max} = 10$, and the DTW algorithm uses a global constraint (Sakoe-Chiba band = 10) in order to have a comparable settings.

Dataset	K	Jaccard index
Foxstream (1 hour)	10	0.18
Foxstream (1 hour)	25	0.064
Foxstream (1 hour)	50	0.031
Grizzly (1 week)	10	0.11
Grizzly (1 week)	25	0.32

Table 4: Jaccard index on the K most similar pairs identified by TEDDY and DTW.

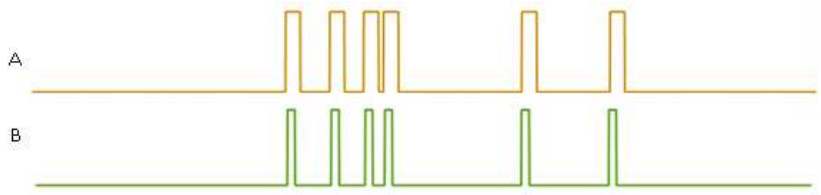
Table 4 reports the Jaccard index between the K pairs of streams using DTW and the K most confident temporal dependencies discovered by TEDDY for different value of K. On Foxstream, the Jaccard indexes are very low (between 3×10^{-2} and 2×10^{-1}), but are slightly higher on Grizzly(between 1×10^{-1} and 3×10^{-1}). These results show that these two approaches do not aim to capture the same patterns.

Dataset	Kendall's τ
Foxstream (1 hour)	0.02
Grizzly (1 week)	0.22

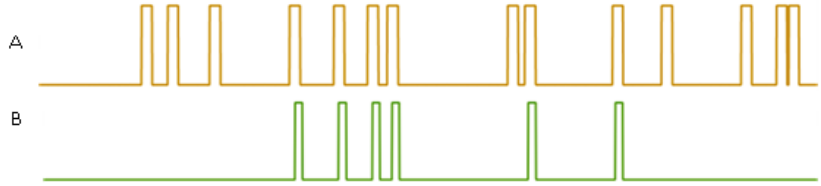
Table 5: Kendall's τ measure on the rankings given by TEDDY and DTW.

Let us now consider the rankings of the pairs of streams provided by the confidence measure of TEDDY and the similarity measure of DTW. The Kendall rank correlation coefficient (Kendall's τ) between these two rankings is given in Table 5. The low Kendall's τ measures indicates for a lack of correlation between

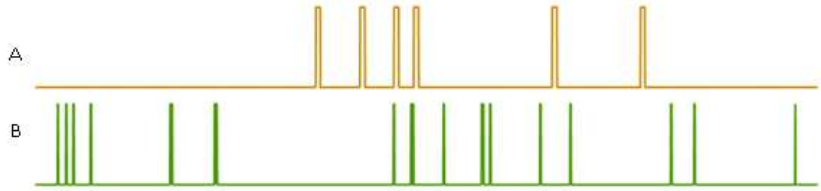
TEDDY and DTW. Figure 20 gives examples of temporal relationships that are detected by DTW but not by TEDDY, and vice versa.



(1) Relation found by both methods.



(2) Relation found by TEDDY, but not found by DTW.



(3) Relation found by DTW, but not found by TEDDY.

Figure 20: Examples of relations detected by TEDDY and/or DTW.

We can conclude that TEDDY is a novel approach to automatically detect dependencies that are not identified by other existing methods, such as DTW.

5 Case-studies

In this section, we present two case studies. The first one is based on the analysis of the streams generated by video and thermographic cameras as well as motion sensors to monitor the movements around an office building. In the second case study, we seek to identify dependency relationships between frozen geographic areas from data produced by a sensor network deployed in the city of Lyon.

5.1 Outdoor monitoring using heterogeneous sensors

The first case-study is based on the data generated by the real-world smart environment Foxstream presented in Section 4. The data comprise one week of monitoring of an office building with a smart environment composed by two video-cameras, two thermographic cameras and four motion sensors. These data are pre-treated in a way that generates 1604 streams as explained above.

5.1.1 TEDDY's ability to detect a spatial phenomenon

Dependency graphs extracted from the Foxstream data contain up to ten thousand temporal dependencies. This high number is mainly due to the large number of data streams mined. In fact, a moving object generates events in many different areas, leading to as many reliable temporal dependencies. Figure 21

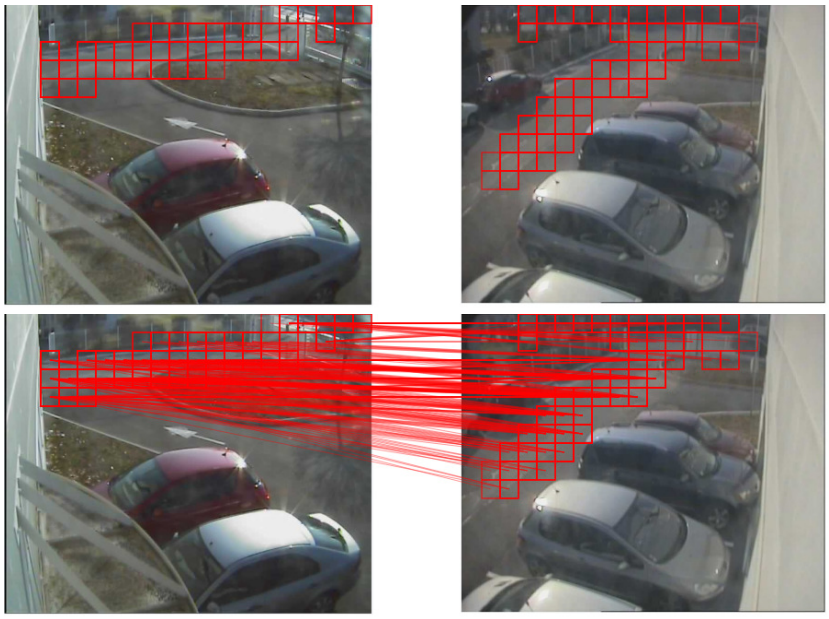


Figure 21: Temporal dependencies between data sources corresponding to subareas of two cameras. Images on top represent the subareas involved in dependencies; Bottom images display the temporal dependencies between them ($T = 3600$ and $t_{max} = 10$).

shows the extracted dependencies between subareas of two overlapping cameras. These dependencies have been extracted on Tuesday between 10am and 11am. The corresponding dependency graph is represented in Figure 22. This graph contains two connected components. If we study the evolution of this graph through time, it appears that the changes are small during the office hours of working days. However, at night, the graph is reduced to the connected component of the top. Therefore, our approach makes it possible to automatically discover two distinct type of behaviour: One component captures the activity around the building (cars and persons) and the other depicts the activity (car traffic, pedestrians) within the street located in the neighbourhood of the building, but not in the private parking lot.

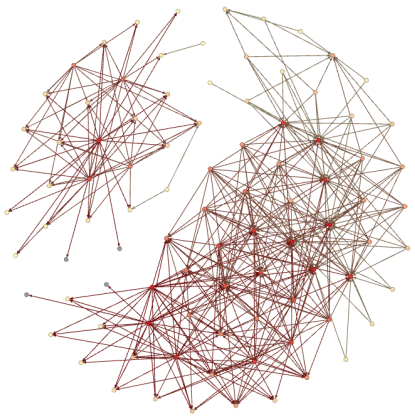


Figure 22: Dependency graph corresponding to data sources of Figure 21. The two connected components corresponds to two different type of behavior: Motion around the building (bottom/right component) and the background street activity (up/left component).

5.1.2 TEDDY's ability to identify anomalies in a temporal phenomenon

To investigate whether the temporal dependencies are well suited to describe a temporal phenomenon, we try to find anomalies in the weekly activity observed around the monitored building. More precisely, we compare dependency graphs obtained from one-hour batch lengths taken every 24 hours. The idea is to find significant changes while the recorded activity is similar. Such changes can correspond to a sensor dysfunction or to a new pattern in the observed activity.

The video surveillance cameras used in this dataset are equipped with infrared LEDs, used to illuminate the scene in the infrared spectrum. This technology produces heat which attracts all sorts of insects, especially spiders. Spider's webs partially obstruct the view of the camera and blur the captured images. This phenomenon is even more important when it rains, water droplets hanging from the web causing serious obstruction of the camera view. Our goal is to automatically detect cameras that are subject to interference by spiders, using the evolution of the dependency graph in time. To this end, we apply a spider's web mask on the camera images captured during Wednesday and study the impact of the spider's web on the dependency graphs as follows. For each one-hour batch of Wednesday, we build the dependency graph from TEDDY's output and select the dependencies that involve a subarea of the camera. This so-called camera egocentric dependency graph is compared to the similar graph built from the batch of events produced 24 hours before, that is on Tuesday. The Jaccard index is used to evaluate the similarity of the set of arcs of these

Figure 23: Temporal dependencies between two cameras for a batch corresponding to a period of no activity around the building (night-time). The corresponding dependency graph is the up/left component of the graph from Figure 22.

two graphs. Furthermore, it enables us to overcome the fluctuations related to night/day phenomena. We assume that if the similarity measure varies by more than two times the standard deviation σ of the preceding similarities, the corresponding camera is suspected of undergoing an unusual phenomenon, which may be due to the presence of spider's web on the camera lens.

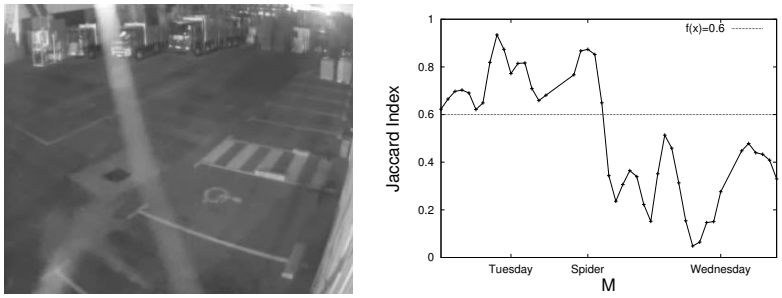


Figure 24: An image with a spider's web (left). Jaccard index evolution over Tuesday and Wednesday (right) ($T = 3600$ and $t_{max} = 5$).

Figure 24 (right) reports the Jaccard index values between the camera egocentric dependency graphs obtained at time M and $M - 24$. The spider mask is introduced at $M = 21$. We can observe a drop in the Jaccard index that coincides with the appearance of the spider's web. Afterwards, the value of the Jaccard index varies by more than $2 * \sigma$. This experiment demonstrates that it is possible to capture unusual phenomena / anomalies with TEDDY's output. It is a proof of concept and more sophisticated treatments can be defined.

5.2 Road deicing operation assistance

Every winter, deicing or snow-clearing of roads is an important issue that impacts on the local economy, public finances and the environment. On the one

hand, city authorities are encouraged to deploy significant means for road deicing as the road network is a key infrastructure for a highly connected and just-in-time economy. On the other hand, deicing expenses constitute an important part of the public works budget that covers the deployment of substantial means. Moreover, road salting may have an important negative environmental impact [8, 35]: Deicing salts leach into the soils where the ions may accumulate and eventually become toxic to the organisms and plants growing on them. The contamination of the ground water may also result in health impact on the urban population. In addition, deicing operations plays havoc with technical equipment: Salt tends to cause corrosion, rusting the steel used in most vehicles as well as the threaded rods used in bridge concrete.

Deicing of roads must therefore be well-organized in order to limit its negative environmental, technical and health impacts. To this end, road operators rely on weather forecasting. However, weather alerts are on the scale of an entire urban area whereas topographic (e.g., hill) and urban (e.g. parks, buildings) disparities can cause differences in temperature and freezing phenomena. As a result, many roads are processed without this being necessary and some slick roads are not deiced because no freezing alert was triggered at the urban scale. To enhance deicing operation management, and process only slick roads, the weather forecasting must be done at a smaller scale. It consists of considering short-lived atmospheric phenomena that are smaller than mesoscale, about 1 km or less [6].

5.2.1 Freezing Alert Model Construction

To construct the freezing alert model, we use the data provided by HiKoB sensor sites between November 2012 and April 2014 (Summer 2013 is excluded). A first analysis of these data show a strong spatial variability among the temperature measured over the sites. The difference between the air temperature of the sites and the one of the weather station of the city varies from 0.18°C to 1.5°C in average.

We used these data to built a weather dependency model over the sites based on the temporal relationships between the weather events. The weather conditions at a site are defined by the five following attributes:

1. The site location name (ID),
2. The period of the day (POD) – One of the four following periods: 12 a.m.-6 a.m., 6 a.m.-12 p.m., 12 p.m.-6 p.m. or 6 p.m.-12 a.m,
3. The road temperature (RT): The greatest integer less than or equal to the road temperature,
4. The sign of the road temperature gradient (∇RT): +1 if RT is increasing, -1 if RT is decreasing, 0 otherwise. The road temperature gradient at time t (in seconds), $\nabla RT(t)$, is evaluated by $\nabla RT(t) = \frac{\overline{RT_{(t-600)}} - \overline{RT_{(t-0)}}}{600}$ with $\overline{RT_{t-x}} = AVG(\{RT(t) \mid 600 + x \leq t < x\})$. It evaluates the evolution of the temperature over last two 10 minutes periods,

5. Freezing condition (FZ): 1 if the weather conditions are right for freezing to happen, that is air temperature is less than 0 and RT is less than or equal to the frost point value; 0 otherwise.

These weather events mainly rely on the road temperature. But as this value is strongly influenced by day / night phenomena, we also take into account this information as well as its trend. As we want to predict glaze ice on the road, the freezing condition is also considered. Notice that during the considered period, the freezing condition occurred during 4 479 hours out of 55 499 hours of measurement.

Furthermore, to take into account the road operators logistical and technical constraints, and make the model effective in the deicing decision process, the freezing alert must be triggered at least two hours before a freezing event. Therefore, we set parameters α, β to be as follows: $2\ hours \leq \alpha \leq \beta \leq 4\ hours$. Notice that the construction of the model takes 45 seconds. We obtain $22,972$ valid and significant temporal dependencies out of which $2,315$ correspond to a weather event with freezing condition. As an example, we obtain the following rule [45]: *When the site Lacassagne has a road temperature equals to* $1°C$*, with a decreasing gradient and no freezing conditions during the night, then SainteFoy has a road temperature equals to* $0°C$*, with a decreasing gradient and freezing conditions during the morning and this appears between 2 and 4 hours later.*

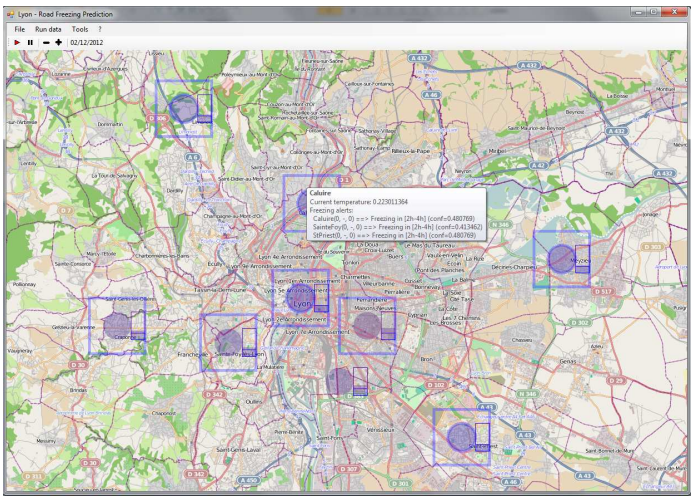


Figure 25: Road freezing prediction in Grand Lyon.

5.2.2 Freezing dependencies identified by TEDDY

Once the temporal dependency-based model has been constructed, we used it to trigger freezing alerts on new data. A graphical user interface makes it possible to visualize the current sensor values as well as the triggered alerts

(see Figure 25)[4]. The sensor locations are displayed on the Grand Lyon city map[5]. The current temperature is represented by a bargraph. When a freezing alert is triggered, the site is surrounded by a square box. A bold blue circle signals that a road freezing episode is currently happening. By clicking on the sensor location on the map, the end-user can obtain the details of the temporal dependencies that are triggered during the current freezing alert.

Our aim is to use the temporal dependency-based model to predict freezing weather conditions. To evaluate the accuracy of this approach, we use a cross-validation procedure to assess how this analysis will generalize to an independent data set. To this end, the data are divided into ten time periods so that each period contains the same number of freezing events. Then, one round of cross-validation consists in building the temporal dependency-based model using 9 time periods and testing the model on the last time period.

Our system is set up to predict freezing event occurrences at least two hours ahead. The average time interval size of computed temporal dependencies is of 17 minutes. A freezing alert is considered valid if a freezing event appears during the prediction interval or within the four following hours. Indeed, the salt spread on the road remains active at least four hours. Figure 26 reports the location of each sensor site on Lyon's map. The precision and recall for each site are displayed in Figure 27 and Table 6 reports the influence of each site to trigger freezing events of other sites.

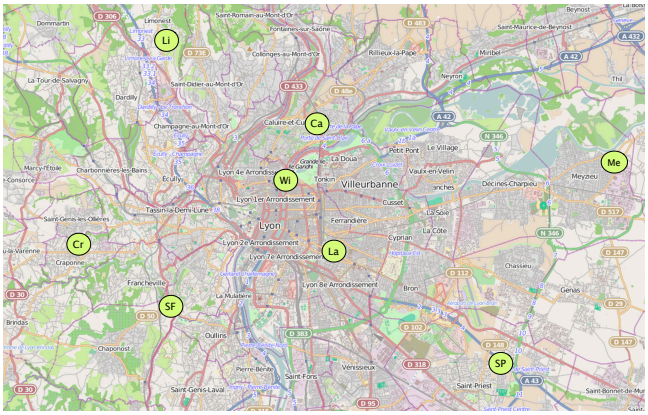


Figure 26: The sensor locations over the Lyon urban area.

Most of the sites have high precision and recall values. Indeed 70.14% of the freezing events are correctly predicted. The sites that have the smallest values are located on the periphery of the urban area. The Caluire (Ca) and Craponne (Cr) sites have the worst precision or recall values. Caluire has the

[4]A video of TEDDY system is available here: `http://liris.cnrs.fr/~mplantev/doku/doku.php?id=teddy`

[5]We use a map provided by `www.openstreetmap.org`.

best precision value but the worst recall one. This is due to the prevalent cold winter wind that is north-easterly in this area. As it is the most north-eastern site, it encounters freezing conditions before the other sites. The Craponne site is rather different from the other ones: It is one of the highest sites and the most western one. Furthermore, this is a residential area with detached houses and little road traffic. Freezing alerts detected on Craponne site are mainly triggered thanks to the Meyzieu site that is also a residential area but located in the East and down in the plain. It is noteworthy that both sites are not able to predict freezing conditions by their own: No freezing alert was triggered thanks to these two sites. Both sites also have minor influence (3% in average) on the other sites (see Table 6).

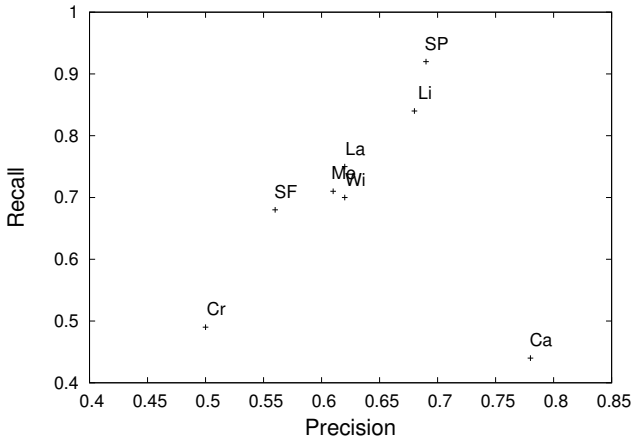


Figure 27: Confidence and recall of each sensor location.

Besides providing good performance for triggering freezing episodes, our system produces new actionable insights that are very hard to obtain using simulation models based on whole urban area. Indeed, considering at the same time locale weather measurements and short time intervals (prediction every ten minutes) makes it possible to highlight *trajectories* of freezing alerts. Such insight is very useful to well-organize and optimize the deicing operations.

Lastly, this temporal dependency model can be used to locate new deployment sites. For instance, to enhance the freezing alert triggered on Caluire location, new sites should be deployed in the North East area.

6 Related work

This paper makes a significant contribution to data stream management. The novelty of our approach is represented by the knowledge nuggets discovered over multiple heterogeneous data streams. Due to the type of processed data and

Station location	Ca	Cr	La	Li	Me	SF	SP	W	Avg. influence
Ca	0	0	0.06	0	0.02	0.08	0.04	0.06	0.03
Cr	0	0	0	0	0.14	0	0	0.11	0.03
La	0	0	0.21	0.18	0.17	0.13	0.10	0.14	0.12
Li	0.23	0.11	0.09	0.24	0	0.10	0.17	0.12	0.13
Me	0	0.41	0.27	0	0.32	0.22	0.16	0.16	0.19
SF	0	0	0.17	0	0.16	0.25	0.08	0.21	0.11
SP	0.70	0.21	0.13	0.51	0.13	0.15	0.40	0.13	0.30
Wi	0.07	0.28	0.07	0.07	0.06	0.07	0.05	0.06	0.09

Table 6: Influence of each sensor location to trigger freezing alerts (e.g., SP site is responsible of 70% of freezing alerts on Ca site).

the confidence measure used, our proposal could be considered to be related to the contributions from time series area. In this area of research, algorithms are devised for measuring the similarity between two time series [49]. Most of them extend the Dynamic Time Warping (DTW) algorithm [46, 28, 49, 29] that makes it possible to find an optimal time alignment between two time series. However, even if DTW is commonly known as the most effective time series distance measure, this measure is not suited for capturing temporal dependencies and for characterizing the time-delay between event occurrences. An empirical comparison is reported in Section 4.6. Indeed, time-series are warped non-linearly in the time dimension so as to determine a measure of their similarity independent of certain non-linear time variations. In our work, we aim to consider linear transformation of the interval-based events in order to find out temporal dependencies and their most specific time-delay intervals.

The contribution presented in this paper can also be compared with the work on attribute dependency analysis. Using the rough set theory [40] as well as the equivalence class structure [52], the dependency of attributes makes it possible to discover which variables are strongly related. A dependency can be interpreted as the proportion of objects for which it suffices to know the values of some attributes to determine the values of some other attributes. The attribute dependency problem has also been investigated in the data base community with (conditional) functional dependencies [18]. The temporal dependencies we introduce in this paper are different on two points. First, they involve two streams (objects) while the classical attribute dependencies are "intra object" (i.e., involve a single object). Second, we take into account the time relation which is neglected in other forms of dependencies.

More generally, our work is related to various research areas: (i) temporal pattern mining on event intervals, (ii) trajectory and workflow mining, and (iii) mining smart environments.

Taking temporal information into account, most of the existing approaches aim at discovering frequent patterns among a set of sequences (i.e., temporal information is used to order the events within the sequences). Such approaches include mining of sequential patterns [3] or episodes [33] on sequences of "point

events" (i.e., events with no duration), with applications in data stream processing [12, 34, 11]. In addition, some approaches show a particular interest in the time transition between events, either pushing some specific time constraints inside the search, like the well-known mingap, maxgap and window-size, or trying to characterize the lag intervals between two event types [22, 48] or between items within a sequence [20]. Furthermore, based on the fact that sequential pattern mining on point event sequences is inadequate in discovering more sophisticated relations than the "before" / "after" relation, some works consider "interval events", i.e., events that have a duration. This introduces more complex relations between events extending Allen's algebra [13, 36, 47, 24, 9, 39, 51, 50]. Some approaches define events based on the interval model, but only the "before" / "after" temporal relation is supported [15, 4, 16]. We emphasize the fact that these approaches, except [32, 30, 31], are not dedicated to data stream mining tasks. Moreover, they aim at discovering regularities in a collection of sequences, whereas we wish to highlight some temporized dependencies between streams based on their interval-based events. Furthermore, incorporating statistical metric like the χ^2 test within the pattern mining process is a well-studied issue [37, 25]. But these measures are often considered in addition to others such as confidence and support measures. In this paper, this statistical assessment is also used to automatically set some thresholds. Indeed, threshold setting is a difficult issue for end-users who are often not data miners and not familiar with these techniques. Thus, the χ^2 test used in our proposal enables us to obtain more valuable results while limiting the number of parameters the end-user has to set. Finally, we are convinced that the two tasks are different and complementary. Indeed, two data sources may support several frequent sequences without having a dependency relation between them and vice versa.

Temporally annotated sequences [20] have been successfully applied to workflow mining [7] and trajectory pattern mining [21]. Once again, our approach differs in the nuggets that are discovered. For process or trajectory mining, a collection of logs is examined to highlight the regularities. In our approach, we search for dependencies between data sources.

From an application point of view, our work is close to the research conducted in the smart environment community, where one of the main challenges is activity discovery. In [43], Rashidi and Cook proposed mining sequential patterns over time from streaming non-transactional data using tilted-time windows [23]. They extended their previous work [44], thereby introducing the first stream mining method for discovering human activity patterns in sensor data. Based on these proposals, association rule mining was applied to discover temporal relations of daily activities in [38]. Nevertheless, the motivations are different and these approaches are supervised while ours is unsupervised.

7 Conclusion

Designing new methods to discover relations over multiple heterogeneous data streams is a challenge. To the best of our knowledge, recently proposed methods

focus on the discovery of regularities among events within streams. No proposal that we are aware of takes on the challenge of discovering particular relations between data sources that produce multiple heterogeneous data streams. Our work investigates a new direction in data stream mining. It identifies temporal dependencies between data streams. First, we define the novel problem of mining temporal dependencies over multiple heterogeneous data streams. Then, we design and implement a complete, though scalable, algorithm that efficiently computes such temporal dependencies. Our approach is robust to the temporal variability of events and characterizes the time intervals during which the events are dependent. It links two events if the occurrence of one is often followed by the appearance of the other in a certain time interval. The proposed interval-based approach determines the most appropriate time intervals of a temporal dependency whose validity is assessed by a χ^2 test. As several intervals may redundantly describe the same dependency, the approach retrieves only the few most specific intervals with respect to a dominance relationship over temporal dependencies, and thus avoids the classical problem of pattern flooding in data mining. The TEDDY algorithm takes advantage of various properties to prune the search space while guaranteeing the discovery of all valid and significant temporal dependencies. We conducted an extensive experimental study of both synthetic and real-world data streams from smart environments equipped with various kinds of sensors (cameras, motion sensors, etc.). From these experiments, we conclude that the pruning techniques are very efficient and speed up TEDDY running time by a factor that varies between 2 and 60. A qualitative analysis of the output shows that TEDDY produces a small set of non-redundant dependencies that accurately describe the phenomenon captured by the data.

There are several ways of extending the main results of this paper. First of all, we plan to investigate the dynamics of dependency graphs through time. The set of temporal dependencies can be viewed as an attributed graph whose nodes describe events. Mining such dynamic attributed graphs would enable us to discover periodic phenomenon and other evolving behaviors that cannot be easily discovered without such a graph abstraction. We also plan to make temporal dependencies more actionable from a database perspective. We are convinced that such dependencies can be integrated into continuous query engines. Indeed, some temporal dependencies could be the basis of a semantic indexation of data sources to better support human monitoring or object tracking in a set of cameras.

References

[1] C. C. Aggarwal, editor. *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer, 2007.

[2] C. C. Aggarwal, Y. Li, P. S. Yu, and R. Jin. On dense pattern mining in graph streams. *PVLDB*, 3(1):975–984, 2010.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[4] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *PVLDB*, 1(1):66–77, 2008.

[5] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[6] P. S. Arya. *Introduction to micrometeorology*, volume 79. Academic press, 2001.

[7] M. Berlingerio, F. Pinelli, M. Nanni, and F. Giannotti. Temporal mining for interactive workflow data analysis. In *KDD*, pages 109–118, 2009.

[8] M. Bester, E. Frind, J. Molson, and D. Rudolph. Numerical investigation of road salt impact on an urban wellfield. *Ground water*, 44(2):165–175, 2005.

[9] M. Böttcher, F. Höppner, and M. Spiliopoulou. On exploiting the power of time in data mining. *SIGKDD Explorations*, 10(2):3–11, 2008.

[10] T. Calders, N. Dexters, J. J. M. Gillis, and B. Goethals. Mining frequent itemsets in a stream. *Inf. Syst.*, 39:233–255, 2014.

[11] L. Chang, T. Wang, D. Yang, and H. Luan. Seqstream: Mining closed sequential patterns over stream sliding windows. In *IEEE ICDM*, pages 83–92, 2008.

[12] G. Chen, X. Wu, and X. Zhu. Sequential pattern mining in multiple streams. In *IEEE ICDM*, pages 585–588, 2005.

[13] Y. Chen, C. Chen, W. Peng, and W. Lee. Mining correlation patterns among appliances in smart home environment. In *PAKDD*, pages 222–233, 2014.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[15] A. J. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. M. White. Cayuga: A general purpose event monitoring system. In *CIDR*, pages 412–422, 2007.

[16] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W.-P. Hsiung, and K. S. Candan. Runtime semantic query optimization for event stream processing. In *ICDE*, pages 676–685, 2008.

[17] C. Faloutsos, T. G. Kolda, and J. Sun. Mining large graphs and streams using matrix and tensor tools. In *SIGMOD Conference*, page 1174, 2007.

[18] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering*, 23(5):683–698, 2011.

[19] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. In *Data Mining: Next Generation Challenges and Future Directions.* AAAI/MIT Press, 2004.

[20] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In *SDM*, 2006.

[21] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *VLDB J.*, 20(5):695–719, 2011.

[22] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.

[23] J. Han, Y. Chen, G. Dong, J. Pei, B. W. Wah, J. Wang, and Y. D. Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distributed and Parallel Databases*, 18(2):173–197, 2005.

[24] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. *Intell. Data Anal.*, 6(3):237–255, 2002.

[25] C. Jermaine. Finding the most interesting correlations in a database: how hard can it be? *Inf. Syst.*, 30(1):21–46, 2005.

[26] R. Jin and G. Agrawal. An algorithm for in-core frequent itemset mining on streaming data. In *IEEE ICDM*, pages 210–217, 2005.

[27] R. Jin and G. Agrawal. Frequent pattern mining in data streams. In C. C. Aggarwal, editor, *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*, pages 61–84. Springer, 2007.

[28] E. J. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

[29] A. Kotsifakos, P. Papapetrou, J. Hollmén, D. Gunopulos, V. Athitsos, and G. Kollios. Hum-a-song: A subsequence matching with gaps-range-tolerances query-by-humming system. *PVLDB*, 5(12):1930–1933, 2012.

[30] M. Li, M. Mani, E. A. Rundensteiner, and T. Lin. Constraint-aware complex event pattern detection over streams. In *DASFAA*, pages 199–215, 2010.

[31] M. Li, M. Mani, E. A. Rundensteiner, and T. Lin. Complex event pattern detection over streams with interval-based temporal semantics. In *DEBS*, pages 291–302, 2011.

[32] M. Liu, M. Li, D. Golovnya, E. A. Rundensteiner, and K. T. Claypool. Sequence pattern query processing over out-of-order event streams. In *ICDE*, pages 784–795, 2009.

[33] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, 1997.

[34] L. F. Mendes, B. Ding, and J. Han. Stream sequential pattern mining with precise error bounds. In *IEEE ICDM*, pages 941–946, 2008.

[35] M. Meriano, N. Eyles, and K. W. Howard. Hydrogeological impacts of road salt. *Journal of contaminant hydrology*, 107(1):66–81, 2009.

[36] F. Mörchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *SIAM SDM*, pages 315–326, 2010.

[37] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, pages 226–236, 2000.

[38] E. Nazerfard, P. Rashidi, and D. J. Cook. Using association rule mining to discover temporal relations of daily activities. In *ICOST*, pages 49–56, 2011.

[39] D. Patel, W. Hsu, and M.-L. Lee. Mining relationships among interval-based events for classification. In *SIGMOD Conference*, pages 393–404, 2008.

[40] Z. Pawlak. Rough set theory and its applications to data analysis. *Cybernetics & Systems*, 29(7):661–688, 1998.

[41] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Psychology Magazine*, 1:157–175, 1900.

[42] C. Raïssi and M. Plantevit. Mining multidimensional sequential patterns over data streams. In *DaWaK*, pages 263–272, 2008.

[43] P. Rashidi and D. J. Cook. Mining sensor streams for discovering human activity patterns over time. In *IEEE ICDM*, pages 431–440, 2010.

[44] P. Rashidi, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Trans. Knowl. Data Eng.*, 23(4):527–539, 2011.

[45] C. Robardet, V.-M. Scuturici, M. Plantevit, and A. Fraboulet. When teddy meets grizzly: temporal dependency discovery for triggering road deicing operations. In I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 1490–1493. ACM, 2013.

[46] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43 – 49, feb 1978.

[47] P. shan Kam and A. W.-C. Fu. Discovering temporal patterns for interval-based events. In *DaWaK*, pages 317–326, 2000.

[48] L. Tang, T. Li, and L. Shwartz. Discovering lag intervals for temporal dependencies. In *KDD*, pages 633–641, 2012.

[49] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. J. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Min. Knowl. Discov.*, 26(2):275–309, 2013.

[50] E. Winarko and J. F. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data Knowl. Eng.*, 63(1):76–90, 2007.

[51] S.-Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Trans. Knowl. Data Eng.*, 19(6):742–758, 2007.

[52] J. Yao and Y. Yao. Induction of classification rules by granular computing. In *Rough Sets and Current Trends in Computing*, pages 331–338. Springer, 2002.