TR 11-010

Cascading spatio-temporal pattern discovery

Pradeep Mohan, Shashi Shekhar, James A. Shine, and James P. Rogers

May 02, 2011

Cascading spatio-temporal pattern discovery

Pradeep Mohan, Shashi Shekhar, James A. Shine, and James P. Rogers

Pradeep Mohan and Shashi Shekhar are with the Department of Computer Science, University of Minnesota, Twin-Cities, MN 55455
E-mail: {mohan,shekhar}@cs.umn.edu
James A. Shine and James P. Rogers are with the Engineering Research and Development Center, US Army Corps of Engineers
E-mail: {James.A.Shine,James.P.Rogers.II}@usace.army.mil

Abstract

Given a collection of Boolean spatio-temporal (ST) event-types, the cascading spatio-temporal pattern (CSTP) discovery process finds partially ordered subsets of these event-types whose instances are located together and occur serially. For example, analysis of crime datasets may reveal frequent occurrence of misdemeanors and drunk driving after and near bar closings on weekends, as well as after and near large gatherings such as football games. Discovering CSTPs from ST datasets is important for application domains such as public safety (e.g. identifying crime attractors and generators) and natural disaster planning(e.g. preparing for hurricanes). However, CSTP discovery presents multiple challenges; three important ones are (1) the exponential cardinality of candidate patterns with respect to the number of event types, (2) computationally complex ST neighborhood enumeration required to evaluate the interest measure and (3) the difficulty of balancing computational complexity and statistical interpretation. Current approaches for ST data mining focus on mining totally ordered sequences or unordered subsets. In contrast, our recent work explores partially ordered patterns. Recently, we represented CSTPs as directed acyclic graphs; proposed a new interest measure, the cascade participation index; outlined the general structure of a cascading spatio-temporal pattern miner (CSTPM); evaluated filtering strategies to enhance computational savings using a real world crime dataset and proposed a nested loop based CSTPM to address the challenge posed by exponential cardinality of candidate patterns. This paper adds to our recent work by offering a new computational insight, namely, that the computational bottleneck for CSTP discovery lies in the interest measure evaluation. With this insight, we propose a new CSTPM based on spatio-temporal partitioning that significantly lowers the cost of interest measure evaluation. Analytical evaluation shows that our new CSTPM is correct and complete. Results from significant amount of new experimental evaluation with both synthetic and real data show that our new ST partitioning based CSTPM outperforms the CSTPM from our previous work. We also present a case study that verifies the applicability of CSTP discovery process.

Index Terms

cascading spatio-temporal patterns, space-time K-Function, Cascade Participation Index, spatio-temporal join, spatio-temporal continuity, positive ST autocorrelation, spatio-temporal partial order.

I. Introduction

We consider a set of Boolean spatio-temporal (ST) event types and their instances. We use Boolean event types because we are primarily concerned with either the occurrence or absence of an event type at a particular location and time. Given such a set, cascading spatio-temporal patterns (CSTPs) are partially ordered subsets of event types whose instances are spatial neighbors and occur in a series of stages. Figure 1 shows an example of a CSTP. The first event[1] in the CSTP is the hurricane event. Subsequent events are represented by events such as heavy rainfall, localized flooding and wind damage. Figure 2(a) shows an example of a CSTP from an urban crime dataset. The first event here is bar-closings (represented by circles), and subsequent events are assaults (represented by triangles) and drunk driving (represented by squares). Figures 2 (b), (c) and (d) show indvidual instances of events of bar-closing, assault and drunk driving with their location and time. Figure 2(e) shows the locations of all event instances. The individual instances of the CSTP in Figure 2(a) are shown as three striped triangles in Figure 3.

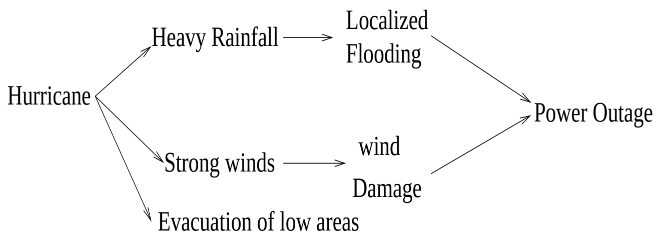


Fig. 1. CSTPs occurring in connection with a hurricane

Bars in large cities are often considered as potential generators of crime that occurs after bar closing time [1]. In crime analysis, CSTPs may suggest interesting hypotheses relating several crime types, which may help law enforcement agencies, public safety groups and policy makers to determine appropriate action for crime mitigation. CSTP discovery is also important in a number of other application domains, including climate change science (e.g. understanding the effects of climate change on food supply[2]) and public health (e.g. tracking the emergence, spread and re-emergence of multiple infectious diseases[3]).

However, CSTP discovery poses three key challenges : (1) exponential cardinality of candidate patterns with respect to the number of event types, makes the problem combinatorially complex [4]; (2) computationally complex ST neighborhood enumeration to evaluate the interest measure and; (3) conflicting demands of computational scalability (i.e. measures are computable in polynomial time) and statistical interpretation (i.e. measures can detect ST correlation as in ST statistics).

Related Work and Additional Contributions: Most previous work on ST frequent pattern mining has focused on discovering unordered patterns or totally ordered patterns. Unordered

[1]For brevity, the phrases "event" and "event-type" are used interchangeably in this paper.

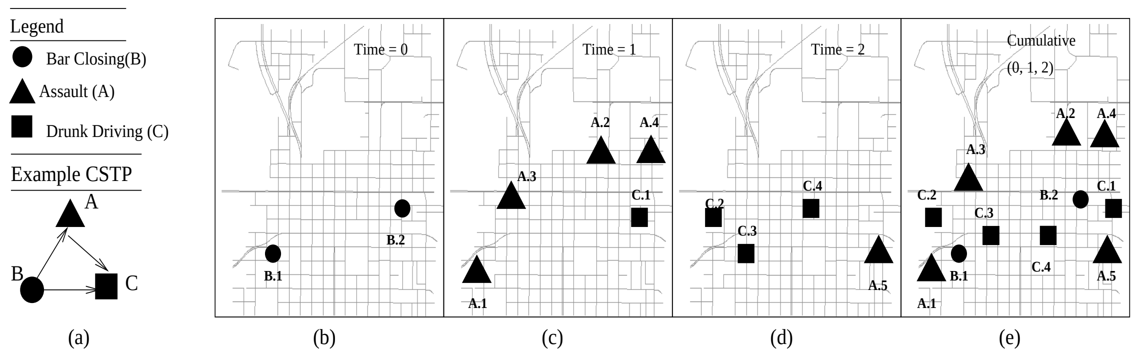


Fig. 2. Illustrative ST crime dataset and cascading ST pattern

patterns (e.g. spatial co-locations and topological patterns) represent unordered subsets of spatial or ST events in a uniform ST framework [5], [6]. Totally ordered patterns (e.g. ST sequences) represent a linear chain reaction of ST event types [7]. In a partially ordered set, for some (but possibly not all) pairs of elements, one of the elements preceedes the other. Totally ordered sets are special cases of partial order where, for all pairs of elements, one of the elements precede the other. In an effort to represent the partial ordered nature of space and time, our research explores cascading ST patterns as directed acyclic graphs.

In our recent work [8], [9], [10], we proposed: (1) a novel CSTP interest measure to quantify statistically meaningful CSTPs; (2) a simple nested loop based CSTPM (SIAM NL-CSTPM, previously called CSTP Miner) to evaluate the interest measure; (3) two novel filtering strategies to discover CSTPs from ST datasets; and (4) primary data analysis to reveal interesting CSTPs from real crime datasets [8]. Our previous algorithm primarily relied on efficient filtering strategies to reduce the number of interest measure evaluations. However, it did not explore ways to reduce the cost of evaluating the interest measure for remaining candidates and did not demonstrate the generalizability of experimental results beyond one real dataset.

Additional contributions in this paper: This paper extends our preliminary work on CSTP discovery [8]. Our new contributions are as follows:

(1) We conducted a bottleneck analysis of the CSTPM which reveals that the bulk of computation time is taken by interest measure evaluation (involving a ST join and pruning) and that candidate generation is a small proportion of computation time by comparison.

(2) We introduce a new algorithm to compute the interest measure: ST partitioning based CSTPM (called as TKDE STP-CSTPM) and show that it does not impact either correctness or completeness of the CSTPM.

(3) We provide an algebraic cost model of the filtering strategies proposed in our previous version.

(4) We experimentally show that TKDE STP-CSTPM (TKDE for short) outperforms the SIAM NL-CSTPM (SIAM for short) using synthetic and real ST crime datasets.

(5) We generalize our previous experimental results to synthetically generated datasets for evaluating the previously proposed filtering strategies (upper bound (UB) filter and multi-resolution spatio-temporal (MST) filter) and include a new synthetic data parameter (clumpiness degree) that controls the extent of positive ST autocorrelation. We also present a filter evaluation on larger real datasets.

(6) We provide an extended case study to find CSTPs from real datasets, and a revised view of CSTP mining processes and the CSTP candidate space in order to illustrate the new algorithm.

Scope: This paper focuses on the computational aspects of discovering CSTPs. This paper does not address ST nonstationarity, the choices of directed neighborhood relationships and interest measure thresholds, or the motion of ST instances. Determining the appropriate grid cell size is also beyond the scope of this paper; while grid cell sizes are crucial to MST filter performance, they are dependent on the application domain.

Convention: The term positive ST autocorrelation refers to a clustering of event-type instances within a ST framework, and not other interpretations of correlation such as cross correlation.

Outline: The rest of this paper is organized as follows: Section II reviews some necessary basic concepts from our previous paper and formulates the CSTP mining problem. Section III reviews the challenges of CSTP discovery, provides a bottleneck analysis of the CSTPM, describes the new optimization (TKDE), reviews the previous filtering strategies (the upper bound(UB) filter and the multiresolution ST (MST) filter) and provides an analytical evaluation to show that all CSTPM versions are correct and complete and discover statistically meaningful patterns. Section IV presents the experimental evaluation using synthetic and real datasets, and an extended case study to find real CSTPs. In Section V we discuss some important differences between traditional data mining and CSTP mining. Section VI concludes the paper and discusses future work.

II. Problem Formulation

The special properties of ST data (e.g. ST correlation, non-linearity, continuity, partial order etc.) motivate different data models and/or representations. This section describes some basic concepts related to ST data modeling, defines an interestingness measure and formulates the CSTP mining problem.

A. Modeling ST data:

ST data is often modeled using events and processes, both of which generally represent change of some kind. Processes refer to ongoing phenomena that represent activities of one or more types without a specific endpoint, such as global climate change [11], [12]. Events refer to individual occurrences of a process with a specific beginning and end. Event-types and event-instances are distinguished. For example, a hurricane event may occur at many different locations and times e.g., Katrina (New Orleans, 2005) and Rita (Houston, 2005). Each event-instance is associated with a particular occurrence time and location. The ordering may be total if event-instances have disjoint occurrence times. Otherwise, ordering is partial.

Partial order over a set of event-instances may be constrained further to define a **directed neighbor relation (R)** in which distance in space, time or both is restricted by a threshold. For example, a drunk driving crime and a bar-closing instance may be considered directed neighbors if separated by a few miles within a reasonable travel time. Partial (or total) order may also be defined on event-types, possibly to represent simple processes. For example, Figure 1 shows a partial order among the event types of hurricane, heavy rainfall, strong winds, evacuation of local areas, localized flooding, wind damage and power outage.

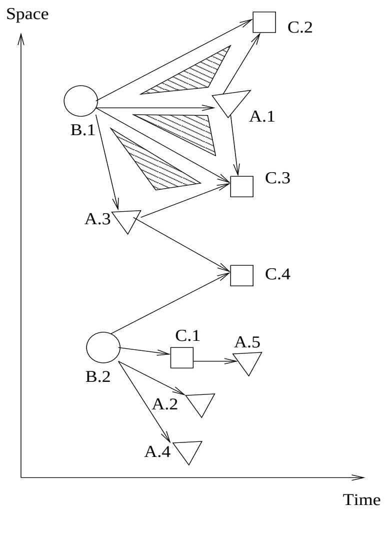


Fig. 3. Directed ST neighborhood graph

A directed neighbor relationship over a set MI of event-instances may be formally represented as a directed acyclic graph, $GI = (MI, EI)$, where EI is a set of directed edges representing ordered pairs in $MI\ X\ MI$. For example, the application of a **directed neighbor relation (R)** on the illustrative dataset shown in Figure 2(a-d) produces the directed acyclic ST neighborhood graph shown in Figure 3. The edges in the graph are computed using a CSTPM.

B. Quantifying the interestingness of CSTPs:

The interestingness of CSTPs may be measured in different ways (e.g. support, joint-probability, conditional probability etc.). In data mining, interest measures are selected using criteria such as computational scalability to large datasets, ease of interpretation, and utility in the context of the application domains. In an ST data mining context, the goal of interest measure selection is to balance the conflicting requirements of computational scalability and statistical interpretation. A key application domain constraint is the ability to predict the instances of a CSTP given an instance of a participating event type. These considerations have lead us to create an interest measure based on conditional probability.

A **Cascade Participation Ratio, CPR(CSTP,M)**, may be interpreted as an estimate of the conditional probability of an instance of a pattern CSTP given an instance of event type M, i.e. $Pr(CSTP|M)$. Formally, CPR(CSTP,M) can be written as
$CPR(CSTP, M) = \frac{\#instances\ (M)\ participating\ in\ CSTP}{\#instances\ (M)\ in\ Dataset}$,
where M is a participating event type in the CSTP.

A **Cascade Participation Index , CPI(CSTP)**, of a pattern is a measure of the likelihood of an instance of a pattern CSTP in the ST neighborhood of an instance of a participating event-type. By definition, CPI(CSTP) is the minimum of CPR(CSTP,M) over all event-types in a CSTP. Since CPR(CSTP,M) may be interpreted as an estimate of the conditional probability of an instance of a pattern given an instance of event type M, CPI(CSTP) may be viewed as a lower bound on the conditional probability $Pr(CSTP|M)$ for any participating event type M. CPI can be formally written as: $CPI = min\{CPR(CSTP, M)\}$ where M is a participating event type in the CSTP. For example, for the CSTP shown in Figure 2, two of the four instances of A belong to a CSTP, so the CPR for A is 2/5 or 0.4. Similarly, the CPR for B and C is 1/2 and 2/4, so the CPI is $CPI = min\left\{\frac{1}{2}, \frac{2}{5}, \frac{2}{4}\right\} = \frac{2}{5} = 0.4$.

CPI is a reliable measure as it is useful for applications that involve predicting the near future occurrence of a pattern in the vicinity of an observed instance of a participating event-type. Morover, it is also computable in polynomial time (e.g. quadratic in datasize). Since ST datasets are partially ordered, this constraint is imposed in the ST neighbor relation specified as input to the CSTP discovery process. More discussion regarding reliability and use of partial order constraint to compute CPI can be found in our recent paper [8].

Based on the above definitions, the CSTP mining problem can be defined as follows:

Given : (a) a ST dataset consisting of a set of Boolean event-types over a common ST framework; (b) a directed neighbor relation R; and (c) a threshold for the CPI.

Find : CSTPs with CPI $\geq$ the user specified threshold

Objective : Minimize computation time.

Constraints : (a) correct and complete sets of CSTPs are discovered; (b) CSTP interest measures find statistically meaningful CSTPs.

Example: In public safety, a set of crime reports with locations, time stamps and event types may represent a ST dataset (as in Figure 2) and events such as bar-closing, drunk driving etc. may represent Boolean event types. The directed neighbor relation R can be defined by using distance (e.g. 0.5 miles, 5 miles etc.) or time (e.g. minutes, hours, days etc.) thresholds. The CSTP discovery problem does not require the number of events in a CSTP as an input.

III. Challenges and Solutions

In this section, we identify the key computational challenges in CSTP mining and present a computational bottleneck analysis of the CSTPM. We outline a general structure of the CSTPM algorithm, present filtering strategies and specific algorithms, provide an analytical evaluation, and briefly outline an algebraic cost model.

A. Broad Challenges:

CSTP mining poses three broad challenges. First, spatio-temporal datasets (e.g crime data) consist of many different event-types. The cardinality of candidate CSTPs is exponential in the number of event-types. CSTPs may be viewed as directed acyclic graphs (DAGs) and the event types in a CSTP may be viewed as nodes of a DAG with distinct node labels. A detailed mathematical relationship between distinct node labels and the cardinality of DAGs has been explored in graph theory [4]. This makes filtering strategies desirable particularly for datasets where many candidates are non-prevalent.

A second challenge in CSTP discovery is the need for computationally complex ST neighborhood enumeration to evaluate the interest measure. This challenge can be addressed by either a neighborhood graph enumeration approach or a ST join based approach. However, since neighborhoods overlap in a ST framework, enumeration strategies (e.g. graph based, join based) are required to examine all combinations of ST relations between $|D|$ data instances, leading to an $O(|D|^2)$ join computation cost.

Our third broad challenge is confronting the ST data mining tradeoff between computational complexity and statistical interpretation. Statistical interpretation implies that a candidate interest measure or pattern discovery process can detect significant ST correlation that exists in the data.

The CPI can be shown to address this complex tradeoff since it can be proved to be anti-monotonic and an upper bound to a ST statistical measure called the space-time K-Function, which is a generalization of the spatial statistical measure Ripley's K [13], [14], [15]. Solutions to different challenges are based on two principles: (1) prevent unneccessary interest measure computation and (2) speed up interest measure computation.

A natural way to deal with the exponential candidate space is either to filter out non-prevalent candidates even before generating them or to overestimate the interest measure of candidate patterns and filter them if this overestimated measure does not pass the threshold. The first strategy can be realized due to the existence of an anti-monotone upper bound to the CPI and is defined as the upper bound (UB) filter. The second strategy can be realized by making use of the property of ST autocorrelation [15]. Using ST autocorrelation leads to a clustering of similar events in localized neighborhoods and can improve computational efficiency by aggregating ST event-types to an underlying user defined grid framework. Aggregation leads to an overestimation in CPI and helps in filtering non-prevalent patterns. This filtering strategy is defined as the multi-resolution spatio-temporal filter (MST filter). The CPI can be computed using a purely ST join strategy or a sub-graph enumeration strategy. CPI evaluation alters the computational structure.

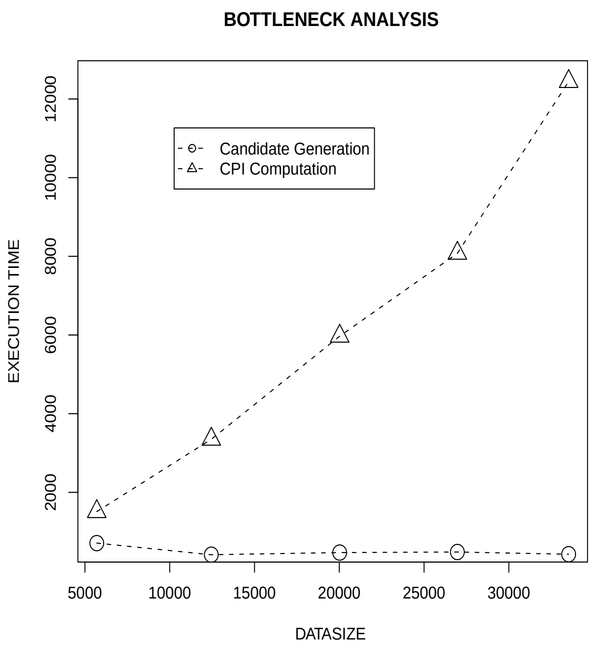


Fig. 4. CPI evaluation is the key bottleneck

B. *Bottleneck Analysis:*

CSTP discovery involves two key operations: (1) candidate generation and (2) interest measure evaluation. Determining which of these two operations is more costly in terms of computation is important for the development of our techniques. For this reason, we performed a thorough bottleneck analysis using computation time as the metric for candidate generation (the first operation) and interest measure evaluation (the second operation). We compared computation times for the two steps on real crime data from Lincoln, NE. The datasets varied in size from 5000 to 34000 crime incidents. Our analysis shows that CPI evaluation constitutes a major portion of the computational cost. Figure 4 shows the computation time in seconds corresponding to a spatial neighborhood of 1 kilometer, a temporal neighborhood of 1 hour, and a CPI of .003 to maximize the number of patterns generated. CSTPs of size 1, 2 and 3 were generated sequentially and the total execution time was plotted. The execution times for CPI evaluation (the ST join and pruning) are significantly greater than for candidate generation. The difference is closest for the smallest data set, since that generated more patterns for size 2 and 3 CSTPs than the larger data sets due to the nature of the CPI metric, which can shrink with larger data sets. This is an issue we will need to address in future work. However, it is obvious that unlike any other data mining situations, the CPI evaluation takes much more computational time than the candidate generation, and efficient strategies to resolve this bottleneck (and not the candidate generation bottleneck) are required. Based on the broad challenges for CSTP mining and insight provided by the bottleneck analysis, we describe the CSTPM algorithm in the next section.

C. CSTPM Algorithm

The CSTPM algorithm generates all CSTPs with a CPI value greater than or equal to a user specified threshold. The algorithm contains two key steps: (1) candidate generation, (2) and

$Algorithm: \ CSTPM$

Input: (a) M Boolean ST event types and their instances;(b) A user specified ST partial ordered neighbor relation R; (c) A single user specified interest measure threshold.

Output: A set of CSTPs with interest measure $\geq$ threshold.

Variables: (a) k: Pattern size (number of edges in a CSTP); (b) Optimization flags to activate UB and MST filters.

Method:

```
1. For size of patterns in (1,2...k) do
2.     If(UB is TRUE)
3.         Perform upper bound(UB) filtering. //Prevent non-prevalent candidate generation.
4.     Generate candidate CSTPs of size k using CSTPs of size k-1
5.     If(MST is TRUE)
6.         Perform MST filtering. prune out non-prevalent patterns by over estimating CPI.
7.     Generate pattern instances and compute CPI.
8.     Prune CSTPs based on their prevalence.
9.     Generate prevalent CSTPs
10. End
```

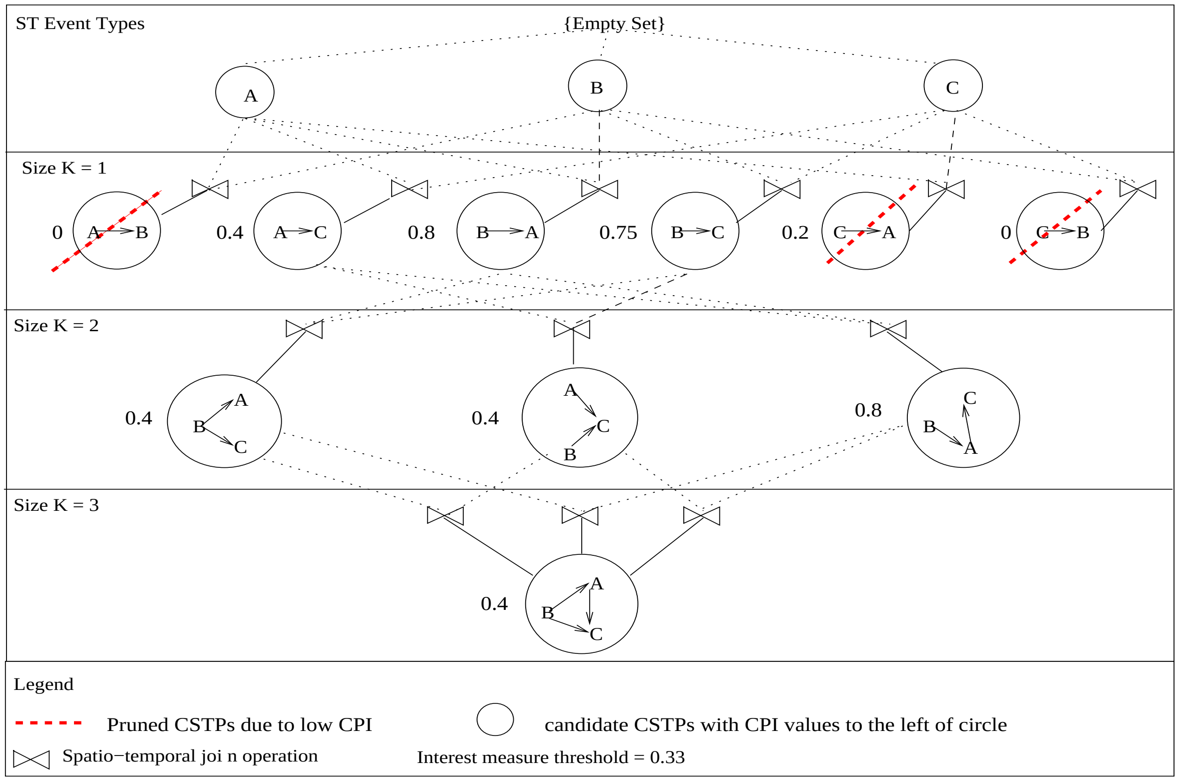

ST Event Types
{Empty Set}
A
B
C
Size K = 1
0
A→B
0.4
A→C
0.8
B→A
0.75
B→C
0.2
C→A
0
C→B
Size K = 2
0.4
0.4
0.8
Size K = 3
0.4
Legend
Pruned CSTPs due to low CPI
candidate CSTPs with CPI values to the left of circle
Spatio−temporal joi n operation
Interest measure threshold = 0.33

Fig. 5. Illustration of the CSTPM algorithm

interest measure evaluation. Performance optimization is conducted before candidate generation, and before and during interest measure evaluation.

The algorithm enumerates the space of candidate patterns as shown in Figure 5. The figure shows the CPI values of CSTPs on the left side of individual patterns. The CSTPs crossed out with dotted lines do not pass the CPI threshold set at 0.33; these patterns can potentially be pruned using filtering strategies. Without pruning, larger size candidate extensions of these patterns will be generated and CPI values will be computed, adding to computation time.

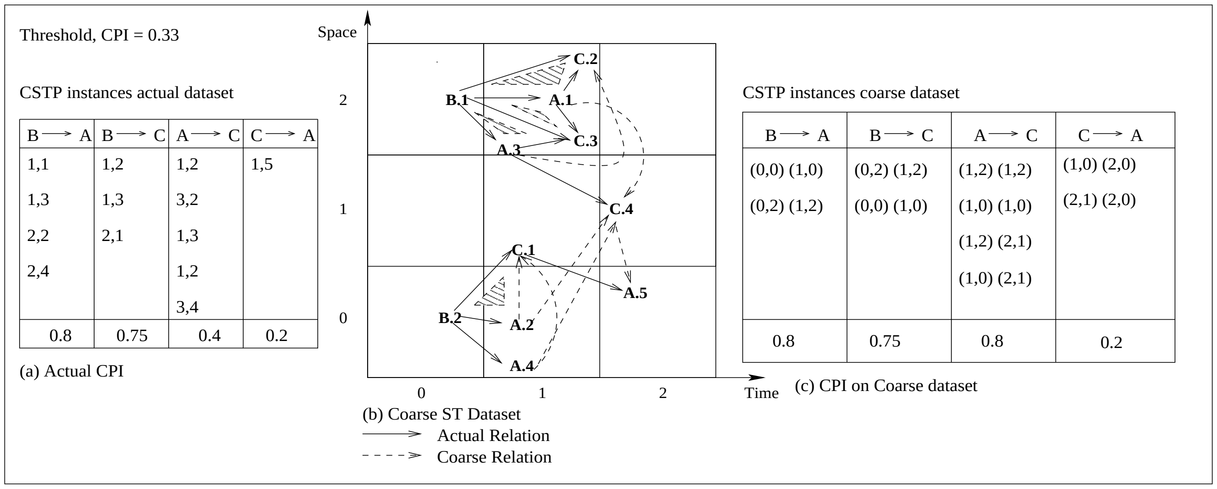
D. Filtering strategies

The CSTPM filters non-prevalent patterns early (i.e. before CPI computation) by using two filters: an upper bound (UB) filter and a multi-resolution spatio-temporal (MST) filter.

1) Upper Bound Filter: The upper bound (UB) filter is based on the existence of an upper bound for the CPI. We first prove that there exists an upper bound to the CPI. In some cases, the CSTPM might generate candidates that could potentially have low interest measure values. Such uninteresting candidates might possibly slow down computation. Hence, we make use of this strategy to prevent uninteresting candidate generation. These upper bound values reflect the maximum possible value of the interest measure for a candidate CSTP.

The **upper bound of the CPI, upper(CPI)**, is the ratio of the minimum and maximum value of the CPR of event-types participating in a CSTP. It can be formally written as: $upper(CPI) = \frac{min\{CPR(CSTP,M)\}}{max\{CPR(CSTP,M)\}}$,
where M is a participating event type in the CSTP.
It is clear from the above definition that $upper(CPI)$ is an upper bound to the CPI because the CPI is the lowest value $upper(CPI)$ can take (when the denominator is 1).

The UB filter is used before candidate generation. When merging two size k-1 CSTPs to generate a candidate size-k CSTP, we compute the upper bound for the candidates to be generated. If this upper bound is less than the user specified threshold, then we do not proceed with candidate generation. For example, Figure 5 shows a few size k=1 patterns, crossed out with thick lines, that would be pruned by the UB filter before candidate generation. This filtering saves the initial cost of candidate generation and the subsequent cost of computing the ST join required for calculating the CPI.



Threshold, CPI = 0.33

CSTP instances actual dataset

$B \longrightarrow A$	$B \longrightarrow C$	$A \longrightarrow C$	$C \longrightarrow A$
1,1	1,2	1,2	1,5
1,3	1,3	3,2	
2,2	2,1	1,3	
2,4		1,2	
		3,4	
0.8	0.75	0.4	0.2

(a) Actual CPI

CSTP instances coarse dataset

$B \longrightarrow A$	$B \longrightarrow C$	$A \longrightarrow C$	$C \longrightarrow A$
(0,0) (1,0)	(0,2) (1,2)	(1,2) (1,2)	(1,0) (2,0)
(0,2) (1,2)	(0,0) (1,0)	(1,0) (1,0)	(2,1) (2,0)
		(1,2) (2,1)	
		(1,0) (2,1)	
0.8	0.75	0.8	0.2

(c) CPI on Coarse dataset

Fig. 6. Illustration of the multi-resolution spatio-temporal filter

2) Multi Resolution ST Filter: The MST filter exploits the geographic clustering of event instances in a ST framework. In ST frameworks, event instances of the same event type could possibly cluster together due to the presence of positive ST autocorrelation. For example, crimes such as auto thefts, assaults, and vandalism are sometimes clustered in high crime neighborhoods such as bar districts or entertainment places. A uniform ST grid defined using the parameters d and t (see Figure 6(b)) is overlaid on the ST dataset. The grid parameters themselves can be specified using commonly known notions like census blocks and common time periods spanning a few hours. The instances of each event type are assigned to specific grids to obtain a coarse (or aggregated) dataset. This procedure is similar to the pagination imposed by a standard ST index structure. Based on this new coarse dataset, a new coarse directed neighbor relation R^C is derived. Two grid cells i and j are neighbors under R^C, if cell i contains an instance x and cell j contains an instance y such that x and y are ST neighbors under relation R defined in Section IIA. For example, in Figure 6, cells $(0,0)$ and $(1,1)$ are neighbors under R^C. During MST filtering, the coarse dataset is substituted for the original dataset. For every candidate CSTP, the MST filter generates a set of CSTP instances on the coarse dataset and computes a coarse CPI. The key idea is that the coarse CPI is an upper bound to the actual CPI for a CSTP. If the coarse CPI is less than the user specified threshold, the pattern is pruned. The coarse CPI can also computed by performing a ST join.

Figures 6 illustrates MST filtering for size 1 patterns. Figure 6a shows the actual instances and Figure 6c shows that the MST filter overestimates the value of the interest measure. For example, the value of CPI for CSTP $A \rightarrow C$ on the actual datset is 0.4, but it is 0.8 on the coarse dataset. Figure 6b shows the coarse neighbor between event instances by means of a broken edge while the actual neighbor relation is represented by the thick edges.

Lemma 1: **MST filtering never underestimates the value of the interest measures compared to the original dataset.**

Proof: MST filtering has two key ideas: (1) instance assignment to grids and (2) coarsening R to R^C. The key intuition behind overestimation of CPI by the filter is that coarsening never reduces the number of neighbors for a particular event. A detailed explanation is available in our previous work [8]. ■

E. Algorithms for Computing the CPI

The definition of CPI suggests that computing the CPI of a CSTP involves performing a ST neighborhood search around each instance of an event type that is part of a CSTP. This operation can be performed by using a ST Join. Figure 5 illustrates this abstraction using the join operator between CSTPs of different sizes. Computing an ST join can be performed using several techniques. We describe a new algorithm for achieving this: a spatio-temporal partition based CSTPM [16]. Figure 7 shows a block diagram overview of the two different CSTPM algorithms. In our recent work presented at the SIAM International Conference on Data Mining, we explored a nested loop based CSTPM shown by the box with dotted lines. The new strategy proposed in this paper is shown in the box with a thick boundary. To summarize, the SIAM algorithm uses a nested loop approach, the TKDE algorithm uses a spatio-temporal partitioning approach, and both algorithms are CSTPMs.

TKDE CSTPM: the TKDE algorithm computes the CPI by making use of the classical

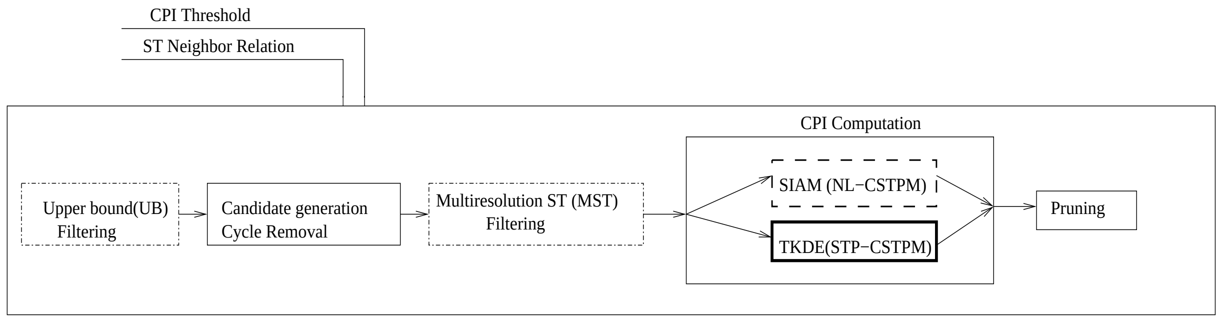

CPI Threshold
ST Neighbor Relation
CPI Computation
Upper bound(UB) Filtering
Candidate generation Cycle Removal
Multiresolution ST (MST) Filtering
SIAM (NL−CSTPM)
TKDE(STP−CSTPM)
Pruning

Fig. 7. Overview of different CSTP Mining algorithms

principle of filter and refine. Given, a size-(k) pattern $CSTP_{AB}$ that was generated using two patterns size-(k-1) patterns, namely, $CSTP_A$ and $CSTP_B$, the algorithm initially determines the ST data universe and partitions the universe into P partitions. Since each pattern instance can be approximated by a minimum bounding box (MBB), the partitions that overlap with these bounding boxes are determined first. The same procedure is repeated for both sets of pattern instances. Once ST instances are assigned to partitions, each partition corresponding to the two different instance sets are joined using a geometric plane sweep technique. Compared to the traditional 2D plane sweep that could be done with spatial data, computing an ST join between any two partitions involves an extra scan along the temporal dimension. Due to the natural directionality provided by time, sorting MBBs is done in the temporal dimension. The refinement step involves merging the actual pattern instances within a pair of related MBBs and generating an instance of $CSTP_{AB}$.

F. Analytical Evaluation:

We show that both the versions of CSTPM are correct and complete. Both CSTPMs address the tradeoff between computational efficiency and statistical interpretation. This is due to the computational and statistical properties of the CPI. We prove that the CPI and its upper bound are anti-monotonic (sometimes called monotonic decreasing). Anti-montonicity is an essential property for computational efficiency of the CSTPM. The CPI draws its statistical interpretation due to its upper bound relationship with a common spatial statistical interest measure, the space-time K-Function [14].

Theorem 1: **The CSTPM is correct.**

Proof: By correctness we mean that no spurious patterns are generated.

Spurious pattern generation is avoided in the CSTPM by computing the CPI correctly and by generating valid CSTP candidates. There are two key elements to correctness: (1) CSTPs are directed acyclic graphs and (2) the pattern instances are computed correctly. The former constraint is ensured by Step 4 of the CSTPM algorithm. In step 4, the CSTPM algorithm generates candidates by procedure similar to graph mining [19]. Once these candidate are generated, acyclic patterns are generated by scanning the list of graph candidates by checking if the pattern is a valid CSTP. If its a valid CSTP its added to the candidate list otherwise, it is removed.

The latter constraint is ensured by Step 7 of the CSTP algorithm which computes a table of CSTP instances by one of the ST join strategies (e.g. nested loop or spatiotemporal partition). The spatiotemporal partitioning strategy keeps track of CSTP instance overlaps across multiple partitions and correctly reports all instances of a candidate CSTP while removing duplicates. Hence it ensures that, the number of instances for each CSTP is reported correctly. This guarantees that the CPI of a pattern is computed correctly. Hence, in Step 8 of the CSTPM algorithm only correct patterns pass the pruning test. Hence the CSTPM Algorithm is correct. ■

Theorem 2: **The CSTPM is complete.**

Proof: By completeness we mean that the CSTPM does not miss any valid patterns and that all prevalent patterns are reported. The key elements of completeness are, (1) filters do not eliminate valid patterns, (2) CPI is computed correctly and (3) valid patterns are not eliminated during CSTP candidate generation. The first constraint is guaranteed by Lemma 1 and the upper bound property of CPI. As was observed earlier, both filters overestimate the value of the CPI. This guarantees that no valid pattern would get removed during filtering. The second constraint is guaranteed by Theorem 1, and the third is ensured by Step 4 of the algorithm which does not eliminate valid CSTPs during candiate generation. Hence CSTPM is complete. ■

Theorem 3: **The CPI and its upper bound are anti-monotonic.**

Proof: The key insight behind the proof is that the value of the CPI and its upper bound are non-increasing with pattern size. We prove this by considering two CSTPs, $CSTP(k)$ and $CSTP(k-1)$, which represent CSTPs of size k and $k-1$ respectively, and establishing that $CPI(CSTP(k)) \leq\ CPI(CSTP(k-1))$ under the addition of an edge. The proof of anti-monotonicity of the upper bound of the CPI is based on the same intuition. Lemma 2 and Lemma 3 explain this idea in detail. ■

Lemma 2: **The CPI is Anti-monotonic.**

Proof: In order to prove CPI is anti-monotonic, we need to prove that the CPI value is non-increasing under edge addition. Let $\{e\} = (u,v) \in\ CSTP(k)\ and\ e \notin\ CSTP(k-1)$.

Here, u and v represent event types that are either $\in CSTP(k)$ or $\in CSTP(k-1)$. By the definition of CPI, we have

$$CPI(CSTP(k)) = min\left\{\ CPI(CSTP(k-1)), \frac{\#instances(u)}{\#instances(u)\ in\ Dataset}, \frac{\#instances(v)}{\#instances(v)\ in\ Dataset}\ \right\}$$

Since, the lower bound of the three quantities in the above step is taken, addition of a new event type to a CSTP will guarantee that only the least of the three fractions is assigned as the CPI. This implies that always, $CPI(CSTP(k)) \leq\ CPI(CSTP(k-1))$. Hence, the CPI is antimonotonic. ■

Lemma 3: **The CPI has an anti-monotonic upper bound**

Proof: Let $\{e\} = (u,v) \in\ CSTP(k)\ and\ e \notin\ CSTP(k-1)$.

Here, u and v represent event types that are either $\in CSTP(k)$ or $\in CSTP(k-1)$. By the definition of upper(CPI), we have,

$$upper(CPI(CSTP(k))) = \\ = \frac{min\left\{\ upper(CPI(CSTP(k-1))), \frac{\#instances(u)}{\#instances(u)\ in\ Dataset}, \frac{\#instances(v)}{\#instances(v)\ in\ Dataset}\ \right\}}{max\left\{\ upper(CPI(CSTP(k-1))), \frac{\#instances(u)}{\#instances(u)\ in\ Dataset}, \frac{\#instances(v)}{\#instances(v)\ in\ Dataset}\ \right\}}$$

After edge addition to a CSTP, there are 5 possible scenarios, namely, (1) numerator decreases and denominator remains same, (2) numerator and denominator remain same, (3) numerator decreases and denominator increases, (4) numerator increases and denominator remains same, and (5) numerator increases and denominator decreases. Scenarios (4) and (5) are not possible because, when a new event type is added that is lower than the CPI of the size(k-1) CPI, the min will automatically adjust or to the lower value and will remain the same if a higher valued even type is added. This implies that, only the first three scenarios are possible. This implies that always, $upper(CPI(CSTP(k))) \leq\ upper(CPI(CSTP(k-1)))$. Hence, upper(CPI) is anti-monotonic.

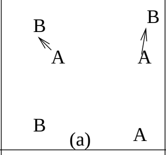
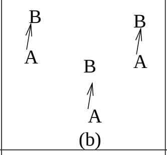
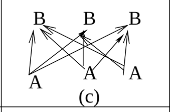
			
ST K−Function	2/9	1/3	9/9=1
CPI	2/3	1	1

Fig. 8. The CPI as an upper bound to the space-time K-Function

Lemma 4: **The CPI is an upper bound to the space-time K-Function** ■

Proof: From Definition of CPI and the definition of the space-time K-Function [14], we have

$CPI = min\left\{\frac{\#\ instances(CSTP,A)}{|A|}, \frac{\#\ instances(CSTP,B)}{|B|}\right\}$

$\frac{K_{AB}}{ST} = \frac{1}{ST^2} \cdot \frac{1}{\lambda_A \cdot \lambda_B} \sum \cdot_i \sum \cdot_j I_{ht}(d(A_i, B_j), t_d(A_i, B_j))$

$= \frac{\#\ instances(CSTP)}{|A| \cdot |B|}$

$\Rightarrow \#\ instances(CSTP, A) \geq \frac{\#\ instances(CSTP,A)}{|B|}$, Similarly,

$\Rightarrow \#\ instances(CSTP, B) \geq \frac{\#\ instances(CSTP,B)}{|A|}$ (either of the values is greater than the average number of instances of A around B and vice versa participating in CSTP)

$\Rightarrow CPI = upperbound(space - time\ K - Function)$

Figure 8 shows different cases of ST interaction between two event types A and B. In all three cases, the CPI is greater than or equal to the space-time K-Function.

G. Algebraic Cost Model

Our algebraic cost model analytically compares the computational costs of different CSTPM algorithms at two levels: (1) CPI computation and (2) filtering. In the case of CPI evaluation, the cost model shows that the proposed TKDE algorithm has an asymptotically better performance than a worst case $O(N^2)$ algorithm for each CSTP. It also shows that filtering to remove non-prevalent patterns yields computational savings. This is proved analytically by showing that the ratio of the costs with filtering to the costs without filtering is less than 1. Notation for the algebraic cost model is listed in Table I.

1) Analysis of CPI computation algorithms: Here, we give the asymptotic costs of the proposed TKDE algorithm and the previously proposed SIAM algorithm. The algorithms are compared only for the interest measure evaluation step corresponding to a size-(k) pattern. Even though the actual performance is a combination of the CPU time and I/O time, the strategies are most likely expected to perform similarly in terms of I/O. Given that, the size of the ST dataset

TABLE I

NOTATION FOR ALGEBRAIC COST MODEL

Notation	**Meaning**
$C_{NF}(k)$	Cost when no filtering is used.
C_{BF}	Cost while using filtering strategies
$C_{UBF}(k)$	Cost of Upper bound filtering
$C_{MST}(k)$	Cost of MST Filtering
$C(CSTP_k)$	Cost of Candidate Generation.
$C_{Prune}(CSTP_{k+1}, Grid-data)$	Cost of pruning the size k+1 candidate set during MST filtering using the coarse dataset.
$C_{Prune}(CSTP^{'}_{k+1}, data)$	Cost of pruning a reduced subset of the size k+1 candidate set using the actual dataset.
$C_{Prune}(CSTP_{k+1}, data)$	Cost of pruning the size k+1 candidate set using the actual dataset.
$C_{Prune}(CSTP^{''}_{k+1}, data)$	Cost of pruning a reduced subset of a subset of the size k+1 candidate set using the actual dataset.
$C_{Prune}(CSTP^{'}_{k+1}, Grid-data)$	Cost of pruning a reduced subset of the size k+1 candidate set using the coarse dataset.
$C_{upper}(CSTP_k)$	Cost of candidate generation during upper bound filtering.

corresponding to two size-(k-1) pattern instances is N. Since, SIAM compares every pair of size-(k-1) instances to generate the size-(k) pattern instances, this incurs a cost of $O(N^2)$ for each pattern. While evaluating interest measures for candidate CSTPs, the TKDE algorithm scans the entire set of instances from two size-(k-1) patterns and then assigns them to partitions. A plane sweep join is performed within each partition so as to determine the size-k instances for the CSTP. Hence, the cost of TKDE includes scanning the entire dataset and the cost of plane sweep per partition, which is asymptotically $O(N + \frac{N \cdot\ P \cdot\ S_\theta \cdot\ T_\theta}{S_U \cdot\ T_U} \cdot\ \log\ (\frac{N \cdot\ P \cdot\ S_\theta \cdot\ T_\theta}{S_U \cdot\ T_U}))$, where, S_θ represents the spatial neighborhood threshold and S_U, the spatial universe of the dataset. The number of partitions P is determined using a procedure similar to the one for spatial joins[20].

2) Analysis of Filtering strategies: All versions of the CSTPM make use of two filtering strategies, namely the UB filter and the MST filter. The analysis of filtering strategies aims to compare different design decisions for the kth iteration of the CSTPM. We evaluate two key design decisions : (a) no filtering prior to the actual pruning phase (b) Filtering with both UB and

MST filters. The key goal is to analytically show that filtering can enhance computational savings. The computational costs of candidate design decisions are : $C_{NF}(k)$, $C_{UBF}(k)$, $C_{MST}(k)$ and C_{BF}. The cost of the kth iteration of the CSTPM without using any filter is given by Equation (1).

$$C_{NF}(k) = C(CSTP_k) + C_{Prune}(CSTP_{k+1}, data) \tag{1}$$

The cost of the kth iteration of the CSTPM using only the upper bound filtering is given by Equation (2).

$$C_{UBF}(k) = C_{upper}(CSTP_k) + C_{Prune}(CSTP^{'}_{k+1}, data) \tag{2}$$

The cost of the kth iteration of the CSTPM while using only the MST filtering is given by Equation (3).

$$C_{MST}(k) = C(CSTP_k) + C_{Prune}(CSTP_{k+1}, Grid - data) + C_{Prune}(CSTP^{'}_{k+1}, data) \tag{3}$$

The cost of the kth iteration of the CSTPM while using both the filters is given by Equation (4).

$$C_{BF}(k) = C_{upper}(CSTP_k) + C_{Prune}(CSTP^{'}_{k+1}, Grid - data) + C_{Prune}(CSTP^{''}_{k+1}, data) \tag{4}$$

UB filtering is applied before candidate generation. Hence, the computational cost of calculating the upper bound for a candidate pattern is linear in the size of the pattern. Even though we incur this cost during upper bound filtering, the upper bound filter would be effective in preventing the merging of large numbers of size k patterns to generate non-prevalent size k+1 patterns. However, the filtering ability still depends on how tight the upper bound value is with respect to the actual measure value. Hence the cost of candidate generation using the upper bound filter would be as high as the candidate generation without the upper bound filter. For the sake of simplicity we choose to ignore the effect of traversing the pattern in the cost model. Based on this, we obtain Equation (5),

$$C_{upper}(CSTP_k) \leq C(CSTP_k) \tag{5}$$

From the definition of $CSTP_{k+1}, CSTP^{'}_{k+1}\ and$
$CSTP^{''}_{k+1}$, we have:

$$|CSTP_{k+1}| \geq |CSTP^{'}_{k+1}| \geq |CSTP^{''}_{k+1}| \tag{6}$$

Based on Equations (1), (2), (3), (4), (5) and (6), ratios between the costs of candidate design decisions are as follows:

Cost of UB Filtering: The ratio of the computational cost using the UB filter (numerator) and using no filters is:

$$\frac{C_{UBF}(k)}{C_{NF}(k)} = \frac{C_{upper}(CSTP_k) + C_{Prune}(CSTP'_{k+1}, data)}{C(CSTP_k) + C_{Prune}(CSTP_{k+1}, data)} \leq 1 \tag{7}$$

The value of the ratio represented by Equation (7) is ≤ 1 because of Equation 5. Since the pruning step has to deal with lesser candidates, the value of $C_{Prune}(CSTP'_{k+1}, data) \leq C_{Prune}(CSTP_{k+1}, data)$. Hence, analytically the CSTPM performs better with the UB filter than without any filters with a worst case equality.

Cost of MST Filtering: The ratio of the computational costs using the MST filter and using no filters is given as:

$$\frac{C_{MST}(k)}{C_{NF}(k)} = \frac{C(CSTP_k) + C_{Prune}(CSTP_{k+1}, Grid - data) + C_{Prune}(CSTP'_{k+1}, data)}{C(CSTP_k) + C_{Prune}(CSTP_{k+1}, data)} \leq 1 \tag{8}$$

$C_{MST}(k)$ and $C_{NF}(k)$ have identical candidate generation costs, $C(CSTP_k)$. Hence the effect of this can be ignored. The dominant costs are $C_{Prune}(CSTP_{k+1}, Grid{-}data)$, $C_{Prune}(CSTP'_{k+1},\ data)$ and $C_{Prune}(CSTP_{k+1},\ data)$. If the ST data distribution is higly skewed, then the number of non-empty grids would be lower and the size of the grid-data would be much lower compared to the original dataset. Hence, in $C_{MST}(k)$, $C_{Prune}(CSTP'_{k+1}, data)$ would dominate over the cost of computing the coarse interest measure for a CSTP. According to Equation 6, the MST filtering strategy filters a large number of candidates. For this reason, we have:

$$C_{Prune}(CSTP'_{k+1}, data) \leq C_{Prune}(CSTP_{k+1}, data) \tag{9}$$

This means that $C_{MST}(k) \leq C_{NF}(k)$ which verifies Equation 8.

Comparison between using both filters and no filters: The ratio of the cost of CSTPM using both filters and no filters is given as:

$$\frac{C_{BF}(k)}{C_{NF}(k)} = \frac{C_{upper}(CSTP_k) + C_{Prune}(CSTP'_{k+1}, Grid - data) + C_{Prune}(CSTP''_{k+1}, data)}{C(CSTP_k) + C_{Prune}(CSTP_{k+1}, data)} \leq 1 \tag{10}$$

In Equation (10), the dominant cost is due to pruning on the original dataset. Thus using both filters with CSTPM reduces the size of the candidate set two times, once during candidate generation through use of the UB filter, and the second time during MST filtering. The use of two filters represents a more efficient design than using no filters. Thus, we have:

$$C_{Prune}(CSTP''_{k+1}, data) \leq C_{Prune}(CSTP_{k+1}, data), \tag{11}$$

verifying Equation (10).

IV. Experimental Evaluation

The goal of the experimental evaluation was: (1) to compare the performance of the two different CSTPM algorithms: SIAM and TKDE algorithms without filtering for size-1 CSTPs, and, (2) to determine the impact of filtering on computational performance. The newly proposed TDKE algorithm was used for the filtering impact experiments. The experiments were performed using both synthetic and real world datasets in order to expand and generalize to new results beyond what was done in our recent work [8]. We used synthetically generated datasets and real world crime datasets from Lincoln, Nebraska. Figure 9 shows the experiment design with different input parameters.

The CSTPM was implemented in Matlab Release 7.9. The experiments were performed on a

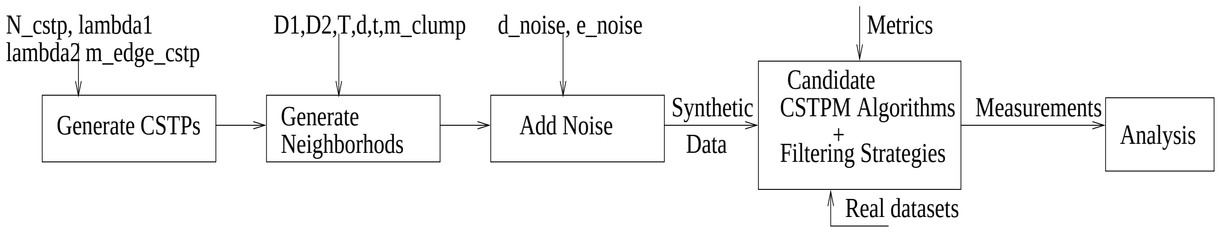


Fig. 9. Experimental setup

quad core Intel Xeon X5355 3.2 GHZ Linux Workstation with 32GB of main memory. The goal of experimental analysis was to evaluate the performance of different candidates in terms of their execution time which includes both the CPU time and I/O time. The two CSTPM algorithms, namely, the TKDE and SIAM are related to the CPI evaluation bottleneck. The TKDE algorithm with and without filtering is related to the exponential cardinality of the candidate pattern space. In order to evaluate the computational savings made in these designs we restrict our focus to the following questions:

- What is the effect of dataset size ?

- What is the effect of the number of event types ?
- What is the effect of using the two filters versus not using them?
- What is the effect of varying a clumpiness parameter? The clumpiness parameter controls the level of clustering (or positive ST autocorrelation) that corresponds to the number of instances of each event type in a unit ST neighborhood. It is similar to the intensity function in spatial statistics.

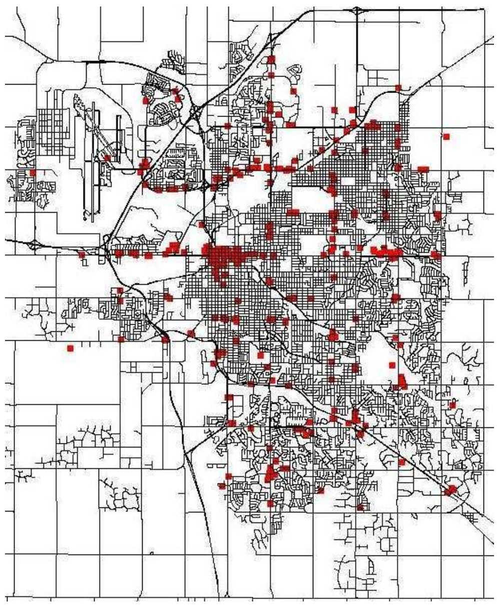
  We evaluate the effect of the first two parameters, dataset size and number of event types, using both synthetic and real datasets. The clumpiness degree can be evaluated only under a controlled condition using synthetic datasets.

A. Datasets

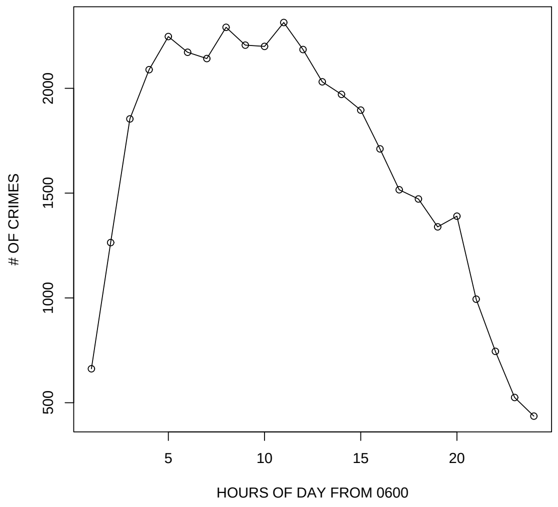
1) Synthetic dataset generation: To evaluated the performance of different algorithms, ST datasets were generated based on the spatial dataset generator proposed for spatial data mining [5]. Figure 9 shows the data flow of the synthetic ST data generation process. The process starts with the generation of core CSTPs based on the parameter N_{CSTP}. The size of each core CSTP is sampled from a Poisson distribution with a mean λ_1. Then a set of event type labels for this core CSTP was randomly chosen. The parameter λ_1 serves to control the number of event types in the ST dataset. For each of the core CSTPs, the parameter *m_edge_cascade* controls the number of CSTPs that share the same core patterns. This parameter allows us to vary the amount of overlap between different patterns that would be generated by the different CSTPM algorithms. The number of instances correspoding to each CSTP was sampled from another Poisson distribution with mean λ_2. Next the set of proximity neighborhoods for each CSTP was generated using parameters d and t. CSTP instances are themselves embedded in $d \times d \times t$ proximity neighborhoods within a $D_1 \times D_2 \times T$ spatial framework. Since ST datasets are naturally self-correlated due to the presence of positive ST autocorrelation, the next step involves clustering multiple instances of each event type within the same proximity neighborhood. This is controlled using the parameter m_{clump}, which determines the degree of clustering or clumpiness in the ST dataset. The final step involves adding noise to the ST dataset using data noise parameters, d_{noise}, e_{noise}. The noise parameters are represented as a percentage of the total number of ST instances and the number of event types respectively.

2) Real Datasets: The experimental evaluation makes use of real crime datasets from Lincoln, Nebraska [17]. The dataset contains 41 crime event types (e.g. vandalism, assaults, larceny etc.)

and approximately 39,000 crime incidents per year. For the purpose of experimental evaluation, subsets of the entire dataset were drawn. Figure 10(a) shows the location of bars on a city map, and Figure 10(b) shows the average crime count per hour in the city in the year 2007. The hours are measured from 0600 (6am) to 0500 (5am) to begin and end at the lowest crime counts. One can clearly notice an anomalous spike in activity around 20(1 AM, when the bars close) indicating a possible CSTP. The data analysis is discussed in section IV.D.



(a) Bars in Lincoln, NE



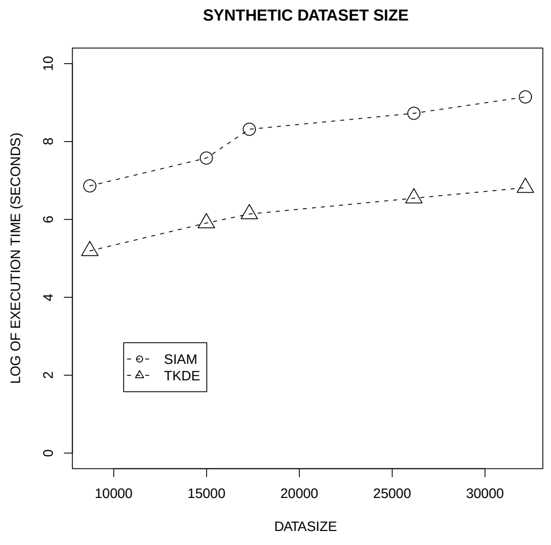
(b) Hourly crime averages in 2007 for Lincoln, NE

Fig. 10. Real ST Crime dataset(a) and (b)

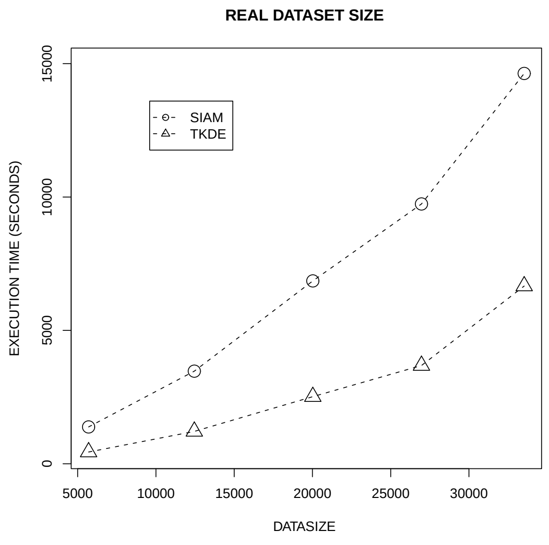
B. Evaluation of different CSTPM designs

The two different designs of the cascading spatio-temporal pattern miner (CSTPM), namely, the TKDE and SIAM were compared are compared with respect to dataset size, number of pattern instances, and clumpiness degree in the ST dataset. The CPI threshold was held constant at 0.5 for synthetic datasets and 0.15 for real datasets. The spatial and temporal neighborhood sizes were set at 15.53 miles and 1.04 days respectively for synthetic datasets; 0.31 miles and 10 days for the real crime dataset.

1) Effect of dataset size: Dataset size determines the number of instances corresponding to any CSTP. Increasing the dataset size increases the average number of instances that need to be joined by the join algorithms. The synthetic datasets were generated by setting the m_{clump} parameter to 10 and the *m_edge_cascade* parameter to 4. The Poisson distribution parameter λ_2



(a)

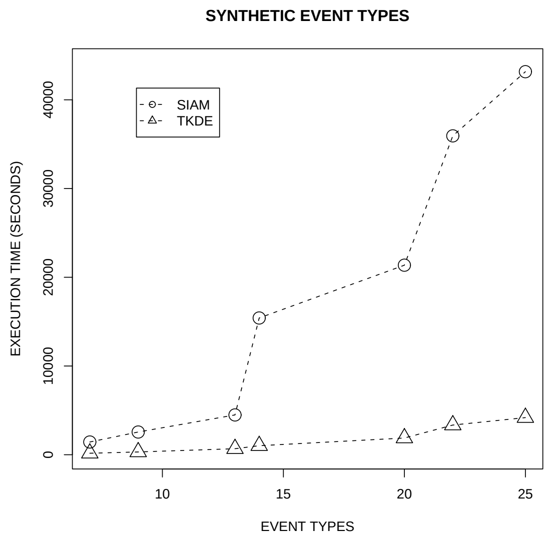


(b)

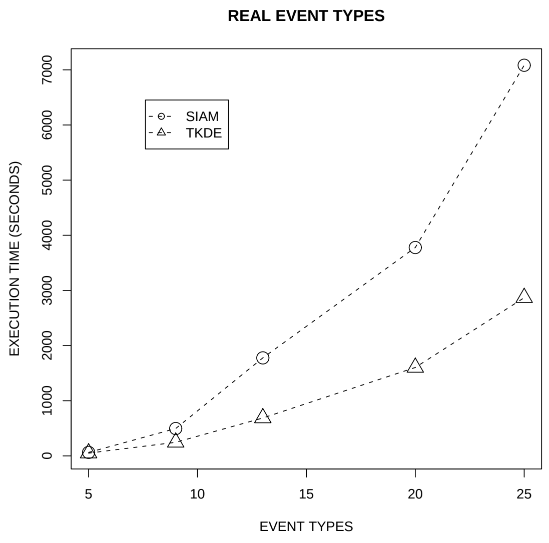
Fig. 11. Effect of dataset size on performance of SIAM and TKDE for size-1 CSTPs(a) and (b)

was varied to generate different sized datasets. The other parameters were held constant to keep the number of event-types steady. Synthetic datasets with sizes ranging from 5000 to 26000 instances were generated. Figure 11(a) shows the performance of the two algorithms on a $\log$ scale for simplicity of representation. The TKDE outperforms the SIAM by a factor of 5 to 10 on synthetic datasets, with the difference growing as the dataset size grows. Figure 11(b) shows the performance of the two join strategies on the real dataset. Here datasets were drawn from 2 months up to 12 months, creating datasets ranging in size from 5000 to 33000 instances. The TKDE outperforms the SIAM again, by a factor of approximately 2 to 3; the factor seems to drop slightly as dataset size increases.

2) Effect of Number of Event types: The event types control the number of CSTPs that are generated by a CSTPM algorithm. The number of CSTPs generated controls the number of CPI evaluations performed by the CSTPM. Hence, the goal of examining the effect of the number of event types on different CSTPM algorithms is to understand the overall scalability obtained when presented with a particular collection of event types. The effect of number of event types was evaluated on both synthetic and real datasets. In case of synthetic datasets, the m_{clump} parameter was set to 10 and the *m_edge_cascade* was set to 4. The number of event types was varied by controlling the size of the core CSTP. The Poisson distribution parameter λ_1 was varied in order to obtain datasets with varying numbers of event types. The number of event types was varied from 5 to 25 on both synthetic and real datasets. In the case of synthetic data, the TKDE outperforms the SIAM by a factor of 10 as shown in Figure 12(a). In the case of real data, TKDE outperforms SIAM by a factor of 2.5 as shown in Figure 12(b).



(a)



(b)

Fig. 12. Effect of No. of event types on performance of SIAM and TKDE for size-1 CSTPs(a) and (b)

3) Effect of Clumpiness degree: The clumpiness degree parameter, defined earlier in this section, controls the number of instances corresponding to each event type in a given ST proximity neighborhood, and can only be varied on a synthetic dataset. While evaluating the CPI, clumpiness affects the number of instances that satisfy a directed ST neighborhood relation. The clumpiness degree is varied with the m_{clump} parameter, which can vary between 4 (fewest instances) and 25 (most instances). For analyzing the effect of clumpiness degree on the performance ST join strategies, spatial and temporal neighborhood size were set to 15.53 miles and 1.04 days respectively. The CPI threshold was fixed at 0.5. Figure 15(a) shows the performance of the TKDE and the SIAM with varying clumpiness degrees on the synthetic dataset. Both filters were used to reduce computation time. The TKDE outperforms the SIAM by a factor of 2 (at clumpiness 4) to 10 (at clumpiness 25).

C. Evaluation of filtering strategies

The primary goal of filtering non-prevalent candidates early is to avoid having to compute the interest measure unnecessarily. The upper bound filter and MST filter act before and after candidate generation respectively. The combined effect of the two filters is helpful in enhancing computational savings. We evaluated the TKDE using no filters and the TKDE using both filters with real and synthetic datasets. For the synthetic datasets, the threshold was set at 0.65, and for real datasets, the threshold was set at 0.15. The spatial and temporal neighborhood sizes were set to 43.5 miles and 3 days respectively for synthetic datasets, and to 0.435 miles and 10 days respectively for the real crime datasets.

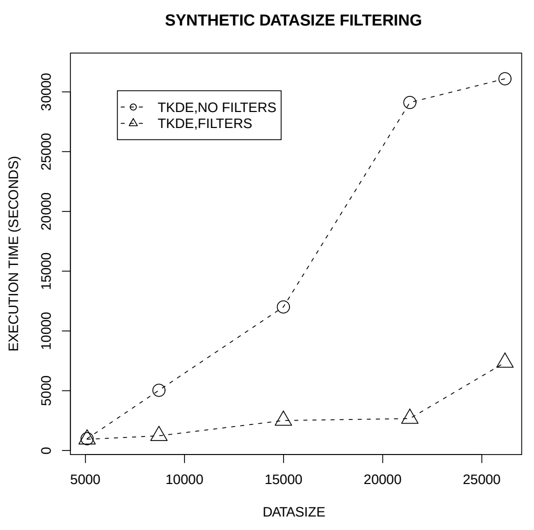
1) Effect of dataset size: Dataset size determines the number of instances for each candidate pattern. When the number of candidates is high and most candidates would have interest measure values just below the user specified threshold, the performance of the CSTPM is likely to degrade. Under these conditions, filtering helps in early pruning of these candidates and avoids unnecessary CPI evaluation, enhancing computational savings. Figures 13(a) and 13(b) quantify the computational savings obtained by making use of the filtering strategies. The synthetic datasets were generated by setting the m_{clump} parameter to 10 and the *m_edge_cascade* parameter to 4. The Poisson distribution parameter λ_2 was varied to generate different sized datasets. The other parameters were held constant to keep the number of event types steady.

Results from the experiments on synthetic datasets show a indicate that the filtering strategies enhance the computational savings up to a factor of 5 for the larger data sets, with a more modest savings for smaller datasets (in actual ratio) as shown in Figure 13(a). Results on real datasets show modest savings of up to a factor of 1.5 as shown in Figure 13(b).

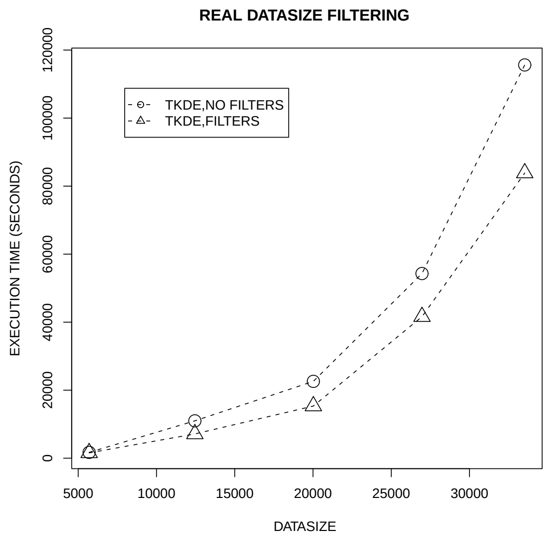
2) Effect of number of event types: The number of candidate CSTPs that could possibly be generated by the CSTPM algorithms is exponential in the number of event types. Preventing non-prevalent candidate generation as well as filtering them before expensive join computation enhances computational savings. For the synthetic datasets, the m_{clump} parameter was set to 10 and the *m_edge_cascade* was set to 4. The number of event types was varied by controlling the size of the core CSTP. The Poisson distribution parameter λ_1 was varied in order to obtain datasets with varying numbers of event types.

Figure 14(a) shows that filtering improves computational performance up to a factor of 2.5 for 25 event types on a synthetic dataset. Figure 14(b)shows computational savings up to a factor of 1.3 with the filters on a real dataset.

3) Effect of clumpiness degree: This parameter summarizes the effect of ST neighborhood thresholds on the performance of the filtering strategies. The filtering strategies benefit from

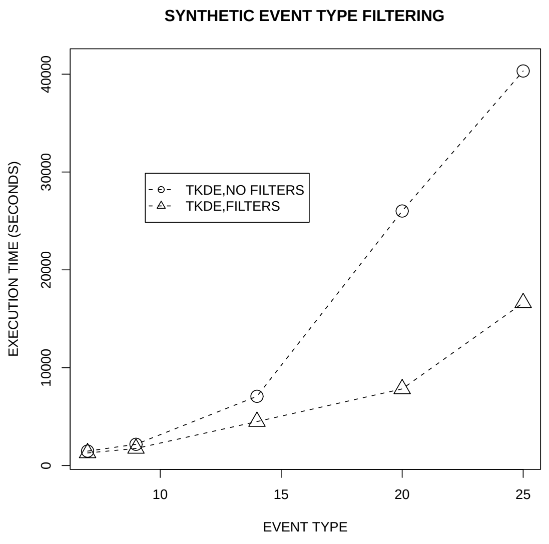


(a)

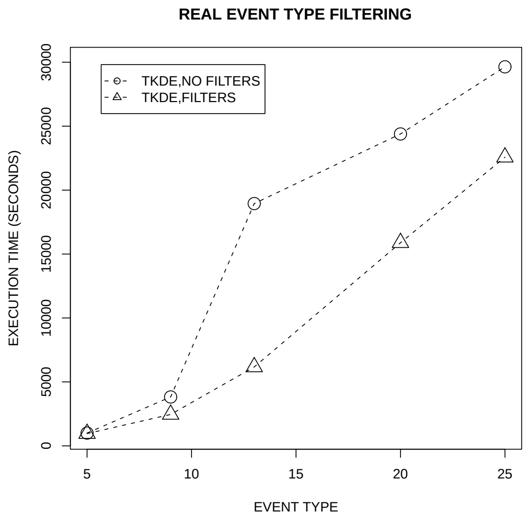


(b)

Fig. 13. Effect of dataset size on performance of Filtering strategies(a) and (b)



(a)



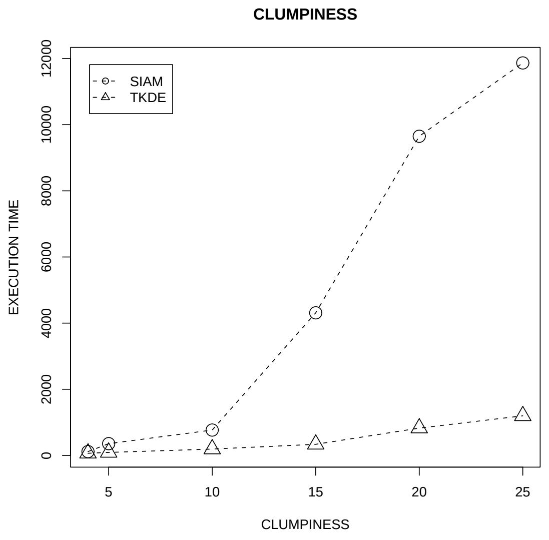
(b)

Fig. 14. Effect of No. of event types on performance of Filtering strategies(a) and (b)

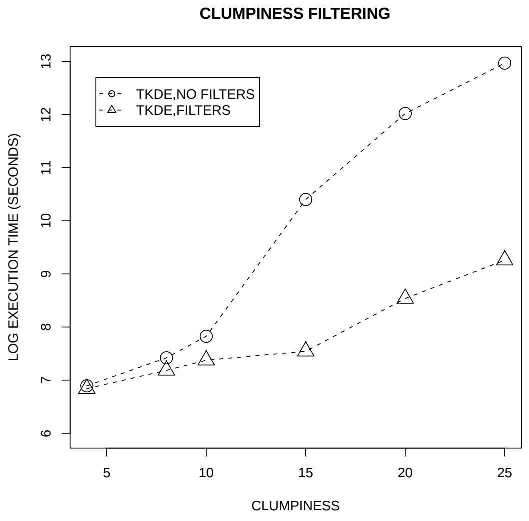
a higher clumpiness degree (i.e. a higher level of positive ST autocorrelation). This benefit is confirmed by Figure 15(b) showing that the design decision with filtering improves performance up to a factor of 40 for clumpiness of 25. This plot is a log plot.

D. *Real dataset case study*

Our case study analyzes the crime data from Lincoln, Nebraska described in Section IV.A.2. Figure 16(a) shows two CSTPs obtained from analyzing the Lincoln crime dataset.

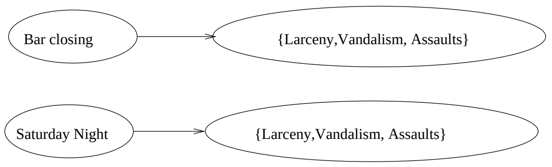


(a) Performance of ST Join strategies with respect to Clumpiness degree

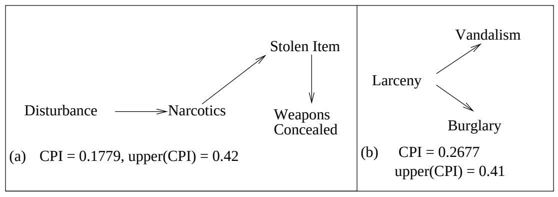


(b) Performance of filtering with respect to Clumpiness degree

Fig. 15. Effect of clumpiness degree (a) and (b)



(a) CSTPs from primary data analysis

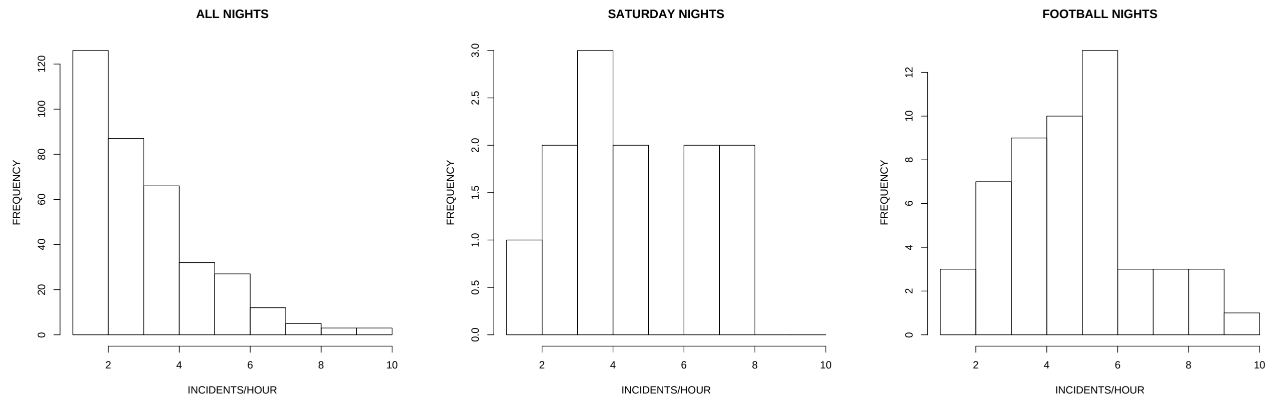


(b) CSTPs from secondary data analysis*(spatial neighborhood = 0.435 miles and temporal neighborhood = 10 days)*

Fig. 16. CSTPs from real dataset (a) and (b)

1) Primary data analysis: We considered whether the Saturday night and football CSTP generators identified are statistically significant compared to ordinary nights. Our analysis revealed that football games and bar closings do indeed generate crime-related CSTPs. Football games are normally held on Saturdays, and bars in Lincoln close around 1 AM. We observed that bar closings on these nights are associated with increased crime activity such as larceny, vandalism and assaults. We compared the number of crimes per hour around bar closing time for Saturdays and football game nights with crimes at bar closing for the entire year. Figure 17 shows histograms of the frequency of crime activity in the three hours surrounding bar closing for three cases: all year, on Saturday nights, and on football nights. The Saturday night and football night histograms seem to show a significant difference in frequency from the all year histogram, with higher frequencies of greater crime inicident values for both. We noted that all three histograms are relatively non-Gaussian, so further analysis is best served by nonparametric tests.

To test the significance of this pattern, we performed the Kolmogorov-Smirnov (KS) non-parameteric test for equality of two statistical distributions [18]. The null hypothesis under the KS test states that the two statistical samples being compared are from the same underlying population. The rejection of the null hypothesis implies that the two samples are from different populations, and thus that Saturday night bar closing and home football games are significant generators of CSTPs. Table II shows the results of the KS test comparing the candidate distributions (crime activity around bar closing on all nights, on Saturday nights, and after football games). As shown by Table II, the KS test rejected the null hypothesis (and inferred a significant difference between populations) when comparing Saturday nights with all nights at a significance level of 0.05; the p-value for this comparison is less than .000001, which is highly significant. The comparison of football nights with all nights is not significant at the .05 level (the data set is small, which makes a significant test far more difficult to achieve). However, at a higher threshold (for example, a CSTP miner detection level of 0.2) the (football game $\rightarrow$ higher crime rate) CSTP is detected. From the last row of Table 1, the Saturday night and football night populations are not signficantly different from each other.



(a) Crime Activity: All year(2007) (b) Crime Activity: Saturday nights(2007) (c) Crime Activity: Football nights(2007)

Fig. 17. Distributions of number of crimes/hour around bar closing with different generators

2) Secondary data analysis: A secondary data analysis by applying the CSTP algorithm revealed many CSTPs; two such CSTPs with CPI values, 0.17 and 0.26 are presented in Figure 16(b). These two CSTPs seem to summarize most of the information relating several additional crime types including larceny, vandalism etc. Hence, making use of the CSTP miner

TABLE II

SIGNIFICANCE OF CSTP GENERATORS

Population I	Population II	KSTAT	P-value	Significance ($\alpha = 0.05$)	CSTP Threshold ($\alpha = 0.2$)
Saturday night	All year	0.4187	$1.2498e - 07$	Yes	Yes
Football night	All year	0.3400	0.1067	No	Yes
Saturday night	Football night	0.1987	0.7899	No	No

on a real dataset provides more scope for performing a post processing on the results provided by the algorithm. Performing further significance testing on the mined patterns is possible, however, a detailed presentation of this is beyond the scope of this paper.

V. DISCUSSION

In this paper, the cascade participation index (CPI) is a lower bound on the conditional probability of a CSTP given one of its participating event types. Other alternatives to quantify interestingness have been explored in the broader data mining literature [19], [20], [21], [22].

For example, transaction based frequent pattern discovery methods for extracting sequences and graphs seek to identify a set of frequent patterns given a set of transactions from market-basket data or other graph structure transactions such as chemical compounds [19], [22], [23]. These methods use support (probability of occurrence) to denote the interestingness of a pattern. However, ST frameworks are continuous. Transactionization/partitioning of a continuous framework misses relationships between event instances at the boundary of these transactions/partitions. Transactionizing via non-disjoint partitioning may lead to double counting of overlapping relationships.

Large sparse graph mining seeks to identify frequent sub-graph patterns from a large sparse graph using computationally expensive measures such as the Maximum Independent Set (MIS) [20]. The problem of computing an MIS is NP-complete to even approximate and the best known algorithm to approximate MIS is known to have an exponential time complexity of $O(1.21^n)$, where n, is the number of nodes in the graph [24], [20]. In addition, a statistical/probabilistic interpretation of MIS has not been explored. A special case of large sparse graph mining is workflow process mining that deals with finding a minimal directed acyclic graph of a given process and a log containing many independent executions of this process [25]. This approach is not suitable for CSTP mining due to potential overlap among CSTP instances and the presence of multiple types of CSTPs in a dataset.

Time snapshot based spatial pattern families are primarily drawn from two areas: (1) moving object database patterns [26], [27], [28] and (2) dynamic graph analysis [29]. Time snapshot based spatial pattern families deal with space and time separately and perform a partitioning over time. Such a partitioning may be meaningful in a moving object database setting (e.g. the movement of the same object or a group of objects is monitored over time) or settings where changes in a network's topology over time are recorded as discrete snapshots (e.g. as in fault diagnosis in communication networks) [29]. However, in the context of ST event databases, space and time need not be separable and partitioning either one may lead to cut relationships across the partition. For example, in Figure 2(b,c,d) in Section II, the three snapshots actually represent a CSTP developing over continuous space and time as shown by Figure 2(e), the aggregated view. Moving object patterns and dynamic graph analysis treat the individual snapshots as separate instances of the same entity at different points in time. Hence, time snapshot based spatial pattern families implictly define partitions over time based on their observed datasets, and they are not designed to handle the continuous nature of space and time as observed in many ST datasets.

Models such as Bayesian networks have been used to represent a joint probability distribution of a set of variables[21]. The evaluation of joint probabilities for a single network that is computed from a database of attributes can be represented as a vector of conditionals. However, the size of this vector is exponential in the maximum in-degree of a node in a Bayesian network, making the join probability computation expensive. Given that the number of candidate Bayesian network models for a given dataset can be exponential, estimating model parameters and making inferences is exponential in the number of parameters, making exact methods intractable. Also, joint probability is similar to the support measure used in transaction graph mining as it measures the probablility of a group of variables occurring together. Hence, an interestingness measure based on joint probability may not be natural for a continuous ST framework.

VI. Conclusions and Future Work

In this paper we formally defined the problem of mining cascading spatio-temporal patterns(CSTP). CSTPs are different from ST sequences due to their topological richness and ability to account for partial order. We proposed the cascade participation index (CPI) as a lower bound on the conditional probablility of observing a pattern given that we have observed an event. The CPI is statistically meaningful as it is an upper bound to a ST statistical measure, the space-time K function. We determined that evaluating the CPI is the dominant computational step and compared different algorithms to perform this efficiently. In order to enhance computational savings by preventing unnecessary CPI evaluation, we proposed two filters based on key properties of the CPI: the existence of an anti-monotone upper bound, and positive ST autocorrelation. We showed that different CSTPM algorithms including the filtering strategies are correct and complete and reports statistically meaningful patterns. Algebraic cost models and experimental evaluation using synthetic and real datasets showed that the CSTPM making use of filtering strategies clearly enhances computational savings compared to scenarios that do not use them. Finally, we verified the applicability of the CSTP discovery framework by analyzing a real crime dataset and discovering plausible CSTPs.

A number of issues merit attention in the future work. The reported CSTPs may be considered statistically meaningful due to the statistical interpretation using the ST K-function. However, establishing statistical significance using random trials is a key task for verifying actionable patterns. Hence, a key component of future work is to perform computationally expensive significance tests by randomly permuting ST instances and accounting for multiple testing [30]. Since the MST filter is sensitive to grid cell sizes, we will explore ways to determine practical grid cell sizes in collaboration with domain scientists. In addition, we will explore new interest measures and algorithms for dealing with ST nonstationarity. Quantitative CSTPs (e.g. $A \to A$) is another direction for future work, we would explore new anti-monotone interest measures for exploring these type of patterns since CPI may not retain its antimonotonicty. In addition, we will explore alternatives to the current algorithms using ideas from query optimization [31]. This is because, a size-3 pattern (e.g. $B \to A$, $B \to C$, $A \to C$) may be computed by joining instances of either ($B \to A$, $B \to C$) and ($B \to C$, $A \to C$) or ($B \to A$, $B \to C$) and ($B \to A$, $A \to C$) or ($B \to C$, $A \to C$) and ($B \to A$, $A \to C$) as shown in Figure 5. A query optimization framework may possibly choose one of the join-paths from these choices. We will also look at incorporating semantics involving motion of ST instances, and expand our real data analysis to other applications including climate/weather datasets pertaining to natural phenomena such as the El Nino effect.

VII. Acknowledgement

We would like to thank Kim Koffolt, Nicole Wayant, Katlyn Winter, and the members of the spatial database and data mining research group at the University of Minnesota for their helpful comments. We are especially grateful to Mr. Tom Casady, Chief of Police, Lincoln City Police Department, Lincoln, Nebraska for providing us with real ST crime datasets. This work was supported in part by the US Army Corps of Engineers and the US Department of Defense.

References

[1] M. S.Scott and K. Dedel, "Assaults in and around bars(2nd edition)," *Problem Oriented Guides for Police, Problem Specific Guides*, vol. 1, pp. 1–78, 2006.

[2] *Committee on Strategic Advice on the U.S. Climate Change Science Program; National Research Council: Restructuring Federal Climate Research to Meet the Challenges of Climate Change*. Washington D.C.: The National Academies Press, 2009.

[3] D. M. Morens, G. K. Folkers, and A. S. Fauci, "The challenge of emerging and re-emerging infectious diseases," *Nature*, vol. 430, pp. 242–249, July 2004.

[4] R. Robinson, "Counting labeled acyclic digraphs," *New directions in the theory of graphs*, pp. 239–273, 1973.

[5] Y. Huang, S. Shekhar, and H. Xiong, "Discovering colocation patterns from spatial data sets: A general approach," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 12, pp. 1472–1485, 2004.

[6] J. Wang, W. Hsu, and M. L. Lee, "A framework for mining topological patterns in spatio-temporal databases," in *CIKM '05: Proc. of the 14th ACM int'l conf. on Information and knowledge management*. NY, USA: ACM, 2005, pp. 429–436.

[7] Y. Huang, L. Zhang, and P. Zhang, "A framework for mining sequential patterns from spatio-temporal event data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 4, pp. 433–448, 2008.

[8] P. Mohan, S. Shekhar, J. A. Shine, and J. P. Rogers, "Cascading spatio-temporal pattern discovery: A summary of results," in *SDM*, 2010, pp. 327–338.

[9] J. A.Shine, J. P. Rogers, S. Shekhar, and P. Mohan, "Discovering partially ordered patterns of Terrorism via Spatio-temporal Data Mining," in *16th Army conference on Applied Statistics, Cory, NC, USA*, 2010.

[10] ——, "Cascade models for spatio-temporal pattern discovery," in *1st USACE Research and Development Conference, Memphis, TN, USA*, 2009.

[11] J. F. Allen, "Towards a general theory of action and time," *Artif. Intell.*, vol. 23, no. 2, pp. 123–154, 1984.

[12] M. F. Worboys, "Event-oriented approaches to geographic phenomena," *International Journal of Geographical Information Science*, vol. 19, no. 1, pp. 1–28, 2005.

[13] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD Conference*, 1993, pp. 207–216.

[14] P.J.Diggle, A. G. Chetwynd, R. Haggkvist, and S. Morris, "Second-order analysis of space-time clustering," *Stat. Methods in Med. Res.*, vol. 4, pp. 124–136, 1995.

[15] B. Ripley, "Modelling spatial patterns," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 2, pp. 172–212, 1977.

[16] J. Patel and D. DeWitt, "Partition based spatial-merge join," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 259–270, 1996.

[17] L. C. P. Department, “Lincoln city crime records,” http://www.lincoln.ne.gov/city/police/, 2008.

[18] F. Massey Jr, “The Kolmogorov-Smirnov test for goodness of fit,” *Journal of the American Statistical Association*, pp. 68–78, 1951.

[19] M. Kuramochi and G. Karypis, “An efficient algorithm for discovering frequent subgraphs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1038–1051, Sept. 2004.

[20] ——, “Finding frequent patterns in a large sparse graph,” *Data Mining and Knowledge Discovery*, vol. 11, no. 3, pp. 243–271, 2005.

[21] J. Pearl and G. Shafer, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. JSTOR, 1988.

[22] R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements,” in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT ’96. London, UK: Springer-Verlag, 1996, pp. 3–17.

[23] J. Pei, J. Liu, K. Wang *et al.*, “Discovering frequent closed partial orders from strings,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 11, p. 1481, 2006.

[24] M. R. Garey and D. S.Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*. New York, USA: W.H. Freeman and Company, 1979.

[25] R. Agrawal, D. Gunopulos, and F. Leymann, “Mining process models from workflow logs,” in *In Sixth International Conference on Extending Database Technology*, 1998, pp. 469–483.

[26] H. Cao, N. Mamoulis, and D. Cheung, “Discovery of collocation episodes in spatiotemporal data,” in *Sixth International Conference on Data Mining, 2006. ICDM’06*, 2006, pp. 823–827.

[27] M. Celik, S. Shekhar, J. P. Rogers, and J. A. Shine, “Mixed-drove spatiotemporal co-occurrence pattern mining,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 10, pp. 1322–1335, 2008.

[28] H. Cao, N. Mamoulis, and D. Cheung, “Discovery of periodic patterns in spatiotemporal sequences,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 4, p. 453, 2007.

[29] J. Chan, J. Bailey, and C. Leckie, “Discovering correlated spatio-temporal changes in evolving graphs,” *Knowledge and Information Systems*, vol. 16, no. 1, pp. 53–96, 2008.

[30] G. Webb, “Discovering significant patterns,” *Machine Learning*, vol. 68, no. 1, pp. 1–33, 2007.

[31] M. Jarke and J. Koch, “Query optimization in database systems,” *ACM Comput. Surv.*, vol. 16, no. 2, pp. 111–152, 1984.