TR 10-001

Cascading Spatio-temporal pattern discovery: A summary of results

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

January 14, 2010

Cascading spatio-temporal pattern discovery: A summary of results[*]

Pradeep Mohan, Shashi Shekhar[†]   James A.Shine, James P.Rogers [‡]

Abstract

Given a collection of Boolean spatio-temporal(ST) event types, the cascading spatio-temporal pattern (CSTP) discovery process finds partially ordered subsets of event-types whose instances are located together and occur in stages. For example, analysis of crime datasets may reveal frequent occurrence of misdemeanors and drunk driving after bar closings on weekends and after large gatherings such as football games. Discovering CSTPs from ST datasets is important for application domains such as public safety (e.g. crime attractors and generators) and natural disaster planning(e.g. hurricanes). However, CSTP discovery is challenging for several reasons, including both the lack of computationally efficient, statistically meaningful metrics to quantify interestingness, and the large cardinality of candidate pattern sets that are exponential in the number of event types. Existing literature for ST data mining focuses on mining totally ordered sequences or unordered subsets. In contrast, this paper models CSTPs as partially ordered subsets of Boolean ST event types. We propose a new CSTP interest measure (the Cascade Participation Index) that is computationally cheap($O(n^2)$ vs. exponential, where n is the dataset size) as well as statistically meaningful. We propose a novel algorithm exploiting the ST nature of datasets and evaluate filtering strategies to quickly prune uninteresting candidates. We present a case study to find CSTPs from real crime reports and provide a statistical explanation. Experimental results indicate that the proposed multiresolution spatio-temporal(MST) filtering strategy leads to significant savings in computational costs.

1 Introduction

Given a set of boolean spatio-temporal(ST) event types and their instances, cascading spatio-temporal patterns (CSTPs) are partially ordered subsets of event types whose instances are located together and occur in successive stages. Figure 1 shows a CSTP observed after a hurricane. The first stage of the CSTP is the hurricane event and the successive stages are represented by events such as heavy rainfall, localized flooding and wind damage. Figure 2 shows an example from a crime dataset of a CSTP involving three event types: bar-closing (represented by circles), assault (represented by triangles) and drunk driving (represented by squares). Bars in large cities are often considered as generators of crimes that occur after bar closing time[25]. In crime analysis, CSTPs may represent interesting hypotheses relating several crime types, which may help law-enforcement agencies and policy makers to determine appropriate action for crime mitigation. CSTPs are important in

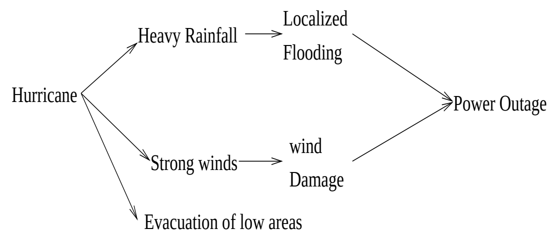


Figure 1: CSTP occurring after a Hurricane

a number of application domains, including climate change science (e.g. understanding the effects of climate change on food supply[1]), public health (e.g. tracking the emergence, spread and re-emergence of multiple infectious diseases[18]) and public safety (e.g. studying the interaction between several crime event types).

CSTP discovery is a challenging problem for two key reasons: (1) quantifying the measure of interestingness of ST patterns has complex constraints that include computational tractability (e.g. measures are computable in polynomial time) and statistical essence (e.g. statistical interpretation is based on ST statistics) and (2) the large cardinality of candidate patterns, which is exponential in the number of event types, makes the problem combinatorially complex[21].

Related Work : Related literature from ST data mining has primarily focussed on ST sequences[13] and unordered co-occurrences[27, 6]. A ST sequence represents a chain of event types in a uniform ST framework under the assumptions of total ordering because of time[13]. Co-occurrences represent un-ordered collection of event types that occur together in a uniform ST framework[27, 6]. These methods do not account for the possible existence of event instances with ei-

[*]Supported by grants from US Army and NSF

[†]Department of Computer Science, University of Minnesota, {mohan,shekhar}@cs.umn.edu

[‡]US Army ERDC, Geospatial Research and Engineering Division, Alexandria, VA {james.a.shine,james.p.rogers.II}@usace.army.mil

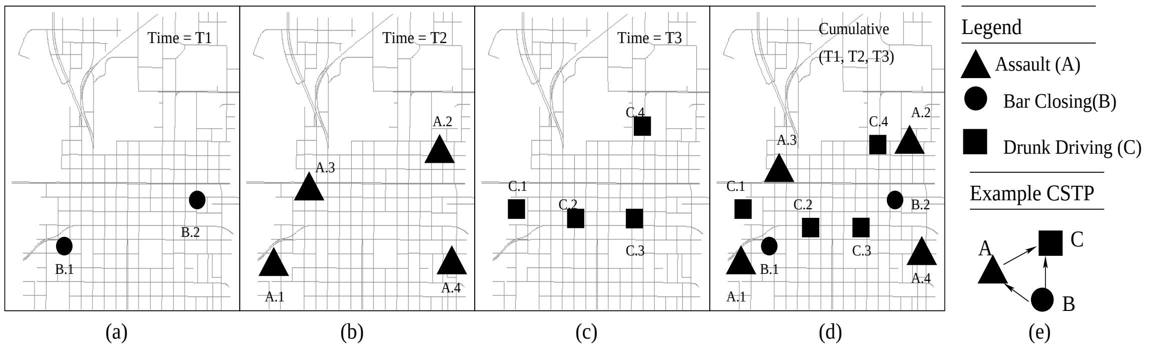


Figure 2: Illustrative Spatio-temporal Crime Dataset

ther disjoint or similar occurrence times. This limits the topological richness of ST patterns to account for notions such as events and processes that are defined in time geography and temporal logic[5, 23, 28]. Hence, existing ST data mining methods are not designed to discover ST partial ordered patterns such as CSTPs. In the broader data mining literature, possible candidates for quantifying the interestingness of CSTPs have been proposed[15, 14, 19]. Section 5 discusses their applicability in a ST data mining context.

Our Contributions: This paper models CSTPs as partial ordered ST patterns.A novel CSTP interest measure, the Cascade Participation Index(CPI) which can be evaluated in $O(n^2)$ (n being the number of instances in the database) computations is proposed. The CPI also exhibits the anti-monotone property to facilitate apriori style pruning [3]. It can be shown that the CPI is statistically meaningful by proving that it is an upper bound to the space-time K Function [20, 9]. The paper introduces a novel CSTP miner and proves that it is correct and complete. In addition to apriori style pruning and upper bound(UB) filtering, the CSTP miner uses the ST nature of the data to further reduce computational cost. In particular, it introduces a multi-resolution spatio-temporal (MST) filter.

This paper makes the following contributions: a) novel CSTP interest measure to quantify statistically meaningful CSTPs; b) an apriori based CSTP miner that uses novel filtering strategies to discover CSTPs from ST datasets; c) an analytical evaluation proving the correctness and completeness of the CSTP Miner; (d) a case study and statistical explanations to find CSTPs from real datasets; and (e) an experimental evaluation of the proposed filtering strategies using different design decisions on real crime datasets.

Scope: This paper focuses on computational aspects of discovering CSTPs. It does not address issues related to choice of directed neighborhood relationships and interest measure and ST neighborhood size thresholds. These are application domain dependent and will be addressed in future in colloboration with domain scientists.

Outline: The rest of the paper is organized as follows: Section 2 defines some basic concepts, including the boolean ST partial order neighbor relation and CSTP interest measures, and formalizes the CSTP discovery problem. Section 3 describes the CSTP Miner, the two filtering strategies (the UB filter and the MST filter) and provides an analytical evaluation of the CSTP Miner. Section 4 presents results of a case study and computational evaluations with real crime datasets. Section 5 presents a relevant discussion and Section 6 concludes the paper.

2 Problem Formulation

The special properties of ST data (e.g. dependence, overlapping neighborhoods, low dimensional embedding etc.) motivate different data models and/or representations. This section describes some basic concepts (e.g. ST data modeling and interestingness measures) and formulates the CSTP mining problem.

2.1 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 [23, 5, 28]. 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-type 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)** by restricting distance in space, time or both by using a threshold. For example, a misdemeanor crime instance and a bar-closing instance may not be considered directed neighbors if separated by six hours and tens of miles.

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 **directed neighbor relation(R)** on the illustrative dataset shown in Figure 2 produces the directed acyclic graph shown in Figure 3 called the Directed ST neighborhood graph. This graph is computed on-the-fly by the CSTP miner.

Partial (or total) order may also be defined on

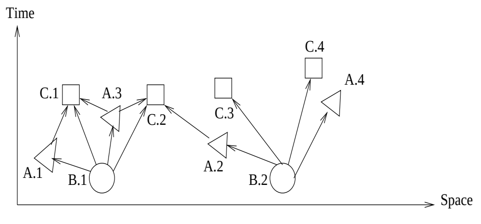


Figure 3: Directed ST neighborhood graph

event-types, possibly to represent simple processes. For example, Figure 1 shows a partial order among event types of hurricane, heavy rainfall, strong winds, evacuation of local areas, localized flooding, wind damage and power outage.

Cascading ST patterns (CSTP) in a broad sense represent partially ordered sets of event-types.

2.2 Quantifying the interestingness of CSTPs:

Interestingness of CSTPs may be measured in different ways (e.g. support, join-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 context of application domains. In a 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 that influences interest measure is the ability to predict the instances of a CSTP given an instance of a participating event type. In order to account for the unique characteristics of ST frameworks, the measures for CSTP mining are a generalization of the measures defined in spatial co-location pattern mining[22].

Definition 2.1.

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, CPT(CSTP,M) can be written as
$CPR(CSTP, M) = \frac{\#instances\ (M_j)\ participating\ in\ CSTP}{\#instances\ (M_j)\ in\ Dataset}$,
where M_j is a participating event type in CSTP with $1 \leq\ j \leq \#\ Event\ Types\ in\ the\ Dataset.$

Definition 2.2.

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 M in pattern 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 M, CPI(CSTP) may be viewed as a lower bound on the conditional probability $Pr(CSTP|M)$ for any participating event-type M. This definition holds under the assumption of ST stationarity. CPI can be formally written as, $CPI = min\{CPR(CP, M)\}$,

For example, the CPI of the CSTP shown in Figure 2 is $CPI = min\left\{\frac{1}{2}, \frac{2}{4}, \frac{2}{4}\right\} = \frac{1}{2}$.

Reliability: CPI, just like many other measures in data mining, represents a trade-off between two conflicting goals: modeling pattern interestingness (or importance), and computational scalability. CPI is computationally less expensive to evaluate for each candidate than related interest measures such as the size of maximum independent set of instances used in graph mining[15]. It is also anti-monotonic to utilize apriori-like pruning[3], which is not the case for interestingness measures based on joint probability, such as those in Bayesian Networks[19]. Interestingness depends on the application. Graph mining and Bayesian Networks measure the frequency/probability of a pattern, while CPI measures the conditional probability of a pattern(such as a CSTP) given an instance of a participating event-type. We feel that the CPI is a useful measure for applications predicting the near future occurrence of a pattern in the vicinity of an observed instance of a participating event-type .

Use of partial ordering: CPI(CSTP) makes of use of partial ordering in many ways. First, partial order is used to define directed ST neighborhood graphs to restrict the direction of influence (e.g. edges cannot start from a later event and end up at an earlier event). Second, partial order is used to define the ordering of event-types in cascading ST patterns and their instances. Thus algorithms to compute CPI(CSTP) ensure that the pattern instances being counted do not violate the partial order constraint. For example, a join-based algorithm to compute CPI(CSTP) will use

the partial order constraint as the join predicate to enumerate relevant instances of the pattern CSTP. In contrast, a graph-based algorithm to compute CPI(CSTP) may enumerate directed neighbor edges consistent with the partial order before counting subgraphs representing pattern instances.
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.
c. A threshold for the CPI
Find :
a. CSTPs with CPI $\geq$ the user specified threshold
Objective :
a. 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. Each event type is considered Boolean because we are primarily concerned with either the occurrence or absence of a crime event type at a particular location or time. The directed neighbor relation R can be defined by using distance (e.g. 0.5 miles, 5 miles etc.) or time thresholds (e.g. minutes, hours, days etc.). The CSTP discovery problem does not require the number of stages of a CSTP as an input.

3 Challenges and Solutions

In this section, we outline some key challenges of mining CSTPs and explore possible solutions to these challenges. We then describe the CSTP Miner and two novel filtering strategies: the upper bound (UB) filter and the multiresolution spatio-temporal (MST) filter).

3.1 Broad Challenges: Spatio-temporal datasets consist of many different event-types. The cardinality of candidate CSTPs is exponential in the number of event-types[21]. Since unfiltered candidate generation will generate an exponential number of potentially uninteresting candidates, smarter filters that could prevent the generation of such candidates need to be designed. ST data mining often faces the conflicting requirements of statistical correctness and computational scalability. An ability to address this complex requirement is one of the desired properties of interestingness measures. We show that the CPI addresses this complex requirement and thus serves as a useful interestingness measure for CSTPs.
ST neighborhood enumeration is a another key challenge in the CSTP mining. It can be addressed by either a neighborhood graph enumeration approach or a ST join based approach. In ST frameworks, there exist many overlapping neighborhoods. This forces candidate enumeration strategies (e.g. graph based, join based) to enumerate all combinations of ST relations between n data instances, leading to an $O(n^2)$ join computation cost. A key design strategy to reduce this cost is to avoid computing joins that may never lead to prevalent CSTPs. The CSTP miner uses the anti-monotonic[3] upper bound property of the CPI and low-dimensional embedding in the ST framework to enhance computational savings.

3.2 CSTP Miner is an algorithm to generate all CSTPs with a CPI value greater than or equal to a user specified threshold. The algorithm contains three key steps: (a) candidate generation, (b) interest measure computation and (c) pruning. Performance optimization is performed before candidate generation and before interest measure computation.

```
Algorithm : CSTP Miner
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:
Set of CSTPs with interest measure ≥ threshold.
Variables:
a. k: Pattern size (number of edges in a CSTP).
b. Optimization flags to activate upper bound(UB) and MST filters.
Method:
1. For size of patterns in (1,2...k) do
2.   If(UB is TRUE)
3.     Perform upper bound filtering.
4.   Generate candidate CSTPs of size k using CSTPs of size k-1
5.   Perform cycle checking and eliminate cycles.
6.   If(MST is TRUE)
7.     Perform MST filtering.
8.   Perform ST join and generate pattern instances.
9.   Prune CSTPs based on their prevalence.
10.  Generate prevalent CSTPs
11. End
```

Explanation of the detailed steps of the algorithm

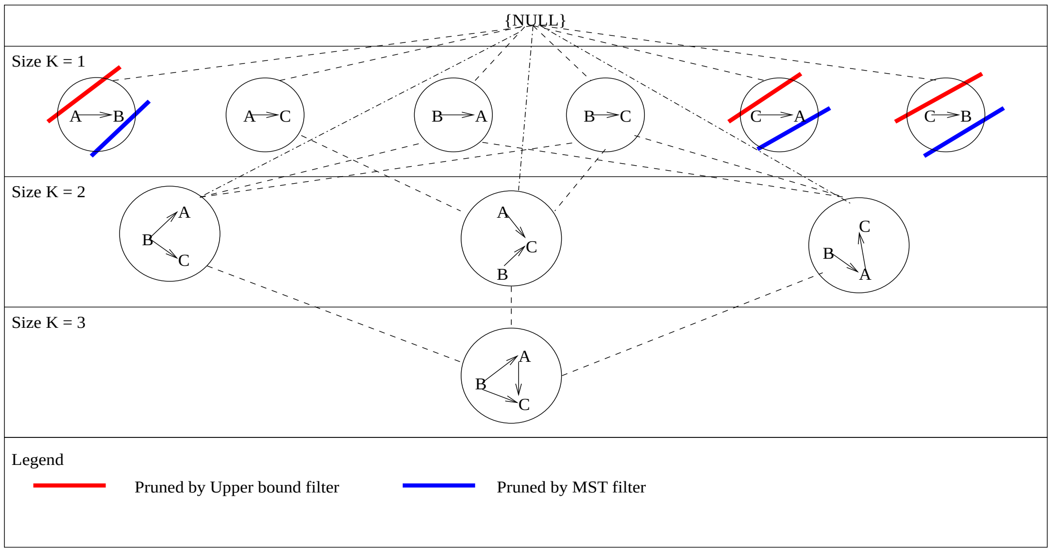


Figure 4: Candidate CSTP space (Best viewed in color)

Steps 1-11 enumerate from size-1 to size-k pattern sets and generate prevalent CSTPs. The enumeration terminates when an empty set of size-k patterns is found.
Steps 2-3 are filtering steps to avoid uninteresting candidate generation. Even though the anti-monotone upper bound in CSTP mining is different from that in graph mining[15], the intuition of avoiding uninteresting candidates by using such a filter is inspired from graph mining [15]. We describe and prove the existence of an upper bound to CPI later in this section.
Step 4 is the actual candidate CSTP computation step which generates size-k candidates from size k-1 frequent patterns. This is similar to the step used in transaction graph mining algorithms such as frequent subgraph discovery [14]. However, the key issue with this step is the generation of patterns with cycles. Cycles are problematic because CSTP is a partially ordered subset of event types represented as a directed acyclic graph. Hence, cycles need to be filtered by performing a cycle checking.
Step 5 filters out patterns that have cycles and generates the set of candidate CSTPs of size k. The cycle filtering step is extra work performed by the CSTP miner.
Steps 6-7 perform multiresolution spatio-temporal (MST) filtering. The MST filter will be described in detail later in this section.
Step 8 uses a ST join to compute the set of instances corresponding to a CSTP. The instances of a size k CSTP are computed by joining the set of instances from its size k-1 sub-patterns. The CPI is computed for the CSTP from the set of join instances.
Step 9 prunes the set of candidate CSTPs by making use of the user specified threshold. Prevalent CSTPs of size k are then generated.
The CSTP miner enumerates the space of candidate patterns as shown in Figure 4. Figure 4 shows a subset of the candidate CSTPs that are reported as prevalent patterns by the CSTP miner using a CPI threshold of 0.5. The CSTPs that are crossed using colored(red or blue) thick lines are patterns that could potentially be pruned out early using one of the pruning strategies(e.g. UB filter and MST filter). If no filtering strategies are employed, the patterns crossed with thick lines would get generated and the actual value of the CPI would be computed, slowing down the computation.

3.3 Filtering strategies :

The CSTP miner's performance is enhanced by using two filters: the upper bound (UB) filter and the multi-resolution spatio-temporal (MST) filter.

3.3.1 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 interest measure might take very low values for candidate CSTPs. 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.

Definition 3.1.

The **Upper bound of the CPI, upper(CPI)**, is the ratio of the minimum and maximum value of the CPR (see Definition 2.1) of event-types participating in a CSTP. It can be formally written as: $upper(CPI) = \frac{min\{CPR(CSTP,M_j)\}}{max\{CPR(CSTP,M_k)\}}$,

where M_j,M_k is an event type $\in CSTP$, with $1 \leq j, k \leq$ *# Event Types in Dataset*.
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 4 shows a few size k=1 patterns crossed with thick red lines and that would be pruned by the UB filter even before candidate generation. This filtering saves the cost of candidate generation and the cost of computing the ST join required for calculating the CPI.

3.3.2 Multi Resolution ST Filter: The MST filter exploits the low dimensional embedding in a ST framework. Figure 5 shows the functioning of the MST filter. A uniform ST grid defined using the parameters d and t is overlaid on the actual ST dataset. 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 grids are neighbors under R^C if and only if they contain at least one pair of instances lying on each of the grids that are neighbors under R. 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 is also computed by performing a ST join.

Figures 5b and 5c illustrate MST filtering for size 1 patterns. Figure 5c also 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 $\frac{1}{2}$, but it is 1 on the coarse dataset.

Lemma 3.1. ***MST filtering never understimates the value of the interest measures compared to the original dataset.***

Proof. MST filtering has two key ideas: (a) instance assignment to grids and (b) coarsening R to R^C. From the definition of R^C, two grids are neighbors if they contain at least one pair of instances from each grid respectively that are neighbors under R. This means that under R^C, two instances could become neighbors even if they were not neighbors under R. For example, in Figure 5(b), instances A.4 and C.4 became neigbors under R^C whereas they were not neighbors under R. This increases the number of instances of each event-type participating in any coarse ST relation, eventually increasing the interest measure value and overestimating it.

3.4 Analytical Evaluation: We show that the CSTP miner is correct, complete and can find statistically meaninful patterns. We prove that the CPI and its upper bound are anti-monotonic. Anti-montonicity is an essential property for computational efficiency of the CSTP Miner. A detailed analytical evaluation with algebraic cost models is presented in the appendix.

Theorem 3.1. ***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. The steps are described in the Appendix (Lemma A.1,A.2).

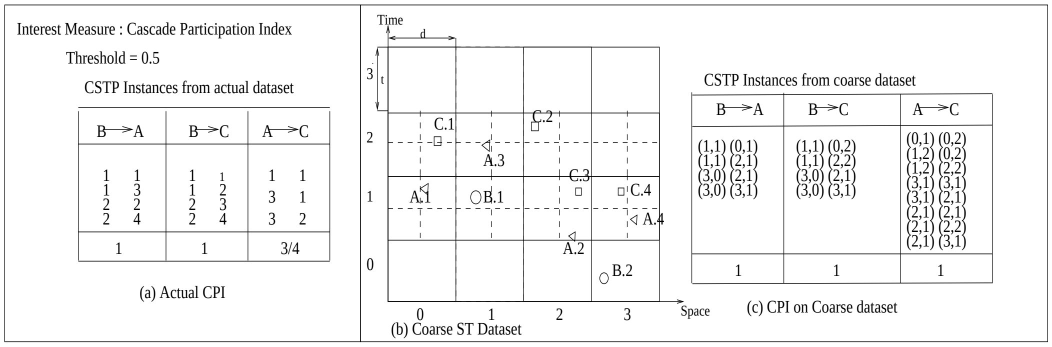
Theorem 3.2. ***The CSTP Miner is Correct.***

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

Spurious pattern generation is avoided in the CSTP miner by computing the CPI correctly and by removing candidate CSTPs with cycles. Step 8 performs a simple nested loop ST join and identifies all related instances. Hence, this step computes the CPI correctly. Step 5 removes patterns with cycles, thereby ensuring that only valid directed acyclic graphs are generated as candidate CSTPs. Only valid CSTPs that pass the user specified threshold are then accepted as prevalent patterns. Thus, the CSTP Miner is correct.

Theorem 3.3. ***The CSTP Miner is Complete.***

Proof. By completeness we mean that the CSTP miner does not miss any valid patterns and that all prevalent patterns are reported. Step 4 computes all candidate CSTPs including the ones that have cycles. Theorem 3.1 ensures that no valid patterns are pruned during


Interest Measure : Cascade Participation Index
Threshold = 0.5
CSTP Instances from actual dataset
(a) Actual CPI
Time
d
t
Space
C.1
C.2
A.3
C.3
A.1
B.1
C.4
A.4
A.2
B.2
(b) Coarse ST Dataset
CSTP Instances from coarse dataset
(c) CPI on Coarse dataset

B→A	B→C	A→C

1 1
1 3
2 2
2 4
	1 1
	1 2
	2 3
	2 4
		1 1
		3 1
		3 2

1	1	3/4

B→A	B→C	A→C
(1,1) (0,1) (1,1) (2,1) (3,0) (2,1) (3,0) (3,1)	(1,1) (0,2) (1,1) (2,2) (3,0) (2,1) (3,0) (3,1)	(0,1) (0,2) (1,2) (0,2) (1,2) (2,2) (3,1) (3,1) (3,1) (2,1) (2,1) (2,1) (2,1) (2,2) (2,1) (3,1)
1	1	1

Figure 5: Illustration of the Multi-resolution spatio-temporal filter

upper bound filtering. Lemma 3.1 guarantees that the interest measure values of patterns are overestimated correctly, and thus that a valid pattern would not get pruned. From Theorem 3.2, one can understand that the ST join phase computes the interest measure value correctly and does not make any approximations. This ensures that all patterns will have a correct value of their interest measure and no valid pattern would get pruned out because of an approximated value. Theorem 3.2 also showed that step 5 of the CSTP miner removes all cycles. Step 5 makes use of a depth first search or a breadth first search and removes patterns that are only cycles ensuring that valid CSTPs are not removed. Hence, the CSTP miner is complete.

Lemma 3.2. ***The CPI is an upper bound to the space-time K-Function***
Proof: *From Definition 2.2 and the definition of the space-time K-Function[20], we have*

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

$$K_{AB} = \frac{1}{ST}\cdot\frac{1}{\lambda_A\cdot\lambda_B}\sum\cdot_i\sum\cdot_j I_{ht}(d(A_i,B_j),t_d(A_i,B_j))$$
$$= \frac{\#\ instances(CSTP)}{|A|\cdot|B|}$$

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

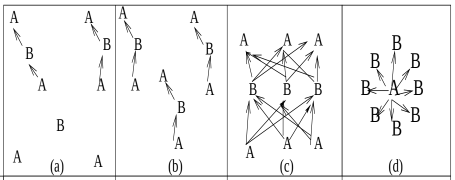
Similarly,

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

(either of the values are greater than the average number of instances of A around B and vice versa participating in CSTP)

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

For example, Figure 6 shows different cases of ST interaction between two event types A and B. In all of the shown neighborhood arrangements, the CPI is greater than or equal to the space-time K-Function.



	(a)	(b)	(c)	(d)
ST K-Function	2/18 = 1/9	1/6	18/18=1	1
CPI	2/3	1	1	1

Figure 6: The CPI as an upper bound to the space-time K-Function[20]

Figure 7 shows bars in the city of Lincoln, NE. The real crime dataset contains crime types (e.g. vandalism, assaults and larceny) and other ancillary features including bars and a football stadium.[8].

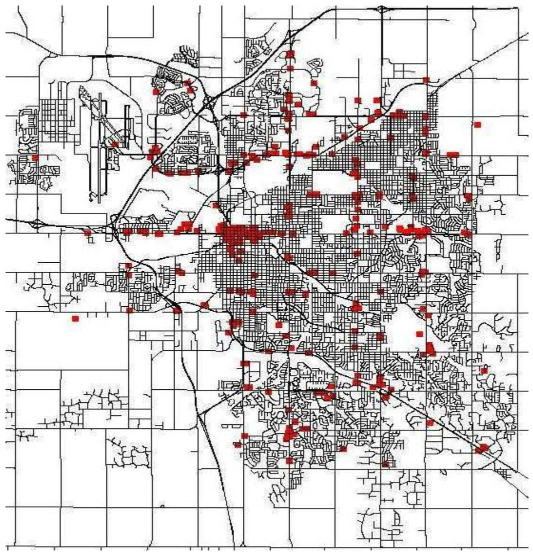


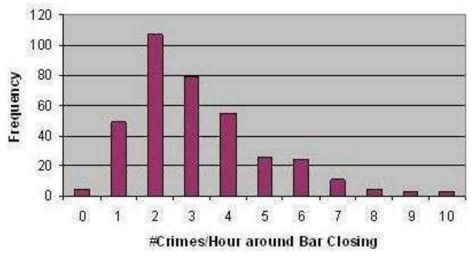
Figure 7: Bars in Lincoln, NE

4 Case Study and Performance Evaluation

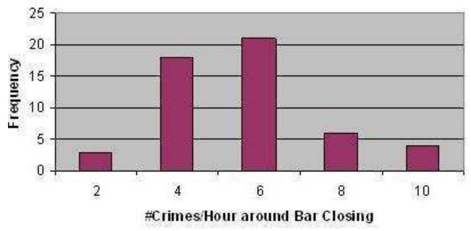
In this section, we present a case study using real crime datasets from Lincoln, Nebraska[8] and a computational performance evaluation of the CSTP miner.

4.1 Case Study and Statistical Insight:

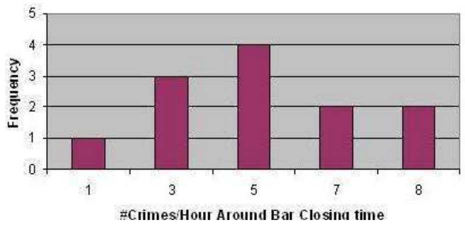
The aim of our case study was to illustrate the discovery of real CSTPs and their generators from real world datasets[8]. In the domain of public safety, events such as bar



(a) Crime Activity: All year(2007)



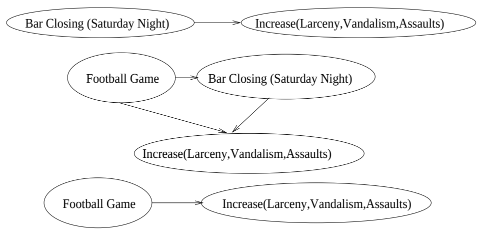
(b) Crime Activity: Saturday nights(2007)



(c) Crime Activity: Football nights(2007)

Figure 9: Distributions of number of crimes/hour around bar closing with different generators

closings and football games are considered generators of crime[25]. We analyzed crime datasets from 2007 for the city of Lincoln, Nebraska to identify real CSTPs. Our analysis revealed that football games and bar closing events do indeed generate CSTPs. Figure 8 shows three such CSTPs from the Lincoln crime dataset. We observed that bar closings on Saturday nights and


Bar Closing (Saturday Night)
Increase(Larceny,Vandalism,Assaults)
Football Game
Bar Closing (Saturday Night)
Increase(Larceny,Vandalism,Assaults)
Football Game
Increase(Larceny,Vandalism,Assaults)
Figure 8: CSTPs from real dataset

bar closings after football games are crime generators. Particularly, we observed that these events lead to an increase in the activity of crimes such as larceny, vandalism and assaults.

4.1.1 Statistical Insight:

We consider whether the CSTP generators identified are statistically significant. 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. Next we confirmed that the CSTPs and their generators were statistically meaningful using two statistical insights. First, 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 9, shows that the mean and median of the crime activity is higher(by one standard deviation) on saturdays and football nights compared to the year as a whole.

For confirmation of the significance of the generators, we performed the Kolmogorov-Smirnov (KS) non-parameteric test for equality of two statistical distributions [16]. We chose a non-parametric test because the empirical distributions of the data display distinctive non-Gaussian properties. The null hypothesis under the KS test states that the candidate statistical distributions under comparison are a part of the same continuous distribution. The rejection of the null hypothesis implies that the two distributions are different, and thus that Saturday night bar closing and home football games are significant generators of CSTPs. Table 1 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 1, the KS test rejected the null hypothesis (and inferred a significant difference between distributions) when comparing Saturday nights with all nights at a significance level of 0.05. In addition, using our CSTP thresold of 0.5 to compare post-football nights and all nights also produces the CSTP (football game → higher crime rate).

Figure 10 shows the empirical frequencies of the three groups. The KS statistic in the fourth column of Table 1 represents the maximum distance between the data frequencies. Figure 10 and the KS results strongly indicate that the saturday night barclosing and the football game bar closing are from significantly different populations than all day bar closing. However, these populations are not signficantly different from each other even though they are entirely different event types.

4.2 Performance Evaluation:

We evaluated candidate design decisions for the CSTP miner by measuring its performance with and without the proposed UB and MST filtering strategies. Figure 11 shows the input parameters used in the experiments. We compared the execution time of the CSTP Miner for four different design decisions: no filtering, UB filtering alone, MST filtering alone, UB and MST filter together. The specific experimental analysis questions addressed were: a)the effect of dataset size; b)the effect of the CPI threshold; c)the effect of the number of event types; d)the effect of the spatial neighborhood size; e)the effect of the temporal

Table 1: Significance of CSTP generators

Distribution I	Distribution II	KSTAT	P-value	Significance ($\alpha = 0.05$)	CSTP Threshold($\alpha = 0.5$)
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

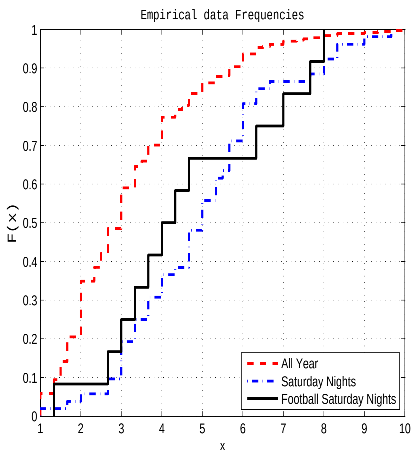
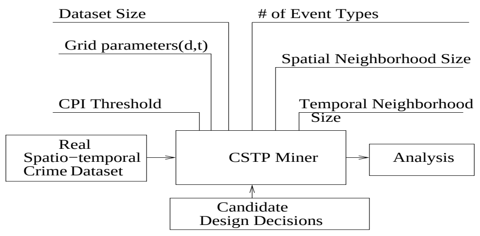


Figure 10: Cumulative data frequency comparison between different populations



Dataset Size
# of Event Types
Grid parameters(d,t)
Spatial Neighborhood Size
CPI Threshold
Temporal Neighborhood Size
Real Spatio−temporal Crime Dataset
CSTP Miner
Analysis
Candidate Design Decisions

Figure 11: Experimental Setup

neighborhood size; f) the effect of multi-resolution grid paramter 'd'; and g)the effect of multi-resolution grid parameter 't'. The CSTP Miner was implemented in Matlab Release 7. The experiments were performed on a quad core Intel Xeon X5355 2.66 GHZ Linux Workstation with 16GB of main memory.

The ST crime datasets consisted of the following fields: id, location, time, crime type and other details related to the crime. The datasets were preprocessed by assigning time stamps to every crime incident. Crime incidents that happened at exactly the same time were awarded identical timestamps. Temporal neighborhood thresholds were defined based on this assignment.

Effect of Data Size: Figure 12 shows the effect of dataset size on execution time. The UB filter creates a modest drop in execution time, while the MST filter and both filters together drop computation time by an order of magnitude.

Effect of Number of Event Types: The effect of number of event types on the computational cost of the

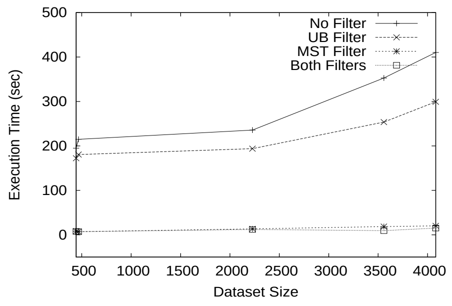


Figure 12: Effect of Dataset Size (Execution times of MST and Both Filters range from 7.9170 sec to 20.2039 sec)

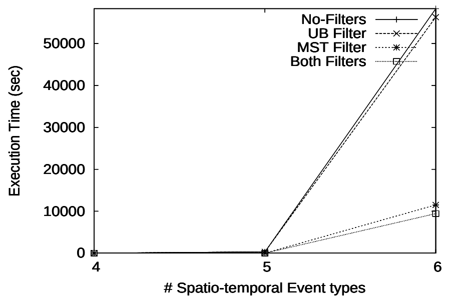


Figure 13: Effect of # event types(Execution times of MST and Both Filters range from 6.3695 sec to 11512 sec)

CSTP Miner is exponential . As illustrated in Figure 13, the execution time rises remarkably even with a small increase in the number of event types. Nevertheless, The results show a significant separation between approaches with and without MST filtering. In line with the algebraic cost model, use of both filters produces the fastest runtimes, followed closely by MST filtering alone. Upper bound filtering run times lag far behind and are only slightly better than for no filtering at all.

Effect of CPI Threshold: The effect of the CPI threshold on execution time can influence decisions to identify appropriate interest measure thresholds for different performance requirements. Figure 14 shows the effect of the CPI threshold on the execution time of the CSTP Miner for various design decisions. It can be understood from Figure 14 that, as the value of the CPI threshold increases, the execution time as well as the separation between design decisions narrows. This occurs because increases in the value of the interest measure leads to the pruning of most of the patterns. This will not only reduce the execution time, but will also reduce the gap between the candidate design decisions.

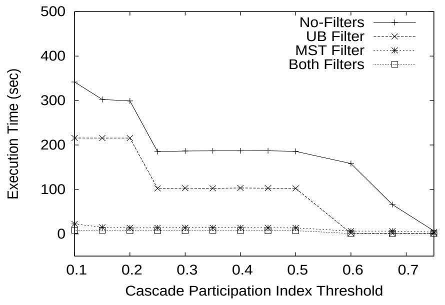


Figure 14: Effect of CPI Threshold(Execution times of MST and Both Filters range from 4.0548 sec to 22.3895 sec)

Again the upper bound filter is much less effective than the MST filter.

Effect of Spatial Neighborhood Size: A ST crime dataset of 4083 instances and 4 event types was used and the spatial neighborhood size was varied from 1 mile to 7 miles keeping the interest measure threshold at 0.2 and the time neighborhood at 1750 time stamps. Figure 15 shows that execution time is fairly constant regardless of neighborhood size. Again, MST and both filters together do much better than no filter, while UB does only modestly better.

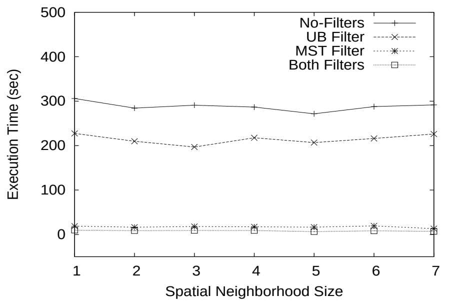


Figure 15: Effect of Spatial Neighborhood Size(Execution times of MST and Both Filters range from 7.022 sec to 19.418 sec)

Effect of Temporal Neighborhood Size: Although spatial neighborhood size did not affect execution time much, the same was not true for temporal neighborhood size. Figure 16 shows that CSTP miner's run time was sensitive to increases in temporal neighborhood size. Nevertheless, MST filtering showed superior performance for all temporal neighborhood sizes, its run time increasing linearly while the filterless run time increased exponentially.

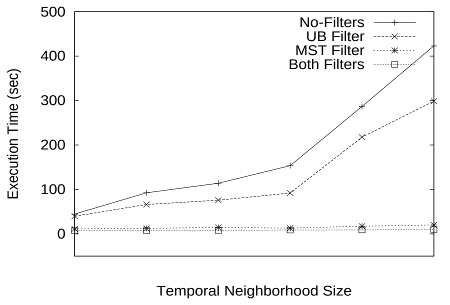


Figure 16: Effect of Temporal Neighborhood Size (Execution times of MST and Both Filters range from 7.368 sec to 20.188 sec)

Grid parameters The MST filter is sensitive to the resolution of the grid that is imposed on the original dataset. Hence, we examined the effect of varying the grid parameters on the design decisions associated with the MST filter.

Effect of Grid parameter 'd': Figure 17 shows the sensitivity of design decisions associated with the MST filter to the grid parameter 'd'. Using both the MST filter and the UB filter resulted in better performance using the MST filter alone. The multi-modal behavior of the design decisions with respect to execution time is primarily due to the fact that ST data distributions are sensitive to scale. Sensitivity of the MST filter with respect to grid size parameter 'd' was determined by keeping the grid size parameter 't' constant at a value of 2000.

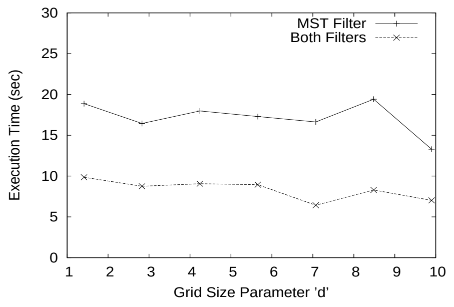


Figure 17: Sensitivity of MST filter design decisions to variation in Grid parameter 'd'

Effect of Grid parameter 't': Figure 18 shows the

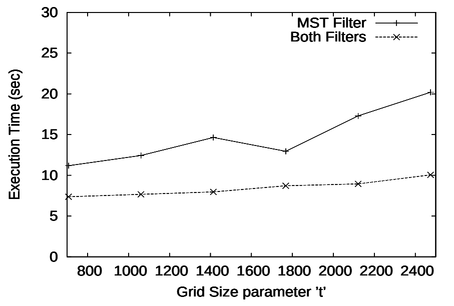


Figure 18: Sensitivity of MST filter design decisions to variation in Grid parameter 't'

sensitivity of the design decisions associated with the MST filter to the grid parameter 't'. Use of the MST and the UB filters together was less sensitive to the variation of the grid parameter 't' because of the combined pruning effect produced by both filters. However, the MST filter alone was sensitive to the variation in parameter 't'. The increases in execution time reflect the work done by the filter while computing the interest measure in the coarse dataset. Sensitivity of the MST filter with respect to the grid size parameter 't' was determined by keeping the grid size parameter 'd' constant at a value of 7.

5 Discussion

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[14, 15, 19, 24].

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[14, 24]. 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)[15]. The problem of computing an MIS is NP-complete[12, 15].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 [2]. It is not suitable for CSTP mining due to potential overlap among CSTP instances and presence of multiple cascade-types in a dataset.

Models such as Bayesian networks have been used to represent a joint probability distribution of a set of variables[19]. 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. 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 also may not be natural for a continuous ST framework. Table 2 provides a comparative summary of the computational cost and the statistical and probablistic interpretation of candidate frameworks and their interestingness measures.

6 Conclusions and Future Work

This paper modeled cascading ST patterns (CSTPs) as ST partial ordered patterns. The paper proposed a novel interest measure, a correct and complete CSTP miner and filtering strategies that were observed to enhance computational performance. The proposed measures and CSTP Miner were also proved to discover statistically meaningful CSTPs from ST datasets.

Our case study discovered a CSTP from a real data set. From our experiments, it is clear that the MST filter shows great promise in improving computational efficiency. While the UB filter did not do well in this set of experiments, we feel that it may do better in future experiments using other datasets. A detailed discussion is available in the appendix.

In future work, we would like to enhance the computational scalability of the CSTP Miner by : (a) examining different ST join strategies and (b)exploring different ST data structures for performing ST joins efficiently. We also plan to perform a rigorous experimental analysis and evaluation of parameters using synthetic datasets and evaluate alternatives to the CPI. We hope to explore several alternatives for designing new interest measures that for account aspects such as scale and ST semantics (e.g., time intervals [11, 4]). Based on ST patterns from applications such as spatial epidemiology [7], spatial economics [10] and chemical morphognesis [26], we plan to explore guidelines to identify neighborhood sizes and compare patterns with those generated by using Bayesian networks.

7 Acknowledgments

We would like to thank Kim Koffolt 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 and NSF.

References

Table 2: Comparative summary of Interestingness measures

Name	CPI	MIS[15]
Framework	CSTP	Graph Mining
Computation cost per candidate network	$O(n^2)$	$O(1.21^n)$
Statistical/Probablistic Interpretation	upper-bound(space-time-K-Function[20]) and lower-bound ($Pr(Network\|Event-type)$)	Not Explored

[1] *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.* The National Academies Press, Washington D.C., 2009.

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

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

[4] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[5] J. F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984.

[6] 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*, 20(10):1322–1335, 2008.

[7] A. Cliff, P. Haggett, J. K.Ord, and G. Versey. *Spatial Diffusion: An Historical Geography of Epidemics in an Island Community.* Cambridge University Press, Cambridge, 1981.

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

[9] E.Gabriel and P.J.Diggle. Second-order analysis of inhomogeneous spatio-temporal point process data. *Statistica Neerlandica*, 63(1):43–51, January 2009.

[10] M. Fujita, P. Krugman, and A. Venables. *The spatial economy: cities, regions and international trade.* The MIT press, 2001.

[11] A. Galton. Towards a qualitative theory of movement. In *Int. Conf. on Spatial Information Theory, LNCS 988*, pages 377–396. Springer, 1995.

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

[13] 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*, 20(4):433–448, 2008.

[14] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1038–1051, Sept. 2004.

[15] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, 11(3):243–271, 2005.

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

[17] P. Mohan, S. Shekhar, J. A. Shine, and J. P. Rogers. Cascading spatio-temporal pattern discovery: A summary of results. Technical report, Department of Computer Science and Engineering, University of Minnesota, 2010.

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

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

[20] P.J.Diggle, A. G. Chetwynd, R. Haggkvist, and S. Morris. Second-order analysis of space-time clustering. *Statistical Methods in Medical Research*, 4:124–136, 1995.

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

[22] S. Shekhar and Y. Huang. Discovering spatial colocation patterns: A summary of results. In *Lecture Notes in Computer Science*, pages 236–256, 2001.

[23] S. Shekhar and H. Xiong, editors. *Encyclopedia of GIS.* Springer, 2008.

[24] R. Srikant, R. Srikant, R. Agrawal, and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. pages 3–17, 1996.

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

[26] A. M. Turing. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London Series B*, 237:37–72, 1952.

[27] J. Wang, W. Hsu, and M. L. Lee. A framework for mining topological patterns in spatio-temporal databases. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 429–436, New York, NY, USA, 2005. ACM.

[28] M. F. Worboys. Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19(1):1–28, 2005.

Appendix

Disclaimer: This section is mainly for the convenience of the reviewers. These will be expanded from camera ready and made available as a technical report

A Theorems and Proofs

A.1 Anti-monotonicity of CPI and its upper bound

LEMMA A.1. ***CPI is Anti-monotonic.***
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 Definition 2.2, we have,*

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

$\Rightarrow CPI(CSTP(k)) \leq \ CPI(CSTP(k-1))$
CPI is Anti-Monotonic.

LEMMA A.2. ***upper(CPI) is Anti-Monotonic.***
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 Definition 3.1, we have,*
$upper(CPI(CSTP(k))) =$

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

$\Rightarrow upper(CPI(CSTP(k))) \leq \ upper(CPI(CSTP(k-1)))$

THEOREM A.1. ***The CSTP miner finds statistically meaningful ST patterns***

Proof. Lemma 3.2, showed that the CPI can find patterns that are found by the space-time K-Function. Hence, any ST correlation that exists between a pair of event types in the data is identified by the CPI. Further, the space-time K-Function cannot be extended for more than two event types and it is not anti-monotone. The CPI guarantees computational efficiency while finding statistically meaningful patterns, while the space-time K-Function does not.

B Algebraic Cost Model

The goal of the algebraic cost model is to analytically compare the computational costs of candidate design decisions for the *kth* iteration of the CSTP Miner. We evaluate four design decisions : (a) no filtering prior to the actual pruning phase; (b) upper bound filtering before candidate generation; (c) MST filtering after candidate generation; and (d) use of a combination of both filtering strategies. The key insight of the algebraic cost model is to analytically show that the MST filter and the UB filter enhance computational savings. Notation for the algebraic cost model is listed in Table 3. 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 CSTP Miner without using any filter is given by Equation (B.1).

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

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

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

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

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

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

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

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 (B.5),

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

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

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

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

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

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

The value of the ratio represented by Equation (B.7) is $\leq$ 1 because of Equation B.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 CSTP Miner performs better with the UB filter than without any filters. However, in the worst case the two costs would be equal according to Equation (B.7).

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

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

Table 3: Notation for Algebraic cost model

Notation	**Meaning**
$C_{MST}(k)$	Cost while using MST Filtering.
$C_{NF}(k)$	Cost when no filtering strategy is used.
$C_{UBF}(k)$	Cost when only the UB filtering strategy is used
C_{BF}	Cost while using both filtering strategies
$C(CSTP_k)$	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.

$C_{MST}(k)$ 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 B.6, the MST filtering strategy filters a large number of candidates. For this reason, we have:

(B.9) $C_{Prune}(CSTP^{'}_{k+1}, data) \le C_{Prune}(CSTP_{k+1}, data)$

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

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

$$\frac{C_{BF}(k)}{C_{NF}(k)} = \tag{B.10}$$

$$\frac{\begin{aligned} &C_{upper}(CSTP_k)\\ &+ C_{Prune}(CSTP^{'}_{k+1}, Grid - data)\\ &+ C_{Prune}(CSTP^{''}_{k+1}, data)\end{aligned}}{C(CSTP_k) + C_{Prune}(CSTP_{k+1}, data)} \le 1$$

In Equation (B.10), the dominant cost is due to pruning on the original dataset. Thus using both filters with CSTP Miner 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:

(B.11) $C_{Prune}(CSTP^{''}_{k+1}, data) \le C_{Prune}(CSTP_{k+1}, data)$, verifying Equation (B.10).

Comparison between the MST filter and UB filter: The ratio of the computational cost using only the MST filter and using only the UB filter is:

$$\frac{C_{MST}(k)}{C_{UBF}(k)} = \tag{B.12}$$

$$\frac{\begin{aligned} &C(CSTP_k)\\ &+ C_{Prune}(CSTP_{k+1}, Grid - data)\\ &+ C_{Prune}(CSTP^{'}_{k+1}, data)\end{aligned}}{C_{upper}(CSTP_k) + C_{Prune}(CSTP^{'}_{k+1}, data)} \le 1(Case(A))\ or \ge 1(Case(B))$$

Case (A) of Equation (B.12) happens when the data is sparse and mostly has skewed distribution; in this case the UB filter would perform at its worst according to Equation (B.7). Case (A) particularly occurs under conditions when Equations (B.9), (B.11) and (B.6) are true. Case (B) of Equation B.12 happens when the MST filter performs its worst, particularly in cases where the dataset is dense and more or less uniformly distributed. This will increase the number of non-empty grid cells and would end up increasing the size of the grid-data, leading to an increase in the cost, $C_{MST}(k)$. Finally, when we make use of both filters together, the CSTP Miner has enhanced pruning ability. Even if one of the filters is not effective, the other would be. Hence, the strategy of using both filters together would result in $C_{BF}(k) \le C_{MST}(k)$ and $C_{BF} \le C_{UBF}(k)$.

C Discussions

Comparison between CSTP Discovery and Graph Mining : Graph Mining from single large graphs[15] defines efficient algorithms for discovering frequent patterns form large sparse graphs. Graph mining makes use of a measure called the Maximum Independent Set (MIS), which is defined as the number of edge-disjoint embeddings. Figure 19, illustrates a subset of the ST instance neighborhood graph to discover a size three pattern. However, using MIS as a measure has the following limitations:

a. MIS computation is NP-Complete[12]. Hence, an exact discovery of patterns is computationally exhorbitant even for smaller sized datasets.

b. MIS computation disallows edge overlapped embeddings.

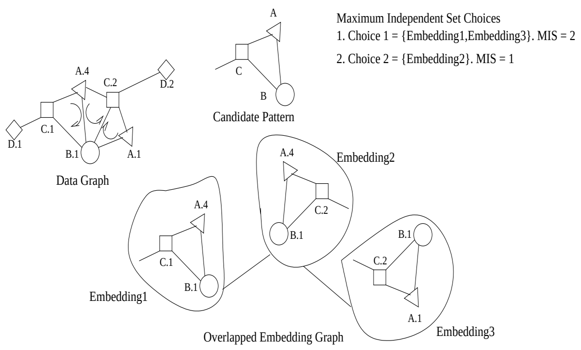

Maximum Independent Set Choices
1. Choice 1 = {Embedding1,Embedding3}. MIS = 2
2. Choice 2 = {Embedding2}. MIS = 1
A
C
B
Candidate Pattern
A.4
C.2
D.2
C.1
D.1
B.1
A.1
Data Graph
Embedding1
Embedding2
Embedding3
Overlapped Embedding Graph

Figure 19: Comparison of CSTP Discovery with Large Sparse Graph Mining

Figure 19 shows three edge overlapped embeddings of the target pattern to be examined, namely, Embedding 1, Embedding 2, Embedding 3. Graph Mining initially constructs the overlapped embedding graph and then identifies the MIS from that graph for a target pattern. To determine the MIS, an approximate greedy heuristic is used to identify as many patterns as possible above a given threshold. For example, if the threshold was set to 1, the MIS based approach may return any one of the two choices shown in Figure 19 and is not always guaranteed to return Choice 1, which is the actual value. Hence this limits the well definedness of the MIS based approach.

c. A third issue with large sparse graph mining is that it cannot take advantage of the low-dimensional embedding of ST framework to design efficient filtering strategies (e.g. MST Filter).

The computational complexity of the fastest exact solution to the MIS is , $O(1.21^N)$, where N is the number of vertices in the data graph, whereas, the computational complexity of ST join is at most $O(N^2)$, where N is the number of vertices in the ST instance neighborhood graph. This implies that the computational cost of computing CSTP measures is still lesser than MIS computation.

A Tighter CPI From, the experimental analysis results, it is quite clear that the upper bound filter is not very effective in enhancing computational efficiency. This is primarily because the upper bound for the CPI is a loose upper bound and the possible presence of noisy features that generated a large number of non-prevalent patterns. For example, Figure 20(b) shows the interaction between a single instance of event type A surrounded by many instances of event type B. It could be possible that A is a noise feature which is not accounted for by the CPI and its upper bound. Hence, the upper bound filter is not very effective. A detailed evaluation of the effect of noise using synthetic datasets is beyond the scope of the current paper. Based on our experiments and analysis, we propose a new measure called the Tight Cascade Participation Index(TCPI).

Definition C.1. ***Tight Cascade Participation Index (TCPI) :*** *The tight cascade participation index can be formally defined as,*
$TCPI = min\left\{\frac{\#instances\ (M_j)}{max\{\#instances\ (M_i)\ in\ Dataset\}}\right\}$
Where, M_j, is an event type $\in$ CSTP, with $1 \leq\ j \leq$ # Event Types in Dataset.
For example, the TCPI of the CSTP shown in Figure 2 and the neighborhood graph shown in Figure 3 is $TCPI = min\left\{\frac{1}{4}, \frac{2}{4}, \frac{2}{4}, \frac{2}{4}\right\} = 0.25$.

The TCPI is a tighter version of the CPI and lowest possible value that the CPI of a candidate CSTP can take. The rationale behind defining such a tight measure is to prune out infrequently occuring event types that may have complete participation with frequently occurring event-types. For example, Figure 20(d) shows a dataset where a single instannce of event type A is surrounded by many instances of event type B. The TCPI takes a value of $\frac{1}{5}$, whereas the CPI takes a value of 1.

The TCPI can also be used by the CSTP Miner as it

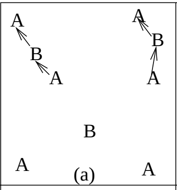
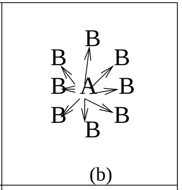
		
CPI	2/3	1
TCPI	1/3	1/5

Figure 20: Illustration of Tight CPI

also possesses the same computational properties as that of the CPI(e.g. anti-monotonicity, anti-monotone upper bound etc.) Similar to the upper bound of the CPI, the TCPI also has an upper bound which is defined as follows:

Definition C.2. ***Upper bound(TCPI) :*** *is formally defined as* $upper(TCPI) = \frac{min\{\#instances(M_j\}}{max\{\#instances(M_i)\}}$
Where, M_j, M_i, are event types $\in$ CSTP, with $1 \leq\ j, i \leq$ # Event Types in Dataset and $i \neq\ j$.
However, $max\ \{\#instances(M_i)\}$
$\leq max\ \{\#instances(M_i)\}\ in\ Dataset$
$\Rightarrow upper(TCPI) \geq\ \frac{min\{\#instances(M_j)\}}{max\{\#instances(M_i)\}\ in\ Dataset}$
$\Rightarrow upper(TCPI) \geq\ min\left\{\frac{\#instances(M_j)}{max\{\#instances(M_i)\}\ in\ Dataset}\right\}$
$\Rightarrow upper(TCPI) \geq\ RCPI$

In addition, the TCPI is an anti-monotonic interest measure which could be used by the CSTP Miner. Lemma C.1 proves that the TCPI and its upper bound are anti-monotonic.

Lemma C.1. ***TCPI is anti-monotonic.***
Proof: *By Definition C.1, we can define,* $MaxInstance =$
$$max\left\{\begin{array}{l}\#instances(M_j)\ in\ Dataset,\\ \#instances(u)\ in\ Dataset,\\ \#instances(v)\ in\ Dataset\end{array}\right\}$$
where, M_j represents event type $\in$ $CSTP(k)$ and $\in$ $CSTP(k-1)$, with $1 \leq\ j \leq$ # Event Types in Dataset.
u represents event type $\in$ $CSTP(k)$ or $\in CSTP(k-1)$
v represents event type $\in$ $CSTP(k)$ or $\in CSTP(k-1)$
and $\{e\} = (u,v) \in$ $CSTP(k)$ and $e \notin$ $CSTP(k-1)$.
$TCPI(CSTP(k)) =$
$$min\left\{\begin{array}{l}\frac{\#instances(M_j)}{MaxInstance},\\ \frac{\#instances(u)}{MaxInstance},\\ \frac{\#instances(v)}{MaxInstance}\end{array}\right\}$$
$However,\ \frac{\#instances(M_j)}{MaxInstance} \leq\ TCPI(CSTP(k-1))$
$Also, TCPI(CSTP(k)) \leq\ \frac{\#instances(M_j)}{MaxInstance}$
$\Rightarrow TCPI(CSTP(k)) \leq\ TCPI(CSTP(k-1))$
TCPI is anti-Monotonic.