MINING SPATIOTEMPORAL CO-OCCURRENCE PATTERNS FROM MASSIVE DATA SETS WITH EVOLVING REGIONS

by

Karthik Ganesan Pillai

A dissertation proposal submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

September, 2013

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

ABSTRACT

The growth of data volumes in nearly all domains of our lives is reaching historic proportions [5], [6]. The current rate of data acquisition in almost all branches of science is leading to an uncontrollable growth of data. Spatiotemporal data mining has emerged in recent decades with one of the main goals focused on developing the understanding of the spatiotemporal characteristics and patterns. This work focuses on discovering spatiotemporal co-occurrence patterns from large data sets with evolving regions. Spatiotemporal co-occurrence patterns (STCOPs) represent the subset of event types that occur together in both space and time.

Major limitations of existing spatiotemporal data mining models and techniques include the following. First, they do not take into account continuously evolving spatiotemporal events that have polygon-like representations. Second, they do not investigate and provide sufficient interest measures for these purposes. Third, computationally and storage efficient algorithms to discover STCOPs are missing. These limitations of existing approaches represent important hurdles to analyze massive spatiotemporal data sets in several application domains, including solar physics, which is an application focus of our interdisciplinary research.

In this proposal, we address these limitations by i) introducing the problem of mining STCOPs from data sets with extended (region-based) spatial representations that evolve over time, ii) developing a set of novel interest measures, and iii) providing a novel framework to model STCOPs. In this proposal, we will provide background information relevant to this work, followed by the overview of our completed research that is already published, and outline the direction of expected future research for the completion of doctoral research.

CHAPTER 1
INTRODUCTION

1.1 Context and Motivation

With the launch of NASA's Solar Dynamics Observatory (SDO) mission, solar physics researchers started dealing with "big data". SDO instruments generate approximately $70,000$ high resolution (4096×4096 pixels) images daily, obtaining one image every ten seconds [7]. SDO sends 0.55 petabytes of raster data each year [7]. This trend in solar data is anticipated to be pushed even further by ground-based Advanced Technology Solar Telescope, which is expected to capture one million images per day and generate three to five PB of data per year [8] starting from year 2015. In Fig. 1.1, we show the volume growth of solar data in recent years [2].

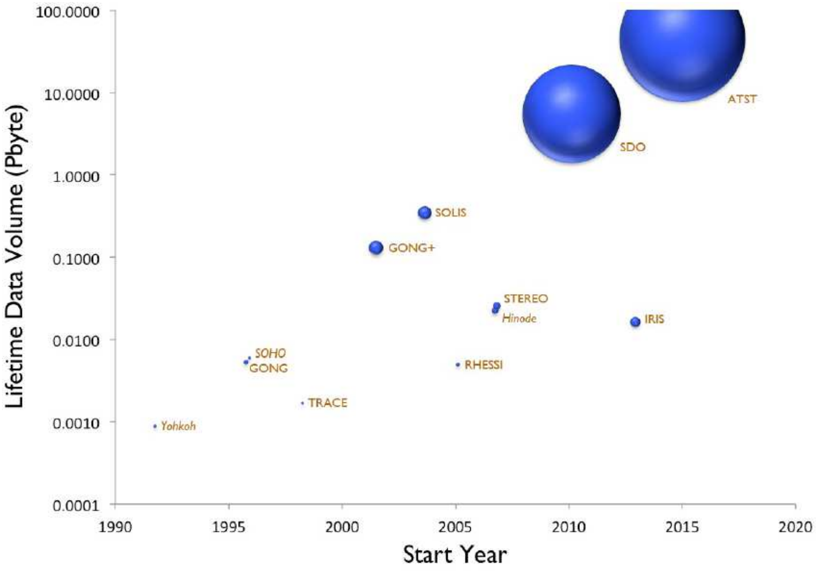


Figure 1.1: The growth of solar data volume (adapted from [2]).

To facilitate the important needs of solar activity monitoring (which can have vital impacts on space and air travel, power grids, GPS systems and communication devices [9]), many software modules are working continously on massive SDO raster data and generating object data with spatiotemporal characteristics. One motivation for our research is quantitative evaluation of solar activity, since spatiotemporal co-occurrence patterns (STCOPs) frequently occur among various solar events.

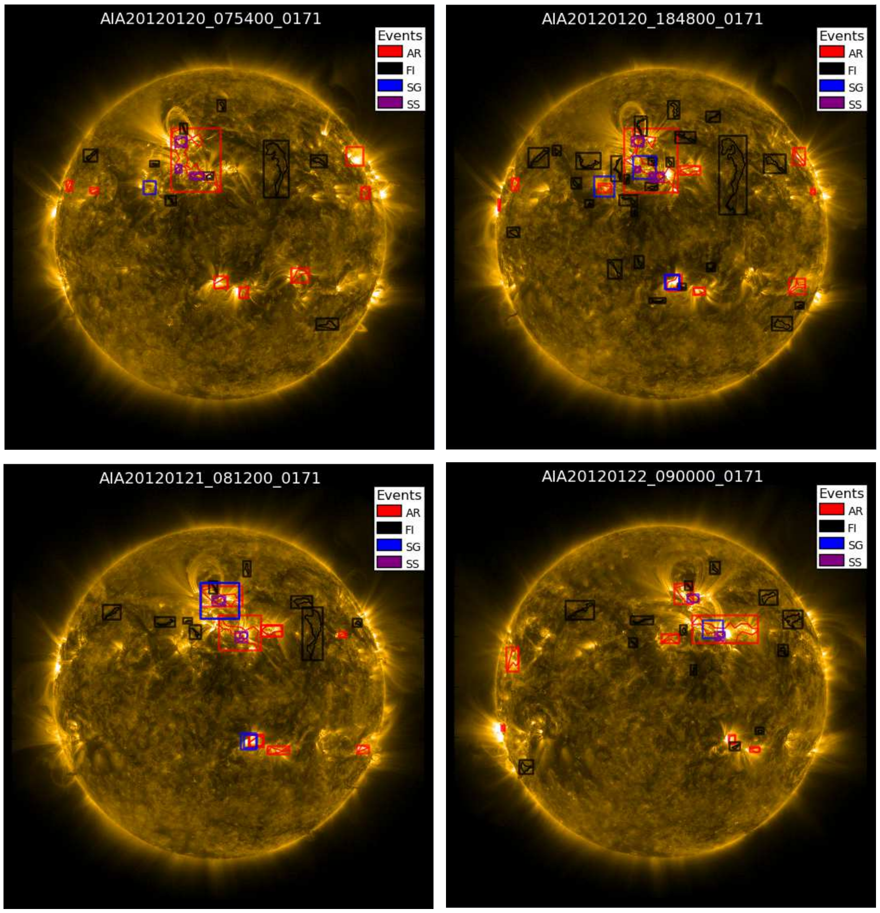


Figure 1.2: The spatiotemporal evolution of four types of solar events (NASA instrument and time stamp are printed on the top of each image).

Fig. 1.2 shows four types of solar events, Active Regions (AR), Filaments (FL), Sigmoids (SG), and Sunspots (SS) in spatial and temporal contexts with their corresponding shapes and bounding boxes. As seen in Fig. 1.2, the shapes of the solar events are represented as extended spatial representations (polygons). Moreover, the shape, size, and location of the solar events continuously evolve over time as shown in the time-series of images in Fig. 1.2. All of these factors influence relationships between various solar events, which lead to complex spatial and temporal interactions. Identifying STCOPs on the Sun could help us measure and better understand the relationships between solar events which may lead to better modeling and forecasting of important events such as coronal mass ejections and solar flares.

Goal: Given a spatiotemporal database in which data objects are represented as polygons and they continuously change their movement, shape, and size, our goal is to identify and quantitatively evaluate STCOPs representing the subset of different event types that occur together in space and time.

1.2 List of Papers Published

Here is the list of thesis related papers **Karthik Ganesan Pillai** published/awaiting publication while being a Ph.D. student at Montana State University: (1) Spatiotemporal Co-occurrence Pattern Mining in Data Sets with Evolving Regions, 2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW 2012), December 2012 [4], (2) Spatiotemporal Co-occurrence Rules, New Trends in Databases and Information Systems (ADBIS 2013), September 2013 [1], and (3) A Filter-and-Refine Approach to Mine Spatiotemporal Co-occurrences, to appear in 2013 Proceedings of the 21th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACMGIS 2013 [10].

1.3 Outline

The rest of this prospectus is organized as follows. Chapter 2 gives brief background relevant to the research work and gives a literature review of the related works. Chapter 3 introduces the research work that we completed and published already or awaiting publication. Chapter 4 presents the goals for our remaining research that needs to be completed and detailed plans to meet the goals.

CHAPTER 2
BACKGROUND

In this chapter, we briefly overview background material relevant to our work. We present important spatial data types, spatial operations, important concepts of temporal relations and spatiotemporal predicates that are neccessary to understand our work on spatiotemporal co-occurrence patterns.

2.1 Spatial Data Types

Spatial data types are data types needed to model geometry and to suitably represent geometric data in database systems and Geographical Information Systems [11]. Spatial data types provide an essential abstraction for modeling the geometric structure of objects in space, their relationships, properties, and operations [3]. Some fundamental/most popular examples of spatial data types are point, line, region; and more complex types like partitions (maps), and graphs (networks). Please see Fig. 2.1 for some examples of spatial data types [3]. Spatial data types such as regions are often referred as extended spatial data types, because they provide spatial extent of the data [3], while points and lines have only zero and one dimensional spatial extents [12].

2.2 Spatial Operations

Manipulation of spatial objects are done using *spatial operations*. *Spatial operations* take spatial objects as operands and return either scalar values (e.g. numerical

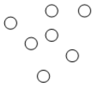
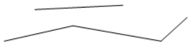
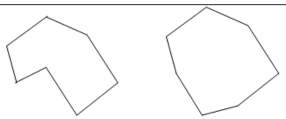
Name	Representation	Description	Example
Point		Location of object in space but not its extent relevant	City Airport Fort
Line and Line String		Movement through space, connections in space	Highway River
Region		Extent of an object relevant	City Lake Forest

Figure 2.1: Some examples of spatial data types (adapted from [3])

or Boolean values) or spatial objects. *Spatial operations* can be classified into the following three categories: (1) Spatial predicates returning Boolean values; (2) Spatial operations returning numerical values; and (3) Spatial operations that return spatial objects.

Spatial predicates returning Boolean values: A spatial relationship is a relationship between two or more spatial objects. A *spatial predicate* compares two spatial objects with respect to some spatial relationship and thus conforms to a binary relationship returning a Boolean value [3]. Spatial *Topological predicates* describe the relative position of spatial objects towards each other and are preserved under topological transformations such as translation, rotation, and scaling. Some examples of topological predicates are *equal*, *disjoint*, *intersect*, *meet* between two simple regions [13].

Spatial operations returning numerical values: One such example operation is *area* computing the corresponding value of a region object [3].

Spatial operations returning spatial objects: Some example operations include *union*, *intersection*, *difference* computing the corresponding value of spatial objects [3].

2.3 Temporal and Spatiotemporal Operations

Allen introduced interval based temporal logic in [14]. The paper also introduces six asymmetric temporal relations (*before, meets, overlaps, during, starts,* and *finishes*), and one symmetric temporal relation (*equal*). These temporal relations (all 13 of them, i.e., 6 asymmetric pairs and 1 symmetric) can be used to capture the relations between two time intervals.

However, for this work we are interested in finding spatiotemporal co-occurring patterns satisfying only a specific subset of Allen's temporal relations: *equal, meets, overlaps, during, starts,* and *finishes.* We only use one general spatial predicate: *spatial intersects* (see Fig. 2.2). Spatial intersects return true if two geometries "spatially intersect".

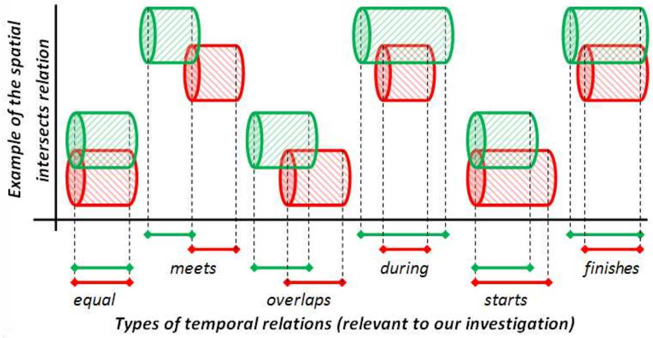

Example of the spatial intersects relation
equal
meets
overlaps
during
starts
finishes
Types of temporal relations (relevant to our investigation)

Figure 2.2: Two evolving polygons satisfying spatial intersects and temporal relations that are important for our investigation [4]

Erwig and Schneider [15] presented a convenient way of thinking about spatiotemporal predicates by applying the idea of temporal lifting and aggregation to spatial predicates. To distinguish spatiotemporal predicates from spatial predicates, following Erwig and Schneider notation, we refer to spatiotemporal predicates by using a capital letter (to begin the word) and spatial predicates by using small letters. For instance, an evolving spatial region can be represented as a three-dimensional object in three-dimensional space, where two dimensions represent spatial characteristics of the object, and the third dimension represents time. Fig. 2.2 represents some examples fulfilling the spatiotemporal relation "*Overlap*". In three-dimensional space, a moving point can be represented by a curve [15], [16] and two co-occurring polygon-like objects can be represented as types with Overlapping trajectories, where these trajectories can be represented as $3D$ spatiotemporal objects themselves.

2.4 Co-location Patterns

In classical market basket data mining, association rule mining problem is an important. Here we recall a typical notation from the literature [17]. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions (i.e., $D : \mathcal{P}(T)$), where each transaction T is a set of items such that $T \subseteq I$. An association rule is of the form $A \Rightarrow B$, where $A \subseteq I$, $B \subseteq I$, and $A \cap B = \emptyset$. $Pr(A)$ is the fraction of transactions containing A. $Pr(A \cup B)/Pr(A)$ is called the confidence of the rule and $Pr(A \cup B)$ is called support of the rule [17], [18]. An association is a subset of items whose support is above the user specified minimum support [17], [18]. Algorithms like Apriori [17] can find the frequent itemsets from all the transactions and association rules can be found from these frequent itemsets. The Apriori algorithm is a level-wise approach that follows a generate-and-test strategy to reduce the number of candidate itemsets [19].

The spatial co-location problem looks analogous to the classic association rule mining problem from market basket data mining. However, it is significantly harder than the association rule mining problem because of the lack of transactions. In market basket data sets, transactions represent sets of item types bought together by customers [18]. In the spatial co-location rule mining problem, transactions are not often explicit. The transactions in market basket analysis are independent of each other [18]. Transactions are disjoint in the sense of not sharing instances of item type. In the spatial co-location problem, the instances of spatial features are embedded in a space and share a variety of spatial relationships (e.g. neighbor) with each other [18], [20]. In this prospectus we are introducing the problem of finding spatiotemporal co-occurrence patterns, and in our problem setup, instances of spatiotemporal features (or events) are embedded in both space and time and share a variety of spatiotemporal relationships with each other.

2.5 Spatiotemporal Data Mining

Since spatiotemporal data mining is an important area, many algorithms have been proposed in the literature for co-location mining in spatiotemporal databases: Topological Pattern Mining [21], Co-location Episodes [22], Mixed Drove Co-occurrence Mining [23], Spatial Co-location Pattern Mining from extended spatial representations [24], Spatiotemporal Pattern Mining in scientific data [25], and Interval Orientation Patterns [26]. None of them; however, provide solutions for our problem of discovery of spatiotemporal co-occurrence patterns from spatiotemporal data with evolving regions. In this section we review the work for co-location pattern mining in spatial and spatiotemporal databases.

Mining topological patterns, also called co-location patterns, from spatiotemporal databases was introduced by Wang *et al.* in [21]. In this paper, the authors introduced a summary-structure to record the number of instances of a feature in a region for a given time window. The authors used the summary-structure to approximate the instance counts of a co-location pattern. The authors also introduced the TopologyMiner algorithm to discover the co-location patterns. The algorithm discovers frequent co-location patterns in a depth-first manner. The TopologyMiner algorithm divides the search space into a set of partitions, and then in each partition, it uses a set of locally frequent features to grow patterns.

There are two phases in the TopologyMiner algorithm. In the first phase, it divides the space-time dimensions into a set of disjoint cubes and builds a summary-structure that records the instance counts information of features in each cube. In the second phase, the count information stored in the summary-structure is used to find frequent co-location patterns in a depth-first manner. To measure how interesting a spatiotemporal pattern is, the authors used the participation index, first introduced in [27]. The Participation index characterizes the strength of a co-location pattern, denoted S, in implying the co-location of features. The Participation index is defined as the minimum probability among all the features of S, that is $pi(S) = min_{i=1}^{|S|} pr(f_i, S) \ s.t \ f_i \in S$, and $pr(f_i, S)$ is the participation ratio. The participation ratio $pr(f_i, S)$, measures the implication strength of a spatial feature in a co-location pattern and is defined as the fraction of the total number of instances of a feature f_i forming co-location instances in S.

Cao *et al.* introduced the problem of mining co-location episodes in spatiotemporal data [22]. In this paper, the authors define a *co-location episode* as a sequence of co-location patterns with some common feature type across consecutive time slots. The authors also introduced a two-step framework for mining co-location episodes. In the first step of the framework, the authors transform the original trajectories of moving objects to a sequence of close features to the corresponding object. Thus, object pairs of different feature types (f_i, f_j) that have close concurrent subsequences, are identified. In the next step of the framework, the authors use an Apriori-based [17] technique to discover the frequent episodes, using the transformed sequence of feature sets. To measure how interesting a co-location episode is, they use common feature types as a key factor to avoid overcounting the same instance of the common feature with different instances of other object types in the pattern.

Celik *et al.* introduced the problem of mining mixed-drove spatiotemporal co-occurrence patterns (MDCOPs) in spatiotemporal data [23]. In this paper, the authors define MDCOP as a subset of spatiotemporal mixed feature types whose instances are neighbors in space and time. They introduced the MDCOP-Miner algorithm, which extends a standard spatial co-location mining algorithm [27] to include time information. The algorithm first discovers all *size-(k)* spatial prevalent MDCOPs, and then applies a time-prevalence based filtering to discover MDCOPs. Finally, the MDCOP-Miner algorithm generates *size-(k+1)* candidate MDCOPs using *size-(k)* MDCOPs. The prevalence measure used by the MDCOP-Miner algorithm is a composition of spatial and time prevalence measures. The spatial prevalence measure is defined as the minimum participation ratio. The time prevalence measure is defined as the fraction of time slots where the pattern occurs over the total number of time slots [23].

Xiong *et al.* introduced the problem of mining spatial co-location patterns from extended spatial representations in [24]. In this paper, the authors introduced a buffer based model to find co-location patterns in data sets with extended spatial representations. In the buffer based model the neighborhood of an extended spatial representation is defined by the spatial buffer operation. The Euclidean neighborhood $N(f)$ of some feature f is defined as the union of neighborhoods for every instance i of the feature f. The Euclidean neighborhood $N(f_1, \ldots, f_k)$ for a feature set $E = \{f_1, \ldots, f_k\}$ is defined as $\cap_{i=1}^{|E|} N(f_i)$ $s.t$ $f_i \in E$. The authors introduced the coverage ratio to measure how interesting a co-location pattern is. The coverage ratio measure is defined as the ratio of the Euclidean neighborhood of feature set E, to the total area of the plane. They introduced the EXCOM algorithm to find spatial co-location patterns in data sets with extended spatial representations. The EXCOM algorithm first uses a geometric filter step to eliminate a lot of feature sets which can not form co-location patterns. In the the next step, an Apriori-based approach is used to generate spatial co-location patterns.

Mining spatiotemporal patterns in scientific data was first introduced by Yang *et al.* in [25]. In this paper, the authors introduced a general framework to discover spatial associations and spatiotemporal episodes for scientific data sets. The authors modeled features as geometric objects rather than points. They also extended their approach to accomodate temporal information and proposed an algorithm to derive spatiotemporal episodes. The authors introduced distance metrics that take into account an object's extent.

The problem of mining interval orientation patterns in spatiotemporal databases was introduced by Patel in [26]. In this work, the author modeled features by taking feature duration into account. Thus, the approach introduced was able to capture the temporal influence of a feature on other features within a spatial neighborhood. An Interval Orientation (IO) pattern is a frequent sequence of features with annotations of temporal and directional relationships between every pair of features. They introduced an algorithm called IOMiner to mine frequent IO patterns. The algorithm uses a two-stage procedure to find IO patterns. In the first stage, disjoint cubes hashing [21] is used to find IO patterns of size two. In the second stage, a hash-based join is used to find IO patterns of size three and more.

The most relevant methods to our task, which are available in the current literature, consider features as spatial point representations with temporal information

[21], [22], [23], [26] or consider features as extended spatial representations [24] with temporal information but do not take feature duration into account [25]. Thus, these methods are not adequate for mining spatiotemporal co-occurrence patterns on data sets with extended spatial representations that evolve over time.

CHAPTER 3
CURRENT CONTRIBUTIONS

In this chapter we provide details on the research we have already completed [4], [1], [10]. We first formulate the problem of mining spatiotemporal co-occurring patterns on regions that evolve over time using spatiotemporal predicates to define the evolving regions neighborhoods. Next we introduce our algorithm to mine spatiotemporal co-occurrence patterns on data sets that have evolving regions. We provide empirical results of our work. These results highlight our algorithm's contributions in terms of number of patterns generated, memory usage, execution time, and the number of rules discovered.

3.1 Spatiotemporal Co-occurrence Patterns

Given a set of spatiotemporal event types $E = \{e_1, \ldots, e_M\}$, and a set of instances of these event types, which evolve over time, $I = \{i_1, \ldots, i_N\}$ such that $M \ll N$. A spatiotemporal co-occurring pattern is a subset of spatiotemporal event types that co-occur in both space and time.

In Fig. 3.1, we show an example data set that we will use to explain the definitions in detail. In Table 3.1, we show the *Instance ID*, Event Type *Start Time*, and *End Time* of instances of different event types from our example data set in Fig. 3.1. This data set contains four event types. The event type e_1 has a total of five spatiotemporal instances ($i_1 \ldots i_5$), e_2 has three instances ($i_6 \ldots i_8$), e_3 has four instances ($i_9 \ldots i_{12}$), and e_4 has two instances ($i_{13} \ldots i_{14}$). For simplicity, in this example we do not show the sequence of $2D$ shapes that reflect the spatiotemporal evolution of our data. In

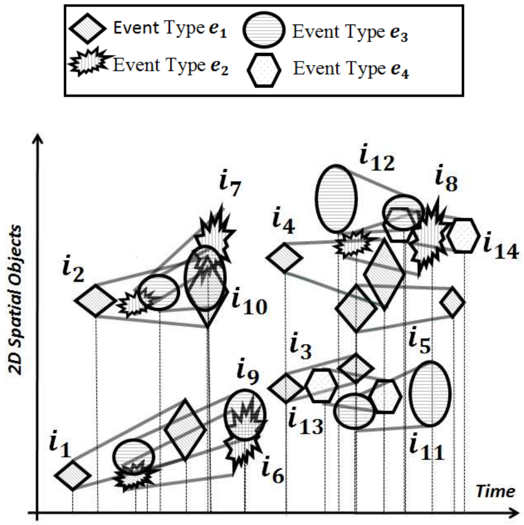


Figure 3.1: An example of spatiotemporal data

our example, $E = \{e_1, e_2, e_3, e_4\}$, $M = 4$, and $N = 14$ (all instance IDs are listed in the first column of Table 3.1).

Definition 1. A *size-(k)* spatiotemporal co-occurrence is denoted as $SE = \{e_1, \ldots, e_k\}$, where $SE \subseteq E$, $SE \neq \emptyset$ and $1 < k \leq M$.

We can have multiple *size-(k)* spatiotemporal co-occurrences derived from the set E, so to separate them we will subscript future definitions, (e.g., SE_i) with an arbitrarily chosen subscript to denote uniqueness, i.e., $SE_i \neq SE_j$. Note that indices (i or j) do not indicate the size of the co-occurrence - for the size we reserve the symbol k.

Definition 2. *pat_instance* is a *pattern instance* of a spatiotemporal co-occurrence SE_i if *pat_instance* contains an instance of all events in SE_i and no proper subset of *pat_instance* is also a *pattern instance.*

For example, $\{i_1, i_6, i_9\}$ is a *size*-3 ($k = 3$) pattern instance of co-occurrence $SE_i = \{e_1, e_2, e_3\}$ in the example spatiotemporal data set presented in Fig. 3.1 and Table 3.1.

Table 3.1: Temporal information about event instances of data shown in Fig. 3.1

Instance_ID	Event Type	Start Time	End Time
		(HH:MM)	(HH:MM)
i_1	e_1	10:00	10:30
i_2	e_1	10:10	10:40
i_3	e_1	11:00	11:20
i_4	e_1	11:00	11:30
i_5	e_1	11:20	11:50
i_6	e_2	10:20	10:50
i_7	e_2	10:20	10:40
i_8	e_2	11:20	11:40
i_9	e_3	10:20	10:50
i_{10}	e_3	10:30	10:40
i_{11}	e_3	11:20	11:40
i_{12}	e_3	11:10	11:30
i_{13}	e_4	11:10	11:30
i_{14}	e_4	11:30	12:00

Definition 3. A collection of pattern instances of SE_i is a *table instance* of SE_i, and is denoted as $tab\_ins(SE_i)$.

For example, $\{\{i_1, i_6, i_9\}, \{i_2, i_7, i_{10}\}\}$ is a *size*-3 ($k = 3$) $tab\_ins(SE_i = \{e_1, e_2, e_3\})$ in the example spatiotemporal data set presented in Fig. 3.1 and Table 3.1.

Definition 4. A spatio-temporal co-occurring rule is of the form $SE_i \Rightarrow SE_j(cce, p, cp)$, where SE_i and SE_j are spatio-temporal co-occurrences, such that $SE_i \neq SE_j$, and parameters *cce*, *p*, and *cp* characterize the rule in the following manner.

1. *cce* is an indicator of the strength of spatio-temporal relation's occurrence that is investigated (for our application we used spatio-temporal Overlap. Some

examples of spatio-temporal Overlap are $\{i_1, i_6\}$, $\{i_2, i_7\}$, and $\{i_7, i_{10}\}$ as shown in Fig. 3.1. We will discuss this more in detail in the next subsection),

2. p is the *prevalence measure*. The *Prevalence measure* emphasizes how interesting the spatio-temporal co-occurrences are. In our investigation we used the participation index (*pi*) [27] as the *prevalence measure*. The participation index monotonically decreases when the size of the spatio-temporal co-occurrence pattern increases, which can be exploited for computational efficiency [27],

3. *cp* is the conditional probability [27] of our spatio-temporal co-occurrence rule. The conditional probability gives the confidence of the spatio-temporal co-occurring rule $SE_i \Rightarrow SE_j$. In other words, the conditional probaibility indicates that whenever we observe a spatio-temporal co-occurrence of the instances of SE_i, the probability to find the instances of co-occurrences of SE_j is *cp*.

3.1.1 Measures

To calculate *cce* (in our case the strength of spatiotemporal Overlap) of a *size-(k)* spatiotemporal co-occurrence SE_i, we introduce a spatiotemporal co-occurrence co-efficient. Our spatiotemporal co-occurrence co-efficient is closely related to the Co-efficient of Areal Correspondence (CAC) proposed in [28] for spatial data analysis. CAC is computed for any two (or more, for longer patterns) overlapping polygons as the area of intersection, divided by the area of union. We extend CAC to three dimensions (two dimensions correspond to space and the third dimension corresponds to time), and calculate the spatiotemporal co-occurrence co-efficient based on volumes.

Definition 5. Spatiotemporal Intersection volume (I_v) of a *pat_instance*: The I_v for a pattern instance is the volume resulting from Intersection of trajectories of all instances of spatiotemporal event types in a pattern instance.

Definition 6. Spatiotemporal Union volume (U_v) of a *pat_instance*: The U_v for a pattern instance is the volume resulting from Union of trajectories of all instances of spatiotemporal event types in a pattern instance.

3.1.2 Co-occurrence coefficient *cce*

We use the spatiotemporal co-occurrence coefficient (*cce*) as our measure to assess the strength of the spatiotemporal relation Overlap. *cce* is calculated for a *size-k* pattern instance as the ratio $J = \frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k)}$. The symbol J represents the *Jaccard* measure [29] (see Fig. 3.2). We use *Jaccard* measure to capture the spatiotemporal co-occurrence as it is commonly accepted by data mining practitioners [29, 19]. Computing the *cce* for extended spatiotemporal representations such as evolving polygons is not a trivial task. In Fig. 3.2, we show the movement of a pair of instances of two event types (i.e., *pat_instance* of *size-2*) that change sizes and directions across different time instances. We also show the region of Intersection and the region of Union at different time slots. Moreover, the volumes resulting from the Intersection and Union trajectories of objects are shown in Fig. 3.2. If we assume that instances $\{i_1, i_6\}$, in our example data set (Fig. 3.1 and Table 3.1), have spatiotemporal Intersection volume $V(i_1 \cap i_6) = 241$ and a spatiotemporal Union volume $V(i_1 \cup i_6) = 1005$, then, the spatiotemporal co-occurrence coefficient is equal to $\frac{V(i_1 \cap i_6)}{V(i_1 \cup i_6)} = 0.23$. Please see the notes under Table 3.2 for detailed calculation of *cce*. In Table 3.2, the third column shows time instances (with Δt=10 min. used as our sampling interval), the fourth column $Area(i_1 \cap i_6)$ shows intersection areas, and the fifth column $Area(i_1 \cup i_6)$ shows union areas at each time instant.

Although, we have shown calculation of our *cce* using the *Jaccard* measure, in this work we would like to investigate alternative measures in detail. We analyze six

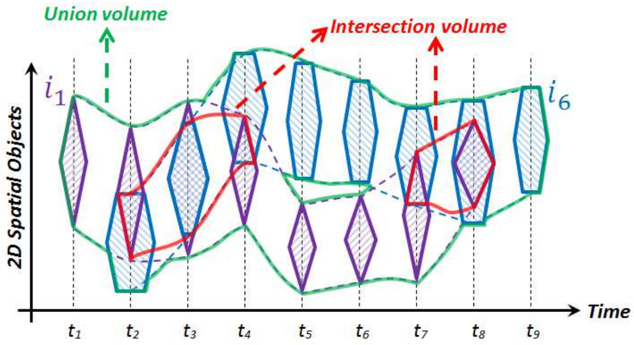


Figure 3.2: Example of size-2 co-occurrence of spatiotemporal objects.

different measures (i.e. denoted *J, OMAX, N, D, C,* and *OMIN* in the first column of Table 3.3) to assess the strength of the spatiotemporal relation Overlap. Also, the second column in Table 3.3 gives the formula for each measure for a *size-k pattern instance* where $i_1, \ldots, i_k$ denote instances of k spatiotemporal events. Moreover, each $i_j \in i_1, \ldots, i_k$ is a three dimensional geometrical object, represented by a sequence of two dimensional spatial polygons, whose shape, size, and location evolve over time. In the third column of Table 3.3 we specify whether a measure has the anti-monotone property. Each of the measures given in Table 3.3 can be used to assess the strength of spatiotemporal relation Overlap by comparing volumes of the objects in Overlapping trajectories of $i_1, \ldots, i_k$. In all the formulas, volumes are represented by the notation V. Note, the measure J shown in Table 3.3 matches the example calculation of *cce* shown earlier in this section.

3.1.3 Prevalence of STCOPs

Definition 7. The participation index $pi(SE_i)$ of a spatiotemporal co-occurrence SE_i is,

$$min_{j=1}^{k} pr(SE_i, e_j) \tag{3.1}$$

Table 3.2: Example of *tab_instance* of $SE_i = \{e_1, e_2\}$ with calculation of *cce* from data shown in Fig. 3.2.

Instance of e_1	Instance of e_2	TimeInstant($t_s = 10$ minutes)	$Area(i_1 \cap i_6)$	$Area(i_1 \cup i_6)$
i_1	i_6	t_1 =10:00	0	60
i_1	i_6	t_2 =10:10	25	120
i_1	i_6	t_3 =10:20	95	115
i_1	i_6	t_4 =10:30	15	140
i_1	i_6	t_5 =10:40	0	150
i_1	i_6	t_6 =10:50	0	140
i_1	i_6	t_7 =11:00	16	130
i_1	i_6	t_8 =11:10	90	90
i_1	i_6	t_9 =11:20	0	60

$$cce_{i_1 i_6} = \frac{V(i_1 \cap i_6)}{V(i_1 \cup i_6)} = \frac{\sum_{j=t_1}^{t_9} t_s \times Area_j(i_1 \cap i_6)}{\sum_{j=t_1}^{t_9} t_s \times Area_j(i_1 \cup i_6)} = \frac{10 \times (0+25+...+90+0)}{10 \times (60+120+...+90+60)} = \frac{241}{1005} = 0.23$$

where k is the length of the pattern (i.e., cardinality of SE_i, $|SE_i|$), and the participation ratio $pr(SE_i, e_j)$ for a spatiotemporal event type e_j is the fraction of the total number of instances of e_j forming spatiotemporal co-occurring instances in SE_i.

For example, from Fig. 3.1 and Table 3.1 we can see that, the pattern instances of spatiotemporal co-occurrence $SE_i = \{e_1, e_2, e_3\}$ are $\{\{i_1, i_6, i_9\}, \{i_2, i_7, i_{10}\}\}$. Only two (i_1, i_2) out of five instances of spatiotemporal event type e_1 participate in co-occurrence $SE_i = \{e_1, e_2, e_3\}$. So, $pr(\{e_1, e_2, e_3\}, e_1) = 2/5 = 0.4$. Similarly $pr(\{e_1, e_2, e_3\}, e_2) = 2/3 = 0.67$, and $pr(\{e_1, e_2, e_3\}, e_3) = 2/2 = 1$. Therefore the participation index of spatiotemporal co-occurrence $SE_i = \{e_1, e_2, e_3\}$ is $pi(\{e_1, e_2, e_3\}) = min(0.4, 0.67, 1) = 0.4$.

Definition 8. The spatiotemporal co-occurrence SE_i is a *prevalent pattern* if it satisfies a user-specified minimum participation index threshold.

In our example above, if the minimum threshold is set to 0.3, then the spatiotemporal co-occurrence pattern $SE_i = \{e_1, e_2, e_3\}$ is a *prevalent pattern.*

Table 3.3: Measures evaluating spatiotemporal relation Overlap (*cce*) for pattern instances of size $k \geq 2$ [1]

Name	Formula	Anti-monotone Property
Jaccard coefficient (J)	$\frac{V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{V(i_1 \cup i_2,...,i_{k-1} \cup i_k)}$	Yes
Overlap coefficient (OMAX)	$\frac{V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{\max(V(i_1),...,V(i_k))}$	Yes
Cosine coefficient (N)	$\frac{\sqrt[k]{k} \times V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{\sqrt[k]{\sum_{j=1}^{k} V(i_j)^k}}$	No
Dice coefficient (D)	$\frac{k \times V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{\sum_{j=1}^{k} V(i_j)}$	No
Cosine coefficient (C)	$\frac{V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{\sqrt[k]{V(i_1) \times V(i_2),...,V(i_{k-1}) \times V(i_k)}}$	No
Overlap coefficient (OMIN)	$\frac{V(i_1 \cap i_2,...,i_{k-1} \cap i_k)}{\min(V(i_1),...,V(i_k))}$	No

Definition 9. The conditional probability *cp*$(SE_i \Rightarrow SE_j)$ of a spatiotemporal co-occurrence rule $SE_i \Rightarrow SE_j$ is the fraction of pattern instances of SE_i that satisfies the spatiotemporal relation strength indicator *cce* to some pattern instances of SE_j. It is computed as,

$$\frac{|\pi_{SE_i}(tab\_ins(\{SE_i \cup SE_j\}))|}{|tab\_ins(\{SE_i\})|} \tag{3.2}$$

where π is the relational projection operation with duplicate elimination [27].

For example, for co-occurrence rule $e_1 \Rightarrow e_2$, from our data set in Fig. 3.1 and Table 3.1, the conditional probability is equal to

$$\frac{|\pi_{e_1}(tab\_ins(e_1 e_2))|}{|tab\_ins(e_1)|} = 3/5 = 60\%.$$

In other words, only three out of five instances (i_1, i_2, i_4) of event type e_1 co-occur with instances (i_7, i_6, and i_8) of event type e_2.

3.1.4 Problem Statement

Given:

1. A set of M spatiotemporal event types $E = \{e_1, e_2, \ldots, e_M\}$ over a common spatiotemporal framework.

2. A set of N event instances $I = \{i_1, i_2, \ldots, i_N\}$, which evolve over time such that $M \ll N$, and each $i_j \in I$ is a tuple <*instance-id, spatiotemporal event type, sequence of <2D shape, matching time instant> pairs*>, where the sequence of $2D$ shape and matching time instant pairs reflects the evolution of the spatiaotemporal event.

3. A user-specified spatiotemporal co-occurrence coefficient threshold (cce_{th}).

4. A user-specified participation index threshold (pi_{th}), which we use as our prevalence measure.

5. A user-specified conditional probability threshold (cp_{th}).

6. A time interval of data sampling (Δt). All events are sampled with the same interval making the shapes of individual events exactly aligned in time.

Objective: *Find the complete and correct result set of spatiotemporal co-occurrence patterns satisfying* $cce > cce_{th}$, $pi > pi_{th}$, *and* $cp > cp_{th}$.

3.1.5 STCOPs-Miner Algorithm

In this section we introduce our STCOPs-Miner algorithm to mine spatiotemporal co-occurrence patterns and rules from data sets with extended spatial representations that evolve over time. Fig. 3.3 gives the pseudocode of our STCOPs-Miner algorithm. In the algorithm, steps 1 and 2 initialize the parameters and data structures, steps 3 through 11 give an iterative process to mine spatiotemporal co-occurrence rules and step 12 returns a union of the results of the spatiotemporal co-occurrence patterns

(patterns of all sizes) and rules (rules of all size). Steps 3 through 11 continue until there is no candidate STCOPs to be mined. Next we explain the functions in the algorithm.

Step 2, i.e., $T_1 = gen\_loc(C_1, I, t_s)$: In this function, argument Δt represents the increment in the number of time steps. The evolution of instances of our spatiotemporal events from their start time slot is projected using Δt (to increment the number of time steps between time slots). The combination of the event instance ID and time step will allow us to identify an event at a particular moment. For example, Fig. 3.4 (a) shows the key columns of table instances of *size-1* for our sample spatiotemporal data set (Fig. 3.1 and Table 3.1). Here, the Δt value was set to 10 minutes. The column denoted t_{e_1} represents the table instance of *size-1* for event type e_1. Similarly, the columns denoted by t_{e_2}, t_{e_3}, and t_{e_4} represent the table instances of *size-1* for event types e_2, e_3, and e_4. The geometric shapes of instances are not shown in Fig. 3.4 (a) for simplicity.

Step 4, i.e., $C_{(k+1)} = gen\_candidate\_coocc(P_k)$: We generate candidate STCOPs in this step. We use an Apriori-based [17] approach to generate the candidates of size-$(k + 1)$ using spatiotemporal co-occurring prevalent patterns of *size-(k)* for anti-monotonic measures. Hence, prevalent patterns of *size-(k)*, which satisfy the user-specified threshold value of a minimum participation index pi_{th}, are used to generate candidate patterns of size-$(k + 1)$. Fig. 3.4 (b) shows the candidate co-occurrence patterns of *size-2* for our example spatiotemporal data set (Fig. 3.1 and Table 3.1). However, we generate candidate patterns of *size-(k+1)* from all *size-k* patterns for non anti-monotonic measures (see Table 3.3).

Input :

(1) E= A set of spatiotemporal event types, which can be represented as $2D$ shapes at each time step.

(2) I= <*instance-id, spatiotemporal event type, sequence of <2D shape, matching time instant> pairs*>.

(3) A user-specified spatiotemporal co-occurrence coefficient threshold (cce_{th}).

(4) A user-specified participation index threshold (pi_{th}), which we use as our prevalence measure.

(5) A user-specified conditional probability threshold (cp_{th}).

(6) A user-specified time sampling interval (Δt), measured as duration between snapshots of evolving objects.

Output :

A set of spatiotemporal co-occurrence rules with *cce*, *pi*, and *cp* greater than the user-specified minimum threshold values given on input.

Variables :

(1) k the co-occurrence size

(2) C_k: a set of candidates for size-(k) STCOPs derived from *size-($k-1$)* prevalent STCOPs

(3) T_k: set of instances of size-(k) spatiotemporal co-occurrences

(4) P_k: a set of *size-(k)* prevalent STCOPs derived from *size-(k)* candidate STCOPs

(5) P_{final}: union of all prevalent spatio-temporal co-occurring patterns (patterns of all sizes)

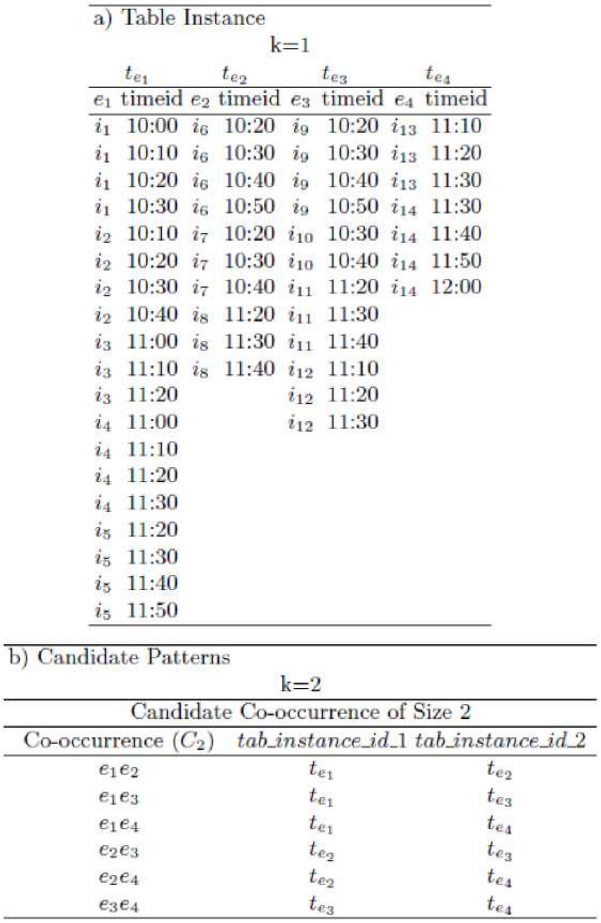
(6) R_k: a set of spatiotemporal co-occurrence rules derived from *size-(k)* prevalent STCOPs

(7) R_{final}: union of all spatiotemporal co-occurrence rules (rules of all sizes)

Algorithm :

```
$k$=1, $C_1$=$E$, $P_1 = E$, $P_{final} = \emptyset$, $R_{final} = \emptyset$;
$T_1 = gen\_loc(C_1, I, t_s)$;
while ($P_k \neq \emptyset$) {
    $C_{(k+1)} = gen\_candidate\_coocc(P_k)$;
    $T_{(k+1)} = gen\_tab\_ins\_coocc(C_{(k+1)}, cce_{th})$;
    $P_{(k+1)} = pre\_prune\_coocc(C_{(k+1)}, pi_{th})$;
    $P_{final} = P_{final} \cup P_{(k+1)}$;
    $R_{(k+1)} = gen\_rules\_coocc(P_{(k+1)}, cp_{th})$;
    $R_{final} = R_{final} \cup R_{(k+1)}$;
    $k = k + 1$;
  }
return $P_{final}$, $R_{final}$;
```

Figure 3.3: STCOPs-Miner Algorithm



a) Table Instance

k=1

t_{e_1}		t_{e_2}		t_{e_3}		t_{e_4}	
e_1	timeid	e_2	timeid	e_3	timeid	e_4	timeid
i_1	10:00	i_6	10:20	i_9	10:20	i_{13}	11:10
i_1	10:10	i_6	10:30	i_9	10:30	i_{13}	11:20
i_1	10:20	i_6	10:40	i_9	10:40	i_{13}	11:30
i_1	10:30	i_6	10:50	i_9	10:50	i_{14}	11:30
i_2	10:10	i_7	10:20	i_{10}	10:30	i_{14}	11:40
i_2	10:20	i_7	10:30	i_{10}	10:40	i_{14}	11:50
i_2	10:30	i_7	10:40	i_{11}	11:20	i_{14}	12:00
i_2	10:40	i_8	11:20	i_{11}	11:30		
i_3	11:00	i_8	11:30	i_{11}	11:40		
i_3	11:10	i_8	11:40	i_{12}	11:10		
i_3	11:20			i_{12}	11:20		
i_4	11:00			i_{12}	11:30		
i_4	11:10						
i_4	11:20						
i_4	11:30						
i_5	11:20						
i_5	11:30						
i_5	11:40						
i_5	11:50						

b) Candidate Patterns

k=2

Candidate Co-occurrence of Size 2		
Co-occurrence (C_2)	*tab_instance_id_1*	*tab_instance_id_2*
e_1e_2	t_{e_1}	t_{e_2}
e_1e_3	t_{e_1}	t_{e_3}
e_1e_4	t_{e_1}	t_{e_4}
e_2e_3	t_{e_2}	t_{e_3}
e_2e_4	t_{e_2}	t_{e_4}
e_3e_4	t_{e_3}	t_{e_4}

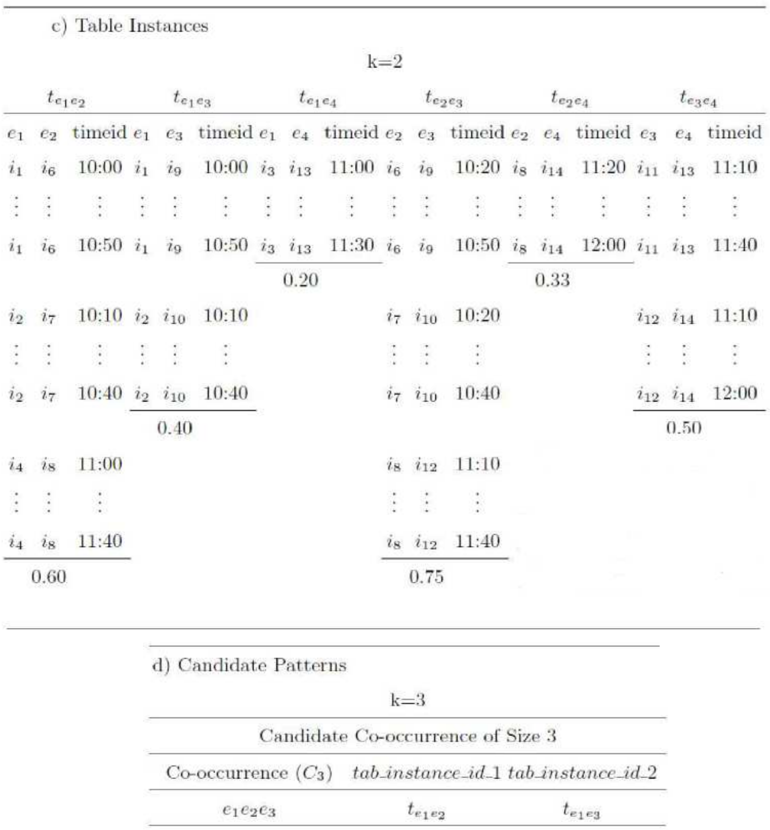
Figure 3.4: Table instances of size-1 (a) and candidate patterns of size-2 (b).

Step 5, $T_{(k+1)} = gen\_tab\_ins\_coocc(C_{(k+1)}, cce_{th})$: This function generates table instances for candidate patterns of *size-(k + 1)*. Pattern instances for each table instance can be generated by a spatiotemporal query. The geometric shapes of the instances at each time step are saved, as these geometric shapes will be used for finding the *cce* of STCOPs of size three or more. Pattern instances that have a *cce* below the user-specified cce_{th} value are deleted from the table instance, if the measure used to calculate *cce* has the anti-monotonic property (i.e., J and $OMAX$). However, for non anti-monotonic measures (see Table 3.3), pattern instances that do not have any volume resulting from intersection of trajectories of the event instances, are deleted from the table instances.

For example, Fig. 3.5 (c) shows the important columns of *size-2* table instances for our sample spatiotemporal data set from Fig. 3.1 and Table 3.1. The column denoted $t_{e_1e_2}$ represent the table instance of *size-2* co-occurrence of event types e_1 and e_2. Similarly, the other columns represents table instance of different event types. We also show the pattern instances that satisfy the threshold $cce_{th} = 0.01$. For simplicity, we just show the running example with *cce* value calculated using anti-monotonic measures. Moreover, we only show the key columns of table instances for simplicity. For example, in the table instance $t_{e_1e_2}$ shown in Fig. 3.5 (c), the rows $i_1, i_6, 10{:}00$ through $i_1, i_6, 10{:}50$ represent a pattern instance that satisfies the threshold $cce_{th} = 0.01$.

Step 6, i.e., $P_{(k+1)} = pre\_prune\_coocc(C_{(k+1)}, pi_{th})$: This function discovers filtered *size-*$(k+1)$ STCOPs by pruning $C_{(k+1)}$ that have $pi < pi_{th}$. For example, we show the *pi* value (see Def. 7) at the end of each table instance in Fig. 3.5 (c). As seen from the Fig. 3.5 (c), the patterns $SE_i = \{e_1, e_4\}$, and $SE_j = \{e_2, e_4\}$ will be pruned if a value of 0.39 is set to pi_{th}. Thus, the patterns that satisfy the $pi_{th} = 0.39$ are $\{\{e_1, e_2\}, \{e_1, e_3\}, \{e_2, e_3\}, \{e_3, e_4\}\}$. These four patterns will be used in the next iteration of the algorithm in Step 4 (see Fig. 3.5 (d)). However, please note, if the measure used to calculate *cce* is non anti-monotonic, we will not prune the candidates based on pi_{th} value. Instead, all the patterns will be used to generate *size-*$(k+1)$ patterns. In Step 7, the algorithm calculates the union of patterns P_{final} and P_{k+1}.

Step 8, i.e., $R_{(k+1)} = gen\_rules\_coocc(P_{(k+1)}, cp_{th})$: In this step we generate spatiotemporal co-occurrence rules. A set of spatiotemporal co-occurrence rules R that have *cp* greater than cp_{th} of *size-*$(k{+}1)$ is generated from $P_{(k+1)}$ [27] for anti-monotonic measures. However, for non anti-monotonic measures we generate rules that have *cp*



c) Table Instances

k=2

$t_{e_1e_2}$			$t_{e_1e_3}$			$t_{e_1e_4}$			$t_{e_2e_3}$			$t_{e_2e_4}$			$t_{e_3e_4}$		
e_1	e_2	timeid	e_1	e_3	timeid	e_1	e_4	timeid	e_2	e_3	timeid	e_2	e_4	timeid	e_3	e_4	timeid
i_1	i_6	10:00	i_1	i_9	10:00	i_3	i_{13}	11:00	i_6	i_9	10:20	i_8	i_{14}	11:20	i_{11}	i_{13}	11:10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i_1	i_6	10:50	i_1	i_9	10:50	i_3	i_{13}	11:30	i_6	i_9	10:50	i_8	i_{14}	12:00	i_{11}	i_{13}	11:40
							0.20						0.33				
i_2	i_7	10:10	i_2	i_{10}	10:10				i_7	i_{10}	10:20				i_{12}	i_{14}	11:10
⋮	⋮	⋮	⋮	⋮	⋮				⋮	⋮	⋮				⋮	⋮	⋮
i_2	i_7	10:40	i_2	i_{10}	10:40				i_7	i_{10}	10:40				i_{12}	i_{14}	12:00
				0.40												0.50	
i_4	i_8	11:00							i_8	i_{12}	11:10						
⋮	⋮	⋮							⋮	⋮	⋮						
i_4	i_8	11:40							i_8	i_{12}	11:40						
	0.60									0.75							

d) Candidate Patterns

k=3

Candidate Co-occurrence of Size 3

Co-occurrence (C_3)	$tab\_instance\_id\_1$	$tab\_instance\_id\_2$
$e_1e_2e_3$	$t_{e_1e_2}$	$t_{e_1e_3}$

Figure 3.5: Table instances of size-2 (c) and candidate patterns of size-3 (d).

greater than cp_{th} from patterns of $P_{(k+1)}$ that have pi value greater than pi_{th} (note, this check is neccessary for non anti-monotonic measures because we do not prune away patterns, see Step 6).

The conditional probability (Def. 9) $cp(SE_i \Rightarrow SE_j)$ of a spatiotemporal co-occurrence rule $SE_i \Rightarrow SE_j$ is calculated using Eq. 3.2, where π is the relational projection operation with duplicate elimination.

Step 9 i.e., $R_{final} = R_{final} \cup R_{(k+1)}$ of the algorithm calculates the union of rules R_{final} and R_{k+1}. The algorithm runs iteratively until no more STCOPs can be generated for anti-monotonic measures. However, for non anti-monotonic measures all the patterns are generated and in a post processing step only the patterns that satisfy pi_{th}

are reported. Finally the algorithm returns the union of all the found spatiotemporal co-occurrence patterns and rules in Step 12.

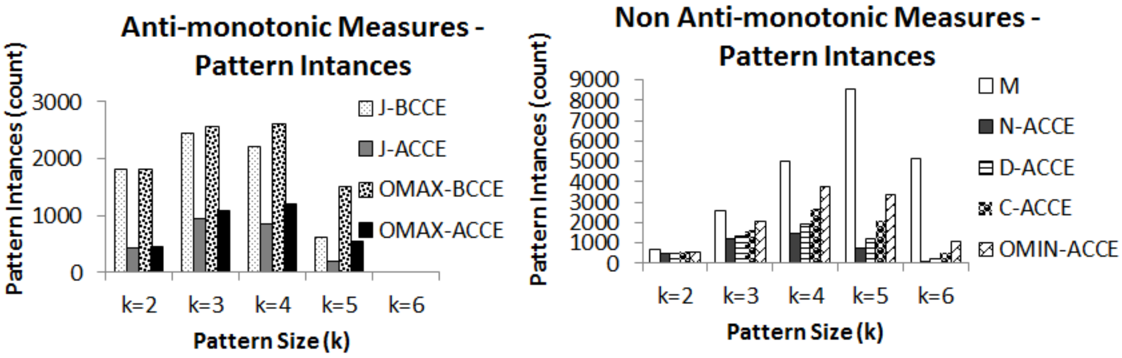
3.1.6 Experimental Evaluation

In our experiments, we use a real-life data set from the solar physics domain. Specifically, we evaluate our algorithm using the measures shown in Table 3.3 using six types of evolving solar phenomena. Our data set contains evolving instances of six different solar event types, which were observed on 01/01/2012. We obtained our data set from the well-known solar data repository called Heliophysics Event Knowledgebase (HEK) [30, 31]. The solar event types are *Active Region*, *Filament*, *Sigmoid*, *Sunspot*, *Flare*, and *Emerging Flux* [32]. Each of these solar event types has different spatial and temporal characteristics (i.e., area, duration).

We investigate STCOPs-Miner with the measures to accurately capture the STCOPs of the six different solar event types represented as evolving polygons. Moreover, an interesting ordering relation on the selectivity of the boolean versions of $J,$ $OMAX,$ $N,$ $D,$ $C,$ and $OMIN$ measures is shown in [33]. We show the ordering relation of the measures on real numbers in our experiments. Specifically, we empirically show that the value of J, $OMAX$, N, D, C, and $OMIN$ for a *size-k* pattern instance follows the ordering $J \leq OMAX \leq N \leq D \leq C \leq OMIN$ for all real positive numbers. We compare and report the number of candidate pattern instances needed to discover actual pattern instances, the storage space requirements of the measures, and the number of rules discovered. For all experiments, the cce_{th} values were set to 0.01, pi_{th} values were set to 0.1, cp_{th} values were set to 0.6, and the sampling time interval Δt is set at 30 minutes. All experiments were performed using PostgreSQL 9.1.4 and PostGIS 1.5.4.

3.1.7 Pattern Instances

We first investigated the number of candidate pattern instances that are used to generate the pattern instances that satisfy the threshold cce_{th} for anti-monotonic and non anti-monotonic measures. In Fig. 3.15 (a), we show the number of pattern instances used by STCOPs-Miner with anti-monotonic measures for different pattern sizes. In Fig. 3.15 (a), J-BCCE (OMAX-BCCE) represent the number of candidate pattern instances generated with measure J ($OMAX$), and J-ACCE (OMAX-ACCE) represent the number of pattern instances after filtering out the candidates that do not satisfy the threshold cce_{th} in the STCOPs-Miner algorithm. In other words, J-BCCE (OMAX-BCCE) compared to J-ACCE (OMAX-ACCE) can be interpreted as the ratio of candidates to actual patterns in classical Apriori. From Fig. 3.15 (a), we can observe that the number of candidate pattern instances and actual patterns for the measures J and $OMAX$ follows the ordering $J \leq OMAX$. In other words, the measure J is more restrictive in the number of pattern instances that satisfy cce_{th} in comparison to $OMAX$.



a) Anti-monotonic Measures Pattern Instances

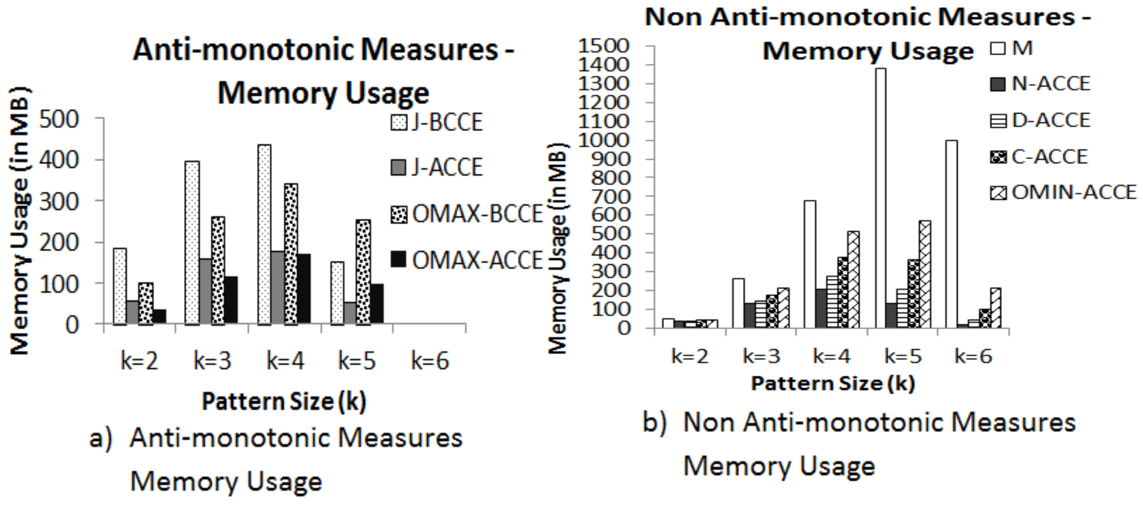
b) Non Anti-monotonic Measures Pattern Instances

Figure 3.6: Comparison of candidate and actual pattern instances generated by anti-monotonic and non anti-monotonic measures in the STCOPs-Miner algorithm.

In Fig. 3.15 (b), we show the number of pattern instances used by the STCOPs-Miner algorithm with non anti-monotonic measures for different pattern sizes. In Fig. 3.15 (b), M represents the number of candidate pattern instances generated (i.e., pattern instances that have $V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k) > 0$), and N-ACCE, D-CCE, C-ACCE, and OMIN-ACCE represent the number of pattern instances that satisfy the threshold cce_{th} in the STCOPs-Miner algorithm (i.e., the actual patterns that are reported on the output). In comparison to the anti-monotonic measures, we keep the candidate pattern instances that do not satisfy the threshold cce_{th} for the N, D, C, and $OMIN$ measures. This shows the effectiveness of our anti-monotonic measures for pruning the pattern instances that do not satisfy the threshold cce_{th}. Moreover, from Fig. 3.15 (b) we can observe that the number of pattern instances that satisfy the threshold cce_{th} for the measures N, D, C, and $OMIN$ follows the order $N \leq D \leq C \leq OMIN$.

3.1.8 Memory Usage

We now investigate the memory usage of the STCOPs-Miner algorithm for the candidate table instances generated. We report the hard-drive memory usage of candidate table instances with all the pattern instances generated, and memory usage of candidate table instances after filtering the pattern instances that do not satisfy cce_{th}. In Fig. 3.7 (a), we show the memory usage of table instances used by the STCOPs-Miner algorithm with anti-monotonic measures for different pattern sizes. In Fig. 3.7 (a), J-BCCE represents the memory usage of table instances for all pattern instances generated, and J-ACCE represents the memory usage of table instances after filtering out the pattern instances that do not satisfy the threshold cce_{th} in the STCOPs-Miner algorithm. As expected, from Fig. 3.7 (a) we can observe that there is a drop in the memory usage after the pattern instances are filtered by applying the threshold cce_{th} (compare J-BCCE (OMAX-BCCE) with J-ACCE (OMAX-ACCE)). This shows the effectiveness of the anti-monotonic property of the measures J and $OMAX$. However, generation of all J-BCCE's (OMAX-BCCE's) is necessary to discover actual STCOPs. Furthermore, from Fig. 3.7 (a), we can observe that the memory usage J is more expensive than $OMAX$ due to cost of union geometries needed for the calculation of J (see Table 3.3). However, when the number of candidate pattern instances increases, the measure $OMAX$ uses more memory (see patterns of *size-5* in Fig. 3.7 (a)).



a) Anti-monotonic Measures Memory Usage

b) Non Anti-monotonic Measures Memory Usage

Figure 3.7: Memory usage used by candidate table instances for the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.

In Fig. 3.7 (b), we show the memory usage of table instances used by the STCOPs-Miner algorithm with non anti-monotonic measures for different pattern sizes. In Fig. 3.7 (b), M represents the memory usage of table instances for all the pattern instances generated (i.e., pattern instances that have $V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k) > 0$), and N-ACCE, D-CCE, C-ACCE, and OMIN-ACCE represent the memory usage of table instances with pattern instances that satisfy the threshold cce_{th} in the STCOPs-Miner algorithm with the measures N, D, C, and $OMIN$, respectively. In other words, M compared to N-ACCE, D-CCE, C-ACCE, and OMIN-ACCE can be interpreted as candidates to actual patterns ratio in classical Apriori. However, in comparison to the J and $OMAX$ (i.e., anti-monotonic measures) we do not filter candidate pattern instances for the not anti-monotonic measures (i.e., N, D, C, and $OMIN$). Thus, for N, D, C, and $OMIN$, the number of candidate pattern instances used to generate patterns of higher sizes is greater than J and $OMAX$. However, note that the memory used by the pattern instances satisfying cce_{th} is similar to the memory used by J and $OMAX$ (compare J-ACCE, OMAX-ACCE with N-ACCE, D-CCE, C-ACCE, and OMIN-ACCE in 3.7 (a) and (b)).

3.1.9 Rules Discovered

Finally, we investigate the number of rules generated using the anti-monotonic and non anti-monotonic measures with the STCOPs-Miner algorithm and report it in Fig. 3.8. We can observe from Fig. 3.8 that the number of rules discovered varies for different measures. The number of rules generated directly depends on the number of unique patterns discovered (patterns satisfying the threshold pi_{th}) and on the selectivity of the measures. Note, the confidence of co-occurrence rules are calculated using the conditional probability (Def. 9). The selectivity of the measures directly influences the confidence of a co-occurrence rule.

The importance of analyzing different measures is shown here in order to accurately capture the spatiotemporal characteristics of different solar events. For instance, J acts similar to measure D [29]; however, it penalizes objects with smaller Intersection volumes, i.e., it gives much lower values than D to objects which have a small Intersection volume - giving a penalty to some of our events that are small

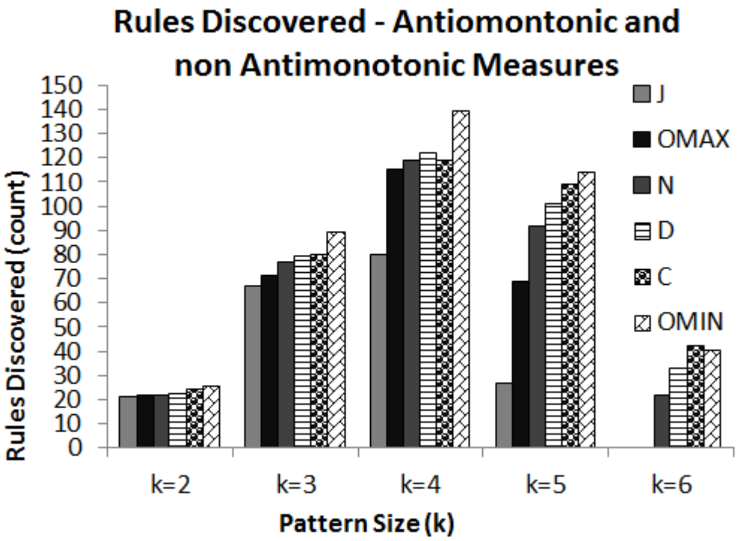


Figure 3.8: Number of rules discovered from the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.

in the area and short-lasting. Similarly, the measures *OMAX* and *N* also penalize objects with smaller Intersection (common) volume. The measure *OMIN* [29] gives a value of one if an object is totally contained within another object. We could say that it reflects inclusion, which benefits the objects that are almost equal in space and time. The measure *C* [29] is more resistant to the size of the objects, making it more appropriate to data sets that contain event types with different life spans and areas (sizes).

3.2 FastSTCOPs-Miner

Following our initial investigation on finding spatiotemporal co-occurrence patterns from data sets with continuously evolving spatiotemporal events that have extended spatial representations with anti-monotonic and non anti-monotonic measures, we next focused on developing a computationally efficient STCOPs mining algorithm

(FastSTCOPs-Miner), using a filter-and-refine strategy to prune irrelavent STCOPs based on the usage of the *all-confidence* ($OMAX$) measure as a filtering mechanism for *Jaccard*-based analysis which is the standard measure in data mining [4, 28, 19]. We provide a theoretical analysis to show the correctness and completeness of our FastSTCOPs-Miner algorithm. We experimentally verify the correctness of proposed algorithm with our naïve STCOPs algorithm [4], [1] on three real-life data sets and one artificial data set, and provide extended experimental results demonstrating the computational and memory efficiency of the FastSTCOPs-Miner algorithm.

3.2.1 Problem Statement

Input:

1. A set of spatiotemporal event types $E = \{e_1, e_2, \ldots, e_M\}$ over a common spatiotemporal framework.

2. A set of N spatiotemporal event instances $I = \{i_1, i_2, \ldots, i_N\}$, where each $i_j \in I$ is a tuple <*instance-id, spatiotemporal event type, sequence of* <*2D shape, matching time instant*> *pairs*>, where the sequence of $2D$ shape and matching time instant pairs reflects the evolution of the given spatiotemporal event.

3. A user-specified spatiotemporal co-occurrence coefficient threshold (cce_{th}).

4. A user-specified participation index threshold (pi_{th}).

5. A time interval of data sampling (Δt). All events are sampled with the same interval making the shapes of individual events exactly aligned in time.

Objective/Output: Find the complete and correct result set of spatiotemporal co-occurrence patterns (STCOPs) satisfying $cce > cce_{th}$ and $pi > pi_{th}$.

3.2.2 Analysis of relations between J and $OMAX$ measures

As discussed in Sec. 3.1.2 we use the spatiotemporal co-occurrence coefficient (*cce*) to assess the strength of the spatiotemporal relation Overlap. *cce* is typically calculated for a *size-k* pattern instance as the ratio $J = \frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k)}$. The symbol J stands for the *Jaccard* measure, which is commonly accepted by data mining practitioners to measure the co-occurence of items in shopping baskets [29, 19], among spatial objects [28], and in spatiotemporal data [4], [1].

We would like to point out here that computing J for spatiotemporal pattern instances is quite expensive (due to the necessary calculations of intersection and union geometries for each time stamp, and storage space required to save these geometries). In this work, we introduce an alternate measure $OMAX$, defined as $\frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{\max(V(i_1), \ldots, V(i_k))}$, that can be effectively used to assess the spatiotemporal co-occurrence strength of a pattern instance, and it will provide significant speed-up for discovery of STCOPs based on the commonly used *Jaccard* measure. $OMAX$ is the foundation of our filter-and-refine approach. We filter out the pattern instances that do not satisfy the user-specified threshold, cce_{th}, with $OMAX$ (we will prove that such patterns can not satisfy J with the same cce_{th} as well), and then calculate J for the reduced set of pattern instances. $OMAX$ represents the *all-confidence* measure [34] in classical association rules mining literature and it is time and storage-wise significantly cheaper to calculate than *Jaccard* on spatiotemporal data. We will show the proofs for the completeness of STCOPs generated with our filter-and-refine approach as well as experimental results confirming our theoretical investigations and space and time scalability of our approach through the rest of this chapter.

For the filter step with $OMAX$ for the J-based Apriori algorithm to be correct the following properties between J and $OMAX$ are necessary: (1) We show that the *cce* values computed using the J and $OMAX$ are monotonically non-increasing as the size of the pattern instance increases for a fixed cce_{th} value (see Lemmas 3.1 and 3.2 below). (2) We show the ordering relation on the selectivity of J and $OMAX$ (shown in Lemma 3.3 below). (3) We show that the STCOPs found using J is a subset of the STCOPs found with $OMAX$ for a fixed cce_{th} and pi_{th} values (see Lemma 3.4 below). All three properties are useful and necessary to reduce the number of candidate STCOPs in an accurate filter-and-refine strategy. This will greatly improve the performance of our naïve STCOPs algorithm [4], [1].

Lemma 3.1 : The measure J is anti-monotone (monotonically non-increasing) as the size of a pattern instance increases.

Proof: The measure J for a *size-k* pattern instance is defined as:

$$\frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k)} \tag{3.3}$$

For any *size*-$(k+1)$ pattern instance denoted as $pat\_instance'$ is equal to $pat\_instance$ $\cup$ (i_{k+1}), where $pat\_instance$ is a *size-k* pattern instance and $i_{k+1} \notin pat\_instance$. We claim the measure J follows the relation:

$$\frac{V(i_1 \cap i_2, \ldots, \cap i_k)}{V(i_1 \cup i_2, \ldots, \cup i_k)} \geq \frac{V(i_1 \cap i_2, \ldots, \cap i_k \cap i_{k+1})}{V(i_1 \cup i_2, \ldots, \cup i_k \cup i_{k+1})} \tag{3.4}$$

Therefore, we need to prove:

$$V(i_1 \cap i_2, \ldots, \cap i_k) \geq V(i_1 \cap i_2, \ldots, \cap i_k \cap i_{k+1}), \tag{3.5}$$

and

$$V(i_1 \cup i_2, \ldots, \cup i_k) \leq V(i_1 \cup i_2, \ldots, \cup i_k \cup i_{k+1}) \tag{3.6}$$

Since, adding one more instance of a different event type to a pattern instance can either reduce or not affect the volume of Intersection of instance trajectories, we obtain the relation $V(i_1 \cap i_2, \ldots, \cap i_k) \geq V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k \cap i_{k+1})$ from Eq. 3.5. Similarly, adding one more instance of different event type to a pattern instance can either increase or not affect the volume of Union of instance trajectories, we obtain the relation $V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k) \leq V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k \cup i_{k+1})$ in Eq. 3.6. Thus, our relation in Eq. 3.4 holds for all positive real numbers that represent volumes of spatiotemporal objects with evolving polygons □.

Lemma 3.2 : The measure $OMAX$ is anti-monotone (monotonically non-increasing) as the size of the pattern instance increases.

Proof: The measure $OMAX$ for a *size-k* pattern instance is defined as:

$$\frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{\max(V(i_1), \ldots, V(i_k))} \tag{3.7}$$

For any *size*-$(k+1)$ pattern instance denoted *pat_instance*$'$ is equal to *pat_instance* $\cup$ (i_{k+1}), where *pat_instance* is a *size-k* pattern instance and $i_{k+1} \notin$ *pat_instance*. We claim the measure $OMAX$ follows the relation:

$$\frac{V(i_1 \cap i_2, \ldots, \cap i_k)}{\max(V(i_1), \ldots, V(i_k))} \geq \frac{V(i_1 \cap i_2, \ldots, \cap i_{k+1})}{\max(V(i_1), \ldots, V(i_{k+1}))} \tag{3.8}$$

Therefore, we need to prove:

$$V(i_1 \cap i_2, \ldots, \cap i_k) \geq V(i_1 \cap i_2, \ldots, \cap i_k \cap i_{k+1}) \tag{3.9}$$

and

$$\max(V(i_1), \ldots, V(i_k)) \leq \max(V(i_1), \ldots, V(i_{k+1})) \tag{3.10}$$

Since, once again, adding one more instance of different event type to a pattern instance can either reduce or not affect the volume of the Intersection of the instance trajectories, we obtain the relation shown in Eq. 3.9. Similarly, adding another instance of a different event type to a pattern instance can not reduce the maximum volume of instance trajectories, so we obtain the relation $\max(V(i_1), \ldots, V(i_k)) \leq \max(V(i_1), \ldots, V(i_k), V(i_{k+1}))$. Thus, Eq. 3.8 holds for all positive real numbers that represent volumes of spatiotemporal objects with evolving polygons $\square$.

Lemma 3.3 : The selectivity of the measures J and $OMAX$ for a *size-k* pattern instance follows the order $\frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k)} \leq \frac{V(i_1 \cap i_2, \ldots, i_{k-1} \cap i_k)}{\max(V(i_1), \ldots, V(i_k))}$, $\forall V \in R^+$ for $k \geq 2$.

Proof: Since the numerators are same, for the ordering relation $J \leq OMAX$, we can derive relations between both denominators:

$$\max(V(i_1), \ldots, V(i_k)) \leq V(i_1 \cup i_2, \ldots, i_{k-1} \cup i_k) \tag{3.11}$$

Maximum volume of all trajectories is always less than or equal to the volume of union of all of them, thus the relation $J \leq OMAX$ always holds for positive real numbers $\square$.

Lemma 3.4 : For a given user-specified participation index threshold pi_{th} and spatiotemporal co-occurrence strength threshold cce_{th}, the set of STCOPs generated using J, is a subset of STCOPs generated using $OMAX$ measure, for the same cce_{th} and pi_{th}.

Proof: From *Lemma* **3.1** and **3.2**, we know that the measures J and $OMAX$ are anti-monotonic as the size of the pattern increases. Also, from *Lemma* **3.3**, we know that ordering $J \leq OMAX$ holds.

For given user-specified thresholds cce_{th} and pi_{th}, we represent the set of all STCOPs generated for J as $STCOP_J$, and the set of all STCOPs generated for $OMAX$ as $STCOP_{OMAX}$. Furthermore, we denote participation index $pi(SE_i)$ of a spatiotemporal co-occurrence SE_i (see Def. 7), derived by using J as $pi_J(SE_i)$, and utilizing measure $OMAX$ as $pi_{OMAX}(SE_i)$. From *Lemma* **3.3**, we know that the number of pattern instances found for a spatiotemporal co-occurrence SE_i follows the order $J \leq OMAX$, thus we get,

$$min_{j=1}^{k} pr_J(SE_i, e_j) \leq min_{j=1}^{k} pr_{OMAX}(SE_i, e_j) \tag{3.12}$$

$$pi_J(SE_i) \leq pi_{OMAX}(SE_i) \quad \square \tag{3.13}$$

Since participation index is anti-monotonic as the size of the pattern increases [27], and from *Lemmas* **3.1** and **3.2**, and from Eq. 3.13, we get $STCOP_J \subseteq STCOP_{OMAX} \quad \square$.

3.2.3 FastSTCOPs-Miner Algorithm

In this section, we introduce the FastSTCOPs-Miner algorithm, which is more efficient than our naïve STCOPs algorithm [4], [1] in the context of needed memory as well as the execution time while leading to exactly the same results. This is because we apply a filter-and-refine strategy in each iteration of the algorithm. The FastSTCOPs-Miner algorithm exploits the containment relation between the STCOPs generated using Jaccard (J) and $OMAX$ measures (see Sec. 3.2), to filter out candidate patterns that can not form STCOPs with the J.

The FastSTCOPs-Miner algorithm first filters STCOPs with $OMAX$, and then uses these filtered STCOPs to find the refined prevalent STCOPs with our standard measure, that is *Jaccard* (J). These refined prevalent STCOPs, like in all Apriori algorithms [17], are used to generate candidate STCOPs in the next iteration of the algorithm. Thus, the FastSTCOPs-Miner algorithm continously uses a filter-and-refine strategy at each iteration of the algorithm to generate prevalent STCOPs.

Even though such duplication of efforts may seem unnecessary, in Sec. 3.2.4. we will experimentally show on a multitude of real-life and artificial data sets, the impressive effectiveness of our filter-and-refine strategy on spatiotemporal data with evolving regions. We will also provide a detailed explanation for this effectiveness.

Next, we give the pseudocode of the proposed FastSTCOPs-Miner algorithm (see Fig. 3.9), and explain the algorithm with a running example using the data set already shown in Fig. 3.1 and Table 3.1.

For our FastSTCOPs-Miner algorithm shown in Fig. 3.9, the inputs and outputs are defined as in Sec. 3.2. Steps 1 and 2 of proposed algorithm intialize the data parameters and data structures, steps 3 through 11 give an iterative process to discover the STCOPs of size greater than two. Steps 3 through 11 continue until there are no candidate STCOPs to be discovered as shown by loop condition in step 3. Step 12 returns the union of all prevalent STCOPs (patterns of all sizes). The explanations of functions in the algorithm are:

Generation of table instances of *size-1* (step 2). In this function, argument Δt represents the size of increment in time. The evolution of instances of our spatiotemporal events from their birth (start) time is registered using Δt as our time sampling frequency. The combination of the event instance ID and time step allows us to identify the appropriate spatial representation of an event at the particular moment. For example, Fig. 3.10 (a) shows the key columns of table instances of *size-1* for our sample spatiotemporal data set (Fig. 3.1 and Table 3.1). Here, the Δt value was set to 10 minutes. The column denoted $tab\_ins(e_1)$ represents the table instance of *size-1* for event type e_1. Similarly, the columns denoted by $tab\_ins(e_2)$, $tab\_ins(e_3)$, and $tab\_ins(e_4)$ represent the table instances of *size-1* for event types e_2, e_3, and e_4. The geometric shapes of instances in each of the presented time instances are not shown in Fig. 3.10 (a) for simplicity.

Inputs :

See Sec. 3.2.

Variables :

(1) k the co-occurrence size

(2) CR_{Jk}: the set of candidates for size-(k) STCOPs derived from *size-*$(k-1)$ refined prevalent STCOPs

(3) T_{OMAXk}: a set of filtered instances of size-(k) spatiotemporal co-occurrences

(4) TR_{Jk}: a set of refined instances of size-(k) spatiotemporal co-occurrences

(5) P_{OMAXk}: the set of *size-k* filtered STCOPs derived from *size-k* candidate STCOPs

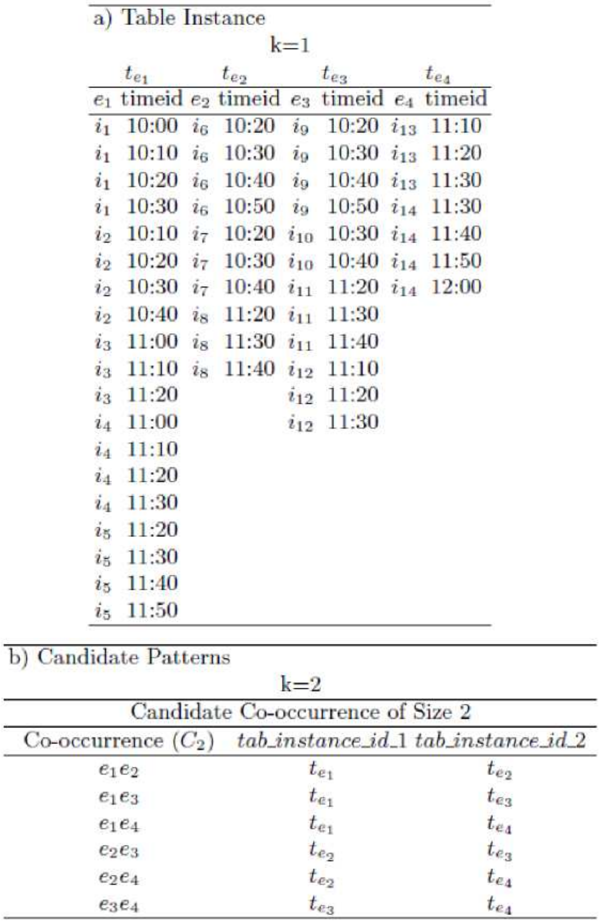
(6) PR_{Jk}: the set of *size-k* refined prevalent STCOPs derived from *size-k* filtered STCOPs

(7) PR_{final}: the union of all refined prevalent STCOPs (patterns of all sizes). // This is the final *Jaccard*-based prevalent patterns

Algorithm :

```
$k$=1, $C_k$=$E$, $PR_{Jk} = E$; $PR_{final} = \emptyset$;
$TR_{Jk} = gen\_loc(C_k, I, \Delta t)$;
**while** $(PR_{Jk} \neq \emptyset)$ {
  $CR_{J(k+1)} = gen\_candidate\_coocc(PR_{Jk})$;
  $T_{OMAX(k+1)} = gen\_tab\_ins\_coocc\_filtered(CR_{J(k+1)}, TR_{Jk}, cce_{th})$;
  $P_{OMAX(k+1)} = pre\_prune\_coocc\_filtered(CR_{J(k+1)}, T_{OMAX(k+1)}, pi_{th})$;
  $TR_{J(k+1)} = gen\_tab\_ins\_coocc\_refined(P_{OMAX(k+1)}, T_{OMAX(k+1)}, cce_{th})$;
  $PR_{J(k+1)} = pre\_prune\_coocc\_refined(P_{OMAX(k+1)}, TR_{J(k+1)}, pi_{th})$;
  $PR_{final} = PR_{final} \cup PR_{J(k+1)}$;
  $k = k + 1$;
}
return $PR_{final}$;
```

Figure 3.9: FastSTCOPs-Miner Algorithm



a) Table Instance

k=1

t_{e_1}		t_{e_2}		t_{e_3}		t_{e_4}	
e_1	timeid	e_2	timeid	e_3	timeid	e_4	timeid
i_1	10:00	i_6	10:20	i_9	10:20	i_{13}	11:10
i_1	10:10	i_6	10:30	i_9	10:30	i_{13}	11:20
i_1	10:20	i_6	10:40	i_9	10:40	i_{13}	11:30
i_1	10:30	i_6	10:50	i_9	10:50	i_{14}	11:30
i_2	10:10	i_7	10:20	i_{10}	10:30	i_{14}	11:40
i_2	10:20	i_7	10:30	i_{10}	10:40	i_{14}	11:50
i_2	10:30	i_7	10:40	i_{11}	11:20	i_{14}	12:00
i_2	10:40	i_8	11:20	i_{11}	11:30		
i_3	11:00	i_8	11:30	i_{11}	11:40		
i_3	11:10	i_8	11:40	i_{12}	11:10		
i_3	11:20			i_{12}	11:20		
i_4	11:00			i_{12}	11:30		
i_4	11:10						
i_4	11:20						
i_4	11:30						
i_5	11:20						
i_5	11:30						
i_5	11:40						
i_5	11:50						

b) Candidate Patterns

k=2

Candidate Co-occurrence of Size 2		
Co-occurrence (C_2)	*tab_instance_id_1*	*tab_instance_id_2*
e_1e_2	t_{e_1}	t_{e_2}
e_1e_3	t_{e_1}	t_{e_3}
e_1e_4	t_{e_1}	t_{e_4}
e_2e_3	t_{e_2}	t_{e_3}
e_2e_4	t_{e_2}	t_{e_4}
e_3e_4	t_{e_3}	t_{e_4}

Figure 3.10: (a) Table Instances of size-1 and (b) Candidate Patterns for size-2 STCOPs.

Generation of candidate co-occurrence patterns (step 4). This function uses an Apriori-based approach to generate candidates of *size-(k+1)* using *size-k* refined prevalent STCOPs (i.e. our PR_{Jk} in Fig. 3.9). However, for $k = 1$ this function uses spatiotemporal event types to generate candidates of *size-2* (i.e. $PR_{J1} = E$ from step 1 in Fig. 3.9).

Generation of filtered table instances of *size-(k+1)* (step 5). This function generates table instances for candidate patterns of *size-(k+1)*. Pattern instances for each table instance can be generated by a spatiotemporal join query. The geometric shapes of the instances at each time step are saved, as these geometric shapes will be used for finding the *cce* of STCOPs of size three or more. In this function, we calculate the *cce* for each pattern instance by using $OMAX$. Pattern instances that have a *cce* below the user-specified cce_{th} value are deleted from the table instance, since we know from proofs in Sec. 3.2. that they also cannot satisfy cce_{th} requirement for J measure.

For example, Fig. 3.11 (c) shows the important columns of *size-2* filtered table instances for our sample spatiotemporal data set from Fig. 3.1 and Table 3.1. The column denoted by $tab\_ins(e_1e_2)$ represents the table instance of *size-2* co-occurrence of event types e_1 and e_2. Similarly, the other columns represent the table instance of different event types. We also show the pattern instances that satisfy the threshold $cce_{th} = 0.01$ calculated using $OMAX$. Moreover, we only show the key columns of table instances for simplicity. For example, in the table instance $tab\_ins(e_1e_2)$ shown in Fig. 3.11 (c), the rows $i_1, i_6,$ 10:00 through $i_1, i_6,$ 10:50 represent a pattern instance that satisfies the threshold $cce_{th} = 0.01$. As another example, in Fig. 3.12 (f) we show the filtered table instances generated from candidate patterns of size $k = 3$.

Generation of filtered prevalent patterns *size-(k+1)* (step 6). This function discovers filtered *size-(k+1)* STCOPs by pruning candidate patterns in $CR_{J(k+1)}$ that have $pi < pi_{th}$.

For example, we show the *pi* value (See Def. 7) at the end of each table instance in Fig. 3.11 (c). As seen from the Fig. 3.11 (c), the patterns $SE_i = \{e_1, e_4\}$, and $SE_j = \{e_2, e_4\}$ will be pruned if a value of 0.39 is set to pi_{th}. Thus, the patterns that satisfy the $pi_{th} = 0.39$ are $\{\{e_1, e_2\}, \{e_1, e_3\}, \{e_2, e_3\}, \{e_3, e_4\}\}$.

As another example, we show the *pi* value (See Def. 7) at end of table instance $tab\_ins(e_1e_2e_3)$ in Fig. 3.12 (f). As seen from the Fig. 3.12 (f), the pattern $SE_i = \{e_1e_2e_3\}$ is a prevalent pattern if a value of 0.39 is set to pi_{th}.

c) Filtered Table Instances for k=2

$tab\_ins(e_1e_2)$			$tab\_ins(e_1e_3)$			$tab\_ins(e_1e_4)$			$tab\_ins(e_2e_3)$			$tab\_ins(e_2e_4)$			$tab\_ins(e_3e_4)$		
e_1	e_2	timeid	e_1	e_3	timeid	e_1	e_4	timeid	e_2	e_3	timeid	e_2	e_4	timeid	e_3	e_4	timeid
i_1	i_6	10:00	i_1	i_9	10:00	i_3	i_{13}	11:00	i_6	i_9	10:20	i_8	i_{14}	11:20	i_{11}	i_{13}	11:10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i_1	i_6	10:50	i_1	i_9	10:50	i_3	i_{13}	11:30	i_6	i_9	10:50	i_8	i_{14}	12:00	i_{11}	i_{13}	11:40
							0.20						0.33				
i_2	i_7	10:10	i_2	i_{10}	10:10				i_7	i_{10}	10:20				i_{12}	i_{14}	11:10
⋮	⋮	⋮	⋮	⋮	⋮				⋮	⋮	⋮				⋮	⋮	⋮
i_2	i_7	10:40	i_2	i_{10}	10:40				i_7	i_{10}	10:40				i_{12}	i_{14}	12:00
				0.40												0.50	
i_4	i_8	11:00							i_8	i_{12}	11:10						
⋮	⋮	⋮							⋮	⋮	⋮						
i_4	i_8	11:40							i_8	i_{12}	11:40						
	0.60									0.75							

d) Refinded Table Instances for k=2

$tab\_ins(e_1e_2)$			$tab\_ins(e_1e_3)$			$tab\_ins(e_2e_3)$			$tab\_ins(e_3e_4)$		
e_1	e_2	timeid	e_1	e_3	timeid	e_2	e_3	timeid	e_3	e_4	timeid
i_1	i_6	10:00	i_1	i_9	10:00	i_6	i_9	10:20	i_{11}	i_{13}	11:10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i_1	i_6	10:50	i_1	i_9	10:50	i_6	i_9	10:50	i_{11}	i_{13}	11:40
i_2	i_7	10:10	i_2	i_{10}	10:10					0.25	
⋮	⋮	⋮	⋮	⋮	⋮	i_7	i_{10}	10:20			
i_2	i_7	10:40	i_2	i_{10}	10:40	⋮	⋮	⋮			
						i_7	i_{10}	10:40			
	0.40			0.40			0.50				

Figure 3.11: (c) Filtered (T_{OMAX2}) and (d) Refined (TR_{J2}) Table Instances of size-2.

Generation of refined table instances of *size-(k+1)* (step 7). This function generates table instances for filtered prevalent STCOPs of *size-(k+1)*. Pattern instances for each table instance can be generated by using the table instances of step 5; however, additionally this function also generates and saves the Union geometries at each time step of the pattern instance. We calculate the *cce* for each pattern instance by using J measure. Pattern instances that have *cce* less than the user-specified cce_{th} value are deleted from the table instance.

For example, in Fig. 3.11 (d) we show the refined table instances generated from the refined prevalent patterns obtained in step 6 for a pi_{th} value of 0.39. In each of the

e) Refined Candidate Patterns for k=3

Refinded Candidate Co-occurrence of Size 3		
Refined Co-occurrence (CR_{J3})	*tab_instance_id_1*	*tab_instance_id_2*
$e_1e_2e_3$	$tab\_ins(e_1e_2)$	$tab\_ins(e_1e_3)$

f) Filtered Table Instances for k=3

$tab\_ins(e_1e_2e_3)$			
e_1	e_2	e_3	timeid
i_1	i_6	i_9	10:00
⋮	⋮	⋮	⋮
i_1	i_6	i_9	10:50
i_2	i_7	i_{10}	10:00
⋮	⋮	⋮	⋮
i_2	i_7	i_{10}	10:40
0.40			

g) Refined Table Instances for k=3

$tab\_ins(e_1e_2e_3)$			
e_1	e_2	e_3	timeid
i_2	i_7	i_{10}	10:00
⋮	⋮	⋮	⋮
i_2	i_7	i_{10}	10:40
0.20			

Figure 3.12: (e) Candidate Patterns of size-3 and (f) Filtered (T_{OMAX3}) and (g) Refined (TR_{J3}) Table Instances of size-3.

table instances shown in Fig. 3.11 (d), we show the key columns of pattern instances that satisfy the threshold cce_{th} = 0.01 value calculated using the J measure. For example, for the table instance $tab\_ins(e_1e_2)$, the pattern instances that satisfy the $cce_{th} = 0.01$ for J are $\{\{i_1, i_6\}, \{i_2, i_7\}\}$. Note the pattern instance $\{i_4, i_8\}$ is dropped from the table instance $tab\_ins(e_1e_2)$ as it does not satisfy $cce_{th} = 0.01$ (see Fig. 3.11 (c) and (d) to compare).

As another example, in Fig. 3.12 (g) we show the refined table instances generated from refined prevalent patterns obtained in step 6 for a pi_{th} = 0.39. In the table instance $tab\_ins(e_1e_2e_3)$ shown in Fig. 3.12 (g), we show the key columns of pattern instances that satisfy the threshold cce_{th} value calculated using J. Please note the pattern instance $\{i_1, i_6, i_9\}$ is dropped from the table instance $tab\_ins(e_1e_2e_3)$ as its cce is smaller than our $cce_{th} = 0.01$ (see Fig. 3.12 (f) and (g) to compare).

Generation of refined prevalent patterns *size-(k+1)* (step 8). This function discovers refined *size-(k+1)* prevalent STCOPs by pruning $P_{OMAX(k+1)}$ that have

$pi < pi_{th}$. As seen from the Fig. 3.12 (g), the pattern $SE_i = \{e_1, e_2, e_3\}$ will be pruned if pi_{th} is set to 0.39.

In step 9, we calculate the union of refined prevalent patterns. The algorithm runs iteratively until no more STCOPs can be generated (our $PR_{J(k+1)}$ is empty), and returns all prevalent STCOPs, in step 12. Since we do not have any patterns left that satisfies the threshold $pi_{th} = 0.39$ in our example data set shown in Fig. 3.1 and Table 3.1, the algorithm would terminate at $k = 3$ for our running example.

3.2.4 Experimental Evaluation

In this section, we compare our FastSTCOPs-Miner algorithm against the classic Apriori-based approach [4], [1] which we call Naïve STCOPs algorithm. In our experiments, we are using three real-life data sets from the solar physics domain and one artificial data set.

In the real-life data sets, we evaluate our algorithms using six types of evolving solar phenomena. Our real-life data sets contain evolving instances of six different solar event types, which were observed on 01/01/2012 (denoted Data Set A), 01/01/2012 through 01/03/2012 (denoted Data Set B), and 01/01/2012 through 01/05/2012 (denoted Data Set C). We obtained our data sets from the well-known solar data repository called Heliophysics Event Knowledgebase (HEK) [31],[35]. The six different solar event types in our data sets are: *Active Region*, *Filament*, *Sigmoid*, *Sunspot*, *Flare*, and *Emerging Flux* [32].

The artificial data set (denoted Data Set D) is generated based on the works of Huang et al. in [27]. The artificial data set generator creates a data set of event instances with spatiotemporal features for spatial framework of size $D \times D$. Event types are generated with random size, speed, duration and area change parameters.

Number of events to be generated is an input parameter to dataset generator, M. We used artificial data set to investigate the behaviour of algorithms for a larger number of event types. All of our data sets are available on-line to let researchers interested in this topic reproduce our experiments, and maybe even improve on our solution. The website for this research can be found at [36].

We investigated the FastSTCOPs-Miner algorithm and Naïve STCOPs algorithm to accurately capture the STCOPs of the six different solar event types in the real-life data sets, and nine different artificial event types in the artificial data set. In all four data sets instances of different event types are represented as evolving polygons, where each instance of these events has significantly different spatial size, duration of life time and dynamics of its evolution. We compare and report the number of pattern instances found, the execution time of the algorithms, and the storage space requirements of the algorithms. For the three real-life data sets, for both algorithms, the cce_{th} values were set to 0.01, pi_{th} values were set to 0.1, and the sampling time interval Δt were set at 30 minutes leading to exactly the same set of final STCOPs. For the artificial data set, for both algorithms, the cce_{th} values were set to 0.01, pi_{th} values were set to 0.05, and the sampling time interval Δt were set at 3 minutes. All experiments were performed using PostgreSQL 9.1.4 and PostGIS 1.5.4. We report results highlighting memory usage efficiency and execution time of our FastSTCOPs-Miner algorithm in comparison to the Naïve STCOPs algorithm.

3.2.5 Memory Usage Comparison

We first investigated the memory usage of the FastSTCOPs-Miner and Naïve STCOPs algorithms for the candidate table instances generated. We report the hard-drive memory usage of candidate table instances with all the pattern instances gen-

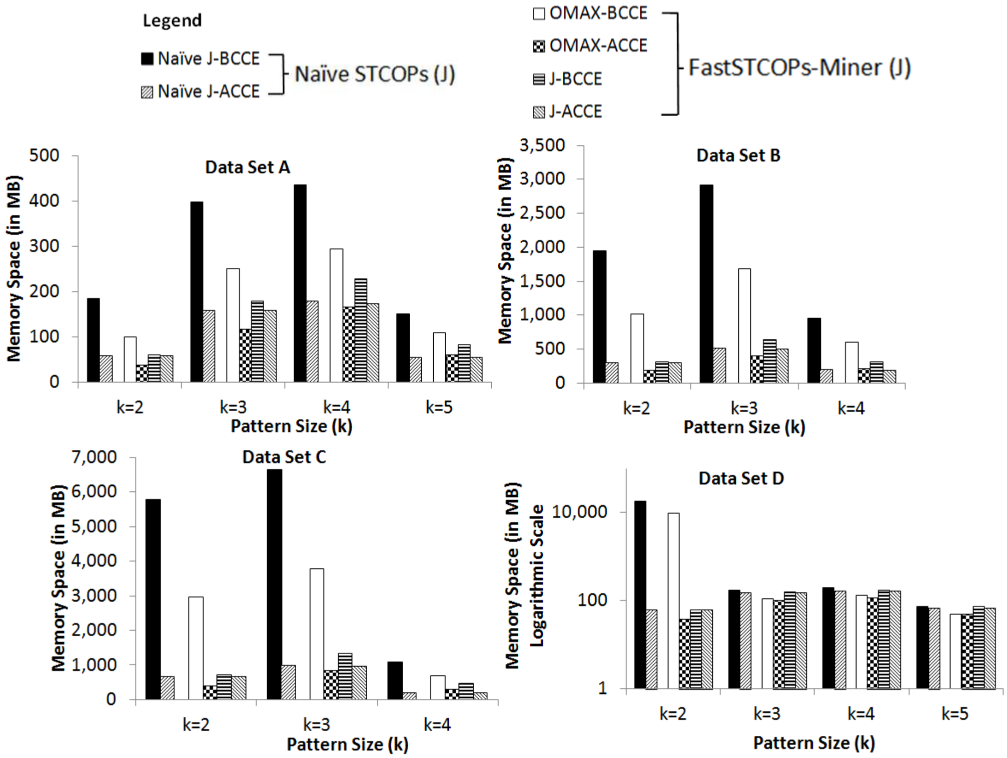


Figure 3.13: Memory usage used by candidate table instances for the FastSTCOPs-Miner and Naïve STCOPs algorithms.

erated (see bars in solid colors - black and white in Fig. 3.13), and memory usage of candidate table instances after filtering the pattern instances that do not satisfy threshold cce_{th} (see bars with pattern markings in Fig. 3.13). In Fig. 3.13 bar labelled Naïve J-BCCE (black bars) represents the memory usage of table instances for all the pattern instances generated (i.e. the candidates), and Naïve J-ACCE (bars with diagonal upward stripes) represents the memory usage of table instances after pruning out the pattern instances that do not satisfy the threshold cce_{th} set up for our experiments (i.e. our actual patterns that are reported on the output). In other words, J-BCCE compared to J-ACCE can be interpretted as candidates to actual patterns ratio in classical Apriori and are indicators of measure selectivity. As we can see from the first two bars in the first chart shown in Fig. 3.13 $Jaccard$ measure was heavily used to prune the 187 MB of candidates for pattern instances to 50 MB of actual *size-2* pattern instances. This is what has happened in the Naïve STCOPs algorithm. The next four bars in each chart shown in Fig. 3.13 show effectiveness of our FastSTCOPs-Miner algorithm. The first of these four bars (labelled OMAX-BCCE, in white solid color) represents candidate pattern instances that we are about to filter using $OMAX$ measure. As we can see this amount of memory is already smaller than storage needed for candidates (compare Naïve J-BCCE$\approx$ 187 MB vs. OMAX-BCCE$\approx$ 99.46 MB in Fig. 3.13 Data Set A). This is because of the fact that to accomplish pruning using $Jaccard$ measure we have to precompute and store both Union and Intersection volumes for the co-occurring patterns, while when we use $OMAX$ only Intersection volumes are needed. This is what causes the reduction in storage in the FastSTCOPs-Miner algorithm (from 187 MB in the first black/Naïve J-BCCE bar to 99.46 MB in the first white/OMAX-BCCE bar). This benefit continues through remaining steps of our algorithm. As expected, from Fig. 3.13 we can observe, that there is a drop in the memory usage after the pattern instances are pruned out by applying the threshold cce_{th} (see and compare Naïve J-BCCE in black color with Naïve J-ACCE marked with diagonal upward stripes in Fig. 3.13). However, generation of all J-BCCE's is neccessary to discover actual STCOPs (i.e our J-ACCE's).

Also, in Fig. 3.13 OMAX-BCCE represents the memory usage of table instances for all the pattern instances generated. OMAX-ACCE represents the memory usage of table instances after filtering the pattern instances that do not satisfy the threshold cce_{th} in FastSTCOPs-Miner algorithm. This time OMAX-BCCE to OMAX-ACCE ratio represents selectivity (i.e. pruning power) of our filter step in the FastSTCOPs-Miner algorithm. Please note, here the cce value is calculated using $OMAX$, so the J-based refine step is fed by the filtered out data (i.e. satisfying cce_{th} and pi_{th} (pi_{OMAX})) but without missing any relevant patterns. Moreover, J-BCCE represents the memory usage of table instances for all the pattern instances generated from the filtered pattern instances (that is from our OMAX-ACCE bars that satisfy pi_{th} (pi_{OMAX})) and J-ACCE represents the memory usage of table instances after filtering the pattern instances that do not satisfy the threshold cce_{th} in FastSTCOPs-Miner algorithm. Please note, here the cce value is calculated using J, to find patterns that are relevant. Also, the total number of pattern instances for OMAX-ACCE (bars with checkered pattern) is greater than or equal to J-BCCE (because of filtering effect of threshold pi_{th} (pi_{OMAX})); however, the memory usage increases for J-BCCE because of the union geometries generated for all of the pattern instances in order to calculate cce's using J measure. Furthermore, from Fig. 3.13 we can observe a decrease in the memory usage after the pattern instances are filtered by applying threshold cce_{th} (please compare OMAX-BCCE with OMAX-ACCE and J-BCCE with J-ACCE, respectively). This shows the effectiveness of the anti-monotone property of the measures J and $OMAX$ (see Lemma 3.1 and 3.2) and the benefit of our $OMAX$-based pruning strategy (see Lemma 3.4).

3.2.6 Execution Time Comparison

Next, we show the execution times of our FastSTCOPs-Miner and Naïve STCOPs algorithms. Fig. 3.14 shows the execution time for patterns of different sizes. As expected, our FastSTCOPs-Miner algorithm outperforms the original Naïve STCOPs algorithm, since it uses a filter-and-refine strategy to find pattern instances that satisfy the threshold cce_{th} for J. The Naïve STCOPs algorithm generates computationally expensive Union geometries for all the pattern instances (see the bars labelled as Naïve

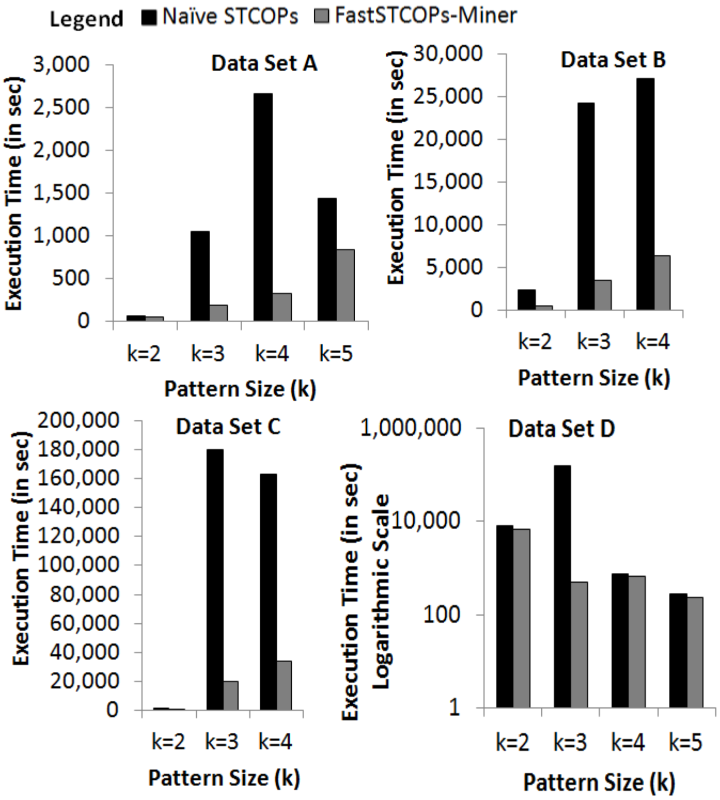


Figure 3.14: Camparison of execution time for the FastSTCOPs-Miner and Naïve STCOPs.

J-BCCE in Fig. 3.13 to realize how much memory overhead this process generates for data sets A, B, C, and D), while our FastSTCOPs-Miner algorithm generates Union geometries for smaller data set (see the bars labelled as J-BCCE in Fig. 3.13). This memory overhead causes the execution time of the Naïve STCOPs algorithm to be slower in comparison to our FastSTCOPs-Miner algorithm.

3.2.7 Pattern Instances Comparison

Fig. 3.15 shows the counts of pattern instances that satisfy the threshold value cce_{th}. We compare the counts of pattern instances satisfying the threshold cce_{th} with $OMAX$ (FastSTCOPs-Miner (OMAX)) and J (FastSTCOPs-Miner (J)) for our

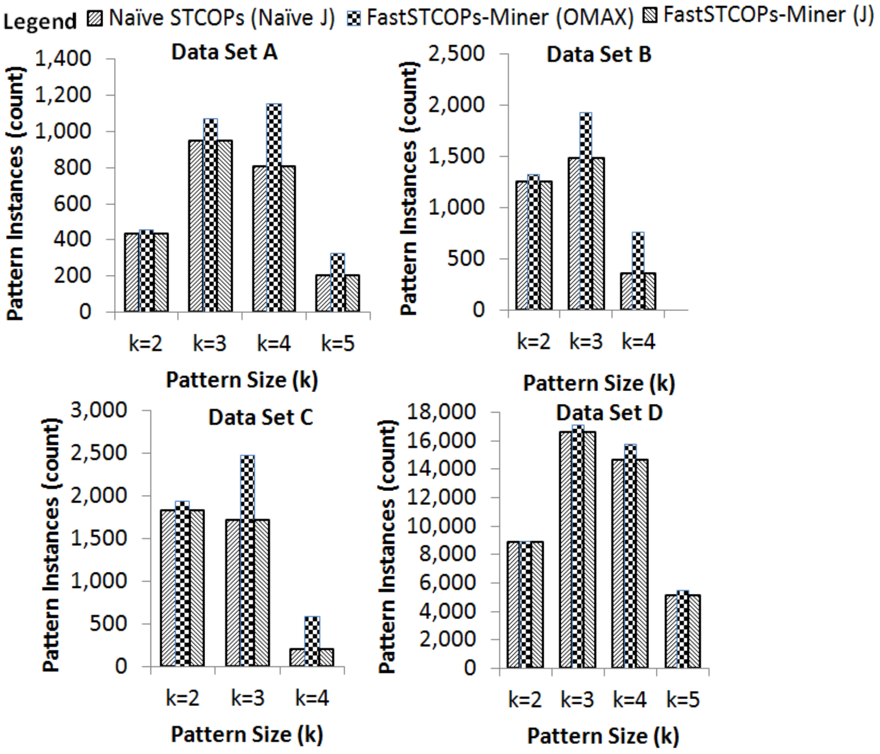


Figure 3.15: Pattern instances found (filtered with OMAX and refined with J) using the FastSTCOPs-Miner and Naïve STCOPs.

FastSTCOPs-Miner algorithm. We also compared the counts of pattern instances of the FastSTCOPs-Miner and Naïve STCOPs algorithms. As shown in Lemma 3.3, we can observe in Fig. 3.15 that the selectivity of the measures $OMAX$ and J follows the order $OMAX \geq J$. Moreover, the count of pattern instances found for the threshold value cce_{th} is the same for our FastSTCOPs-Miner (J) and the Naïve STCOPs (J) algorithm in Fig. 3.15. The identical results between FastSTCOPs-Miner (J) and Naïve STCOPs (J) provides evidence for the correctness of our implementation of the FastSTCOPs-Miner algorithm.

CHAPTER 4
REMAINING WORK

This chapter covers our ongoing and future work. Since we based our algorithms on an Apriori approach we would like to develop algorithms to mine STCOPs using a frequent pattern (FP) growth approach [37] and compare it with our existing Apriori-based algorithms. We next plan to use the most effective of our four algorithms to perform large-scale discovery of STCOPs from the massive SDO data and submit a paper in the solar physics domain.

4.1 Frequent Pattern Growth

We would like to conclude our research focus with developing algorithms based on a frequent pattern growth approach and comparing it with our current algorithms [4], [1], [10] that uses an Apriori-based method. The frequent pattern growth approach does not use the generate-and-test strategy of Apriori. Instead, it converts the data set using a compact data structure called an **FP-tree** and it discovers frequent itemsets directly from this structure. The frequent pattern growth approach is shown to outperform Apriori-based approaches [37]; however, the results were presented on shopping basket data, and no versions of the FP algorithm for spatiotemporal data with evolving regions exist. In this research we plan to develop an algorithm that will incorporate both filter-and-refine (introduced in chapter 3) and frequent pattern growth approaches. We also plan to do a comparative evaluation of four algorithms (i.e., STCOPs-Miner, FastSTCOPs-Miner, Naïve Frequent pattern growth, and Filter-and-Refine Frequent pattern growth).

4.2 Solar Physics

The expected consequence of this research is to apply our algorithms on solar physics data sets to do a large-scale verification of some known theories in solar physics. Furthermore, since humans are not good at identifying long patterns, from using our algorithms we expect to discover interesting new spatiotemporal co-occurrence patterns.

4.3 Research Plan

In this proposal we have covered our research work that had already been published [4], [1] as well as the research that is awaiting publication [10] or still in progress. We now conclude the proposal by presenting our plans about the remaining work to be done, and outline the plan and timeline to accomplish these goals. This is an approximate outline for the work that needs to be done in order to finish the dissertation and meet the research and time requirements.

1. September through November 2013: Work on developing a frequent pattern growth approach algorithm to discover STCOPs. Implement the developed algorithm, run experiments, generate results.

2. December 2013 and January 2014: Compare the results with our current Apriori-based approaches. Write an article from the results generated and submit to a journal/conference.

3. February and March 2014: Work closely with Dr. Piet Martens for large-scale verification of some known theories in solar physics using our algorithms, and submit an article to a journal in the solar physics domain.

4. April and May 2014: Compile Ph.D. dissertation by merging and unifying works we already published with the ones we plan to publish by January, and expanding them with the most recent (large-scale) results.

5. June 2014: Complete Ph.D. dissertation.

6. July 2014: Defend Ph.D. dissertation.

REFERENCES CITED

[1] P.C.H Martens. Content-based image retrieval for solar physics (invited review), Feb 2012. Workshop on Solar Statistics, Feature Recognition, Thermal Structure, Numerical Computation, and Massive Data Streams, Harvard-Smithsonian Center for Astrophysics.

[2] M. Schneider. *Spatial Data Types for Database Systems: Finite Resolution Geometry for Geographic Information Systems.* Lecture Notes in Computer Science. Springer, 1997.

[3] K. G. Pillai, R. A. Angryk, J. M. Banda, M. A. Schuh, T. Wylie. Spatiotemporal co-occurrence pattern mining in data sets with evolving regions. *ICDM Workshops*, pages 805–812, 2012.

[4] J. Grey. Data Avalanche, HP Labs show and tell, Redmond, WA. `http://research.microsoft.com/en-us/um/people/gray/JimGrayTalks.htm`, 2004. note = "[Online; accessed 09-Sep-2013]".

[5] T. Hey, S. Tansley, K. M. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009.

[6] W. D. Pesnell, B. J. Thompson, P. C. Chamberlin. The Solar Dynamics Observatory (SDO). *Solar Physics*, 275(1-2):3–15, 2012.

[7] P.C.H. Martens, G.D.R. Attrill, A.R. Davey, A. Engell, S. Farid, P.C. Grigis, J. Kasper, K. Korreck, S.H. Saar, A. Savcheva, Y. Su, P. Testa, M. Wills-Davey, P.N. Bernasconi, N.-E. Raouafi, V.A. Delouille, J.F. Hochedez, J.W. Cirtain, C.E. DeForest, R.A. Angryk, I. Moortel, T. Wiegelmann, M.K. Georgoulis, R.T.J. McAteer, R.P. Timmons. Computer Vision for the Solar Dynamics Observatory (SDO). *Solar Physics*, 275(1-2):79–113, 2012. `doi:10.1007/s11207-010-9697-y`.

[8] S. Langhof, T. Straume. Workshop report on space weather risks and society. `http://event.arc.nasa.gov/main/home/reports/CP2012216003SpaceWeather.v8.pdf`, 2011. note = "[Online; accessed 04-Sep-2013]".

[9] K. G. Pillai, R. A. Angryk, J. M. Banda, T. Wylie, M. A. Schuh. *Spatiotemporal Co-occurrence Rules*, Volume 241. Springer International Publishing, 2013.

[10] K. G. Pillai, R. A. Angryk, B. Aydin. A filter-and-refine approach to mine spatiotemporal co-occurrences. *To appear in Proceedings of the 21th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '13, New York, NY, USA, 2013. ACM.

[11] M. Worboys, M. Duckham. *GIS: A Computing Perspective.* CRC Press, Inc., Boca Raton, FL, USA, 2004.

[12] S. Shekhar, S. Chawla. *Spatial Databases: A Tour.* Prentice Hall, 2002.

[13] R.H. Güting, M. Schneider. *Moving Objects Databases.* Morgan Kaufmann, 2005.

[14] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983. `doi:10.1145/182.358434`.

[15] M. Erwig, M. Schneider. Spatio-temporal predicates. *IEEE Trans. on Knowl. and Data Eng.*, 14(4):881–901, July 2002. `doi:10.1109/TKDE.2002.1019220`.

[16] M. Koubarakis, T. K. Sellis, A. U. Frank, S. Grumbach, R. H. Güting, C. S. Jensen, N. A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H. Schek, M. Scholl, B. Theodoulidis, N. Tryfona. *Spatio-Temporal Databases: The CHOROCHRONOS Approach.* Springer, 2003.

[17] R. Agrawal, T. Imieliński, A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993. `doi:10.1145/170036.170072`.

[18] S. Shekhar, Y. Huang. *Discovering Spatial Co-location Patterns: A Summary of Results.* Lecture Notes in Computer Science. Springer International Publishing, 2001.

[19] P. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[20] S. Shekhar, H. Xiong. *Encyclopedia of GIS.* Springer, 2008.

[21] J. Wang, W. Hsu, M. L. Lee. A framework for mining topological patterns in spatio-temporal databases. *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 429–436, New York, NY, USA, 2005. ACM. `doi:10.1145/1099554.1099680`.

[22] H. Cao, N. Mamoulis, D. W. Cheung. Discovery of collocation episodes in spatiotemporal data. *Proceedings of the 6th International Conference on Data Mining*, ICDM '06, pages 823–827, Washington, DC, USA, 2006. IEEE Computer Society. `doi:10.1109/ICDM.2006.59`.

[23] M. Celik, S. Shekhar, J. P. Rogers, J. A. Shine, J. S. Yoo. Mixed-drove spatio-temporal co-occurence pattern mining: A summary of results. *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 119–128, Washington, DC, USA, 2006. IEEE Computer Society. `doi:10.1109/ICDM.2006.112`.

[24] H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, J. S. Yoo. A framework for discovering co-location patterns in data sets with extended spatial objects. *SDM*, 2004.

[25] H. Yang, S. Parthasarathy, S. Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. *Patterns in Scientific Data, ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 716–721, 2005.

[26] D. Patel. Interval-orientation patterns in spatio-temporal databases. P. Bringas, A. Hameurlain, G. Quirchmayr, editors, *Database and Expert Systems Applications*, Volume 6261 series *Lecture Notes in Computer Science*, pages 416–431. Springer Berlin Heidelberg, 2010. `doi:10.1007/978-3-642-15364-8_35`.

[27] Y. Huang, S. Shekhar, H. Xiong. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1472–1485, 2004.

[28] P.J. Taylor. *Quantitative Methods in Geography: An Introduction to Spatial Analysis.* Houghton Mifflin, 1977.

[29] C. D. Manning, H. Schütze. *Foundations of statistical natural language processing.* MIT Press, Cambridge, MA, USA, 1999.

[30] N. Hurlburt, M. Cheung, C. Schrijver, L. Chang, S. Freeland, S. Green, C. Heck, A. Jaffey, A. Kobashi, D. Schiff, J. Serafin, R. Seguin, G. Slater, A. Somani, R. Timmons. Heliophysics event knowledgebase for the solar dynamics observatory and beyond. 2010. `arXiv:arXiv/1008.1291`, `doi:10.1007/s11207-010-9624-2`.

[31] HEK. http://www.lmsal.com/isolsearch, Sep 2013.

[32] K.R. Lang. *The Cambridge Encyclopedia of the Sun.* Cambridge University Press, 2001.

[33] L. Egghe, C. Michel. Strong similarity measures for ordered sets of documents in information retrieval. *Inf. Process. Manage.*, 38(6):823–848, November 2002. `doi:10.1016/S0306-4573(01)00051-6`.

[34] E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Trans. on Knowl. and Data Eng.*, 15(1):57–69, January 2003. `doi:10.1109/TKDE.2003.1161582`.

[35] M. A. Schuh, R. A. Angryk, K. G. Pillai, J. M. Banda, P.C.H Martens. A large-scale solar image dataset with labeled event regions. *Int. Conf. on Image Processing (ICIP)*, 2013.

[36] K. G. Pillai. https://www.cs.montana.edu/~k.ganesanpillai/, Sep 2013.

[37] J. Han, J. Pei. Mining frequent patterns by pattern-growth: methodology and implications. *SIGKDD Explor. Newsl.*, 2(2):14–20, December 2000. `doi:10.1145/380995.381002`.