

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

by

Karthik Ganesan Pillai

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

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

August, 2014

© COPYRIGHT

by

Karthik Ganesan Pillai

2014

All Rights Reserved

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I am in great debt to those who assisted in my graduate studies at Montana State University. This dissertation would have never been possible without their tremendous contributions.

First off, I would like to express my sincerest appreciation for my graduate research advisor, Dr. Rafal A. Angryk. He has been extremely patient with me and supported me throughout the entire process. I am very fortunate to have Dr. Angryk as my graduate research advisor.

Dr. Petrus Martens has been a tremendous asset on my graduate committee. He has provided me with valuable insight, especially with the solar physics related questions. I am thankful to Dr. Steven Saar and Dr. Katharine Reeves at Harvard-Smithsonian Center for Astrophysics for providing their insights on solar physics data analysis.

I am also grateful to Dr. John Sheppard, Dr. John Paxton, Dr. Binhai Zhu, Ms. Jeannette Radcliffe, Ms. Kathryn Hollenback, Ms. Phenocia Bauerle, and Ms. Nathalie Caillud who have all helped me during my time at Montana State University and have all been eager to engage me in conversation regardless of the topic.

I would like to express my gratitude to my family who has always supported and encouraged me.

Funding Acknowledgement

This research was supported by two National Aeronautics and Space Administration (NASA) grant awards, 1) No. NNX09AB03G and 2) No. NNX11AM13A.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 Context and Motivation	1
1.2 Outline.....	3
2. PRELIMINARIES	4
2.1 Spatial Data Types	4
2.2 Spatial Operations	4
2.3 Temporal and Spatiotemporal Operations.....	6
2.4 Co-location Patterns	7
2.5 Spatiotemporal Data Mining.....	8
2.6 Preliminaries for STCOPs Analysis	12
2.6.1 Measures.....	15
2.6.2 More on Co-occurrence Coefficient <i>cce</i>	16
2.6.3 Prevalence of STCOPs	18
2.6.4 Problem Statement	19
3. SPATIOTEMPORAL CO-OCCURRENCE PATTERN DISCOVERY ALGORITHMS	21
3.1 STCOPs-Miner Algorithm	21
3.2 Experimental Evaluation	26
3.2.1 Pattern Instances.....	27
3.2.2 Memory Usage	28
3.2.3 Rules Discovered	30
3.3 FastSTCOPs-Miner	32
3.3.1 Problem Statement	32
3.3.2 Relations Between <i>J</i> and <i>OMAX</i> Measures	33
3.3.3 FastSTCOPs-Miner Algorithm.....	38
3.4 Experimental Evaluation	44
3.4.1 Memory Usage	46
3.4.2 Execution Time	48
3.4.3 Pattern Instances.....	50
3.5 Fast-CP Miner	51
3.5.1 Candidate Pattern Trees	53
3.5.2 STOverlapping Candidates.....	55
3.5.3 Task	56
3.5.4 Algorithm	56
3.5.5 Preprocess.....	59

TABLE OF CONTENTS – CONTINUED

3.5.6 Candidate Generation	60
3.5.7 Mining STCOPs	61
3.6 Experimental Evaluation	63
3.6.1 Memory Usage	66
3.6.2 Execution Time	67
3.6.3 Pattern Instances	69
4. INTERDISCIPLINARY APPLICATION - SOLAR EVENTS CO-OCCURRENCES MINING	71
4.1 Background	71
4.2 Data	75
4.2.1 Solar Event Instance Tracking	76
4.3 Experiment and Discussion	78
4.3.1 Results Discussion	80
5. CONCLUSIONS	83
5.1 Closing Remarks	83
5.2 Ideas for Future Work	84
REFERENCES	86
APPENDICES	93
APPENDIX A: Publications and Permissions	94
APPENDIX B: Spatiotemporal Co-occurrence Patterns and Rules	96

LIST OF TABLES

Table	Page
2.1 Temporal information about event instances of data shown in Fig. 2.3....	14
2.2 Example of <i>tab_instance</i> of $SE_i = \{e_1, e_2\}$ with calculation of <i>cce</i> from data shown in Fig. 2.4.	17
2.3 Measures evaluating spatiotemporal relation Overlap (<i>cce</i>) for pattern instances of size $k \geq 2$ [1]	18
3.1 Instance STOverlapping Transactions.	52
3.2 Event STOverlapping Transactions.....	52
3.3 STCOP Candidates.....	60
4.1 Parameters used to download data from HEK	76
4.2 Attributes extracted from data downloaded from HEK	77
4.3 Counts and representation of solar event instances - Data period 01-01-2012 through 01-31-2012	78
4.4 STCORs derived from <i>size-2</i> patterns (see Table B.2 in appendix B).....	80
4.5 STCORs derived from <i>size-3</i> patterns (see Table B.4 in appendix B).....	82
B.1 <i>Size-2</i> STCOPs.....	97
B.2 STCORs derived from <i>size-2</i> patterns	97
B.3 <i>Size-3</i> STCOPs.....	98
B.4 STCORs derived from <i>size-3</i> patterns	99
B.5 <i>Size-4</i> STCOPs.....	100
B.6 STCORs derived from <i>size-4</i> patterns	101

LIST OF FIGURES

Figure	Page
1.1 The growth of solar data volume (adapted from [2]).	1
1.2 The spatiotemporal evolution of four types of solar events (NASA instrument and time stamp are printed on the top of each image).	2
2.1 Some examples of spatial data types (adapted from [3]).	5
2.2 Two evolving polygons satisfying spatial intersects and temporal rela- tions that are important for our investigation [4].	6
2.3 An example of spatiotemporal data	13
2.4 Example of size-2 co-occurrence of spatiotemporal objects.	17
3.1 STCOPs-Miner Algorithm	22
3.2 Table instances of size-1 (a) and candidate patterns of size-2 (b).	23
3.3 Table instances of size-2 (c) and candidate patterns of size-3 (d).	25
3.4 Comparison of candidate and actual pattern instances generated by anti-monotonic and non anti-monotonic measures in the STCOPs-Miner algorithm.	27
3.5 Memory usage used by candidate table instances for the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.	29
3.6 Number of rules discovered from the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.	31
3.7 FastSTCOPs-Miner Algorithm	39
3.8 (a) Table Instances of size-1 and (b) Candidate Patterns for size-2 STCOPs.	40
3.9 (c) Filtered (T_{OMAX2}) and (d) Refined (TR_{J2}) Table Instances of size-2.	42
3.10 (e) Candidate Patterns of size-3 and (f) Filtered (T_{OMAX3}) and (g) Refined (TR_{J3}) Table Instances of size-3.	43

LIST OF FIGURES – CONTINUED

Figure		Page
3.11	Memory usage used by candidate table instances for the FastSTCOPs-Miner and Naïve STCOPs algorithms.	46
3.12	Comparison of execution time for the FastSTCOPs-Miner and Naïve STCOPs.	49
3.13	Pattern instances found (filtered with OMAX and refined with J) using the FastSTCOPs-Miner and Naïve STCOPs.	50
3.14	Candidate generation.....	55
3.15	Fast-CP Algorithm.....	57
3.16	Table instances of size-1.....	59
3.17	Running illustration of Fast-CP Algorithm.....	63
3.18	Memory usage used by candidate table instances for the Apriori-based and CP-tree algorithm.	65
3.19	Comparison of execution time for the Apriori-based and CP-tree algorithm.	68
3.20	Pattern instances found using Apriori-based and CP-tree algorithm.....	69
4.1	The spatiotemporal evolution of four types of solar events (NASA instrument and time stamp are printed on the top of each image).	79

ABSTRACT

Due to the current rates of data acquisition, the growth of data volumes in nearly all domains of our lives is reaching historic proportions [5], [6], [7]. Spatiotemporal data mining has emerged in recent decades with the main goal focused on developing data-driven mechanisms for the understanding of the spatiotemporal characteristics and patterns occurring in the massive repositories of data. This work focuses on discovering spatiotemporal co-occurrence patterns (STCOPs) from large data sets with evolving regions. Spatiotemporal co-occurrence patterns 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 the STCOPs discovery purposes. Third, computationally and storage efficient algorithms to discover STCOPs are missing. These limitations of existing approaches represent important hurdles while analyzing massive spatiotemporal data sets in several application domains that generate big data, including solar physics, which is an application of our interdisciplinary research.

In this work, 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. We also present and investigate three novel approaches to STCOPs mining. We follow this investigation by applying our algorithm to perform a novel data-driven discovery of STCOPs from solar physics data.

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 [8]. SDO sends 0.55 petabytes of raster data each year [8]. This trend in solar data is anticipated to be pushed even further by the ground-based Daniel K. Inouye Solar Telescope (DKIST, formerly the Advanced Technology Solar Telescope, ATST), which is expected to capture one million images per day and generate three to five PB of data per year [9] starting from year 2016. In Fig. 1.1, we show the volume growth of solar data in recent years [2].

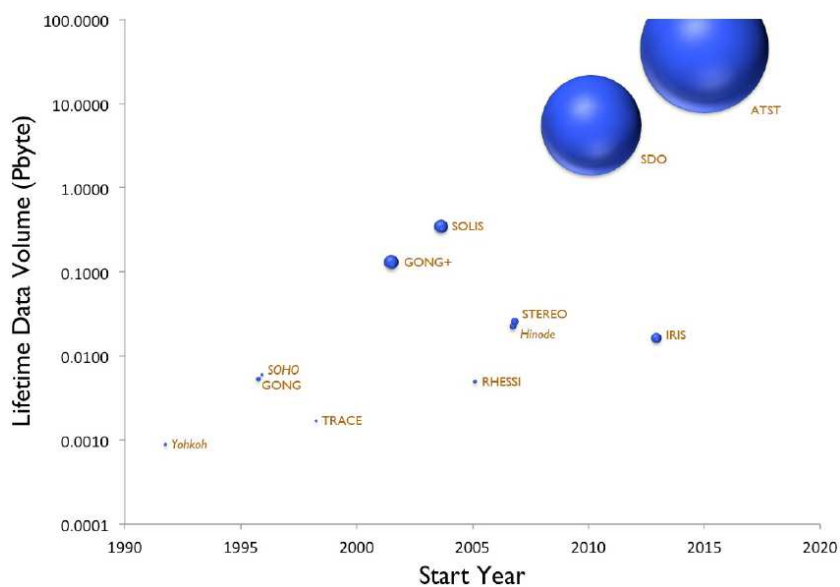


Figure 1.1: The growth of solar data volume (adapted from [2]).

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 [10]), many software modules are working continuously 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 take place among various solar events.

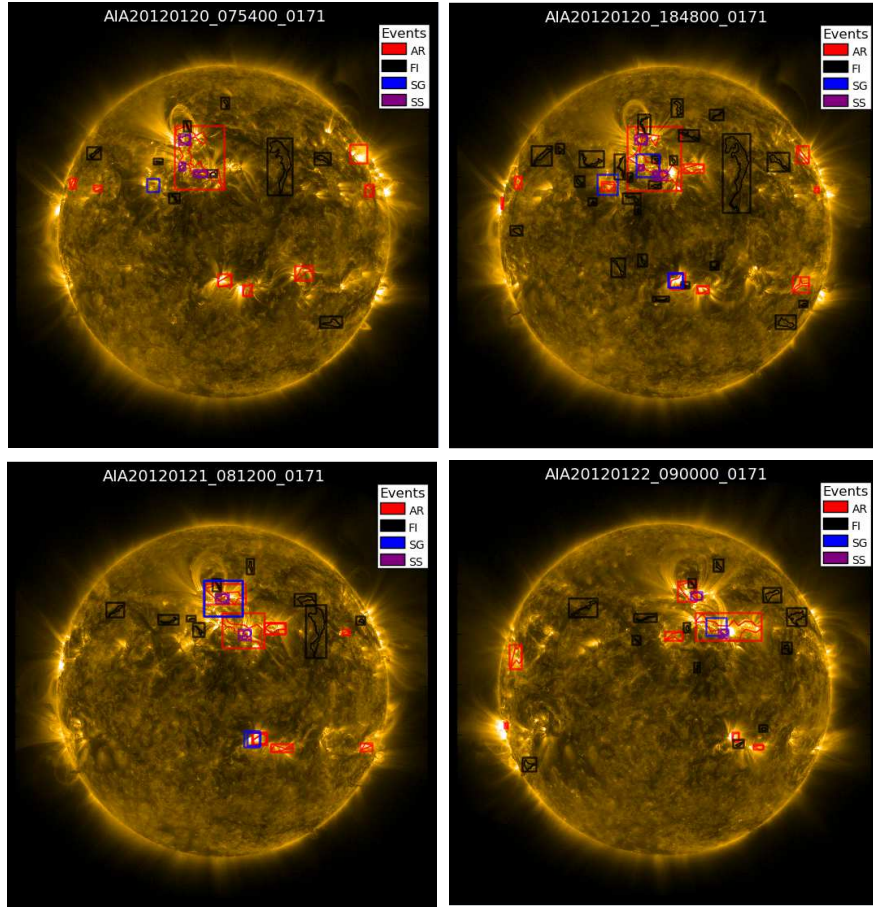


Figure 1.2: The spatiotemporal evolution of four types of solar events (NASA instrument and time stamp are printed on the top of each image).

Fig. 1.2 shows four types of solar events, Active Regions (AR), Filaments (FI), Sigmoids (SG), and Sunspots (SS) in spatial and temporal contexts with their cor-

responding 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 spatiotemporal co-occurrence patterns (STCOPs) representing the subset of different event types that occur together in space and time.

1.2 Outline

The rest of this dissertation is organized as follows. Chapter 2 gives brief background relevant to the research work, gives a literature review of the related works, and provide details on the research we have done regarding modeling of STCOPs for data sets with evolving regions and evaluating these relationships using our interest measures. Chapter 3 introduces the family of our algorithms to mine STCOPs. Chapter 4 presents analysis of solar physics data, and we discuss the patterns and rules discovered. Finally, Chapter 5 contains our conclusions.

CHAPTER 2

PRELIMINARIES

In this chapter, we briefly overview background material relevant to our work. We present important spatial data types, spatial operations, fundamental concepts of temporal relations and spatiotemporal predicates that are necessary to understand our work on spatiotemporal co-occurrence patterns. We also provide details on the research we have done regarding modeling of STCOPs for data sets with evolving regions ([1], [4] [11]).

2.1 Spatial Data Types

Spatial data types are data types used to model geometries and needed to suitably represent geometric data in spatial database systems and Geographical Information Systems [12]. Spatial data types provide an essential abstraction for modeling the geometric structure of objects in space, their relationships, properties, and operations [3]. Some most popular examples of spatial data types are point, line, region; and more complex types such as partitions (in maps), and graphs (in networks). See Fig. 2.1 for some examples of spatial data types [3]. Spatial data types such as regions are often referred to 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 representations [13].

2.2 Spatial Operations

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

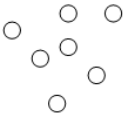
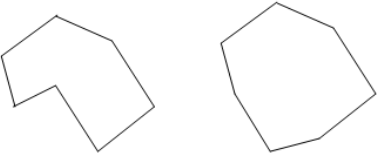
Name	Representation	Description	Example
Point		Location of object in space but not its extent relevant	City Airport Fort
Line and Line String		Movement through space, connections in space	Highway River
Region		Extent of an object relevant	City Lake Forest

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

or Boolean values) or spatial objects. In general *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 [14].

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

In 1983, Allen introduced interval based temporal logic in [15]. The paper also introduced 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”.

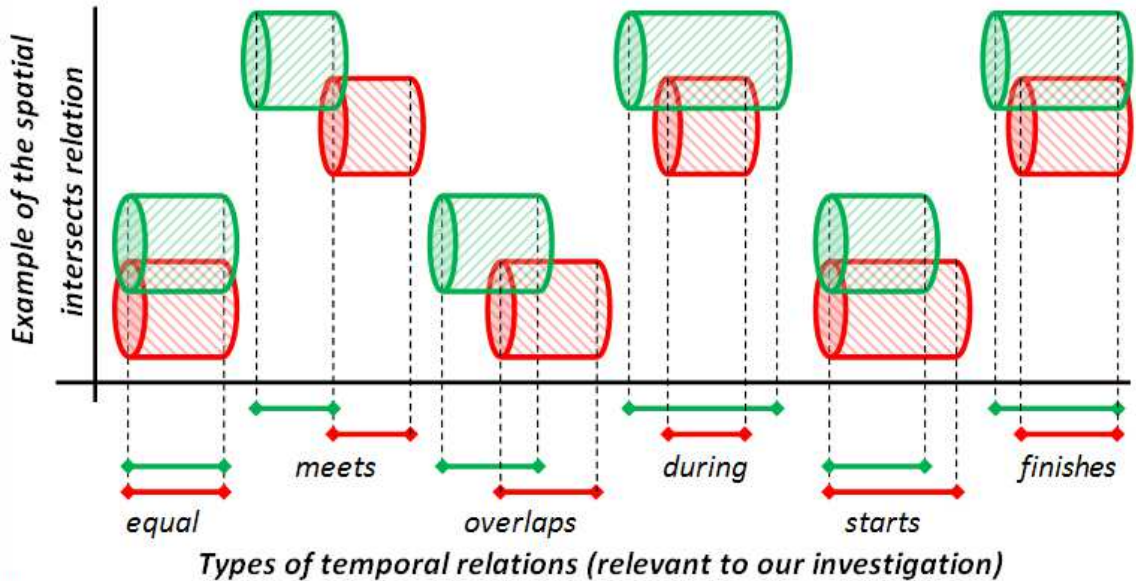


Figure 2.2: Two evolving polygons satisfying spatial intersects and temporal relations that are important for our investigation [4]

Erwig and Schneider [16] 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 lower case. For instance, an evolving spatial region can be represented as a 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 [16], [17] 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 area of growing research. Here we recall a typical notation from the literature [18], motivated by a popular task of shopping baskets analysis. Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals, called items. Let D be a set of transactions (i.e., $D : \mathcal{P}(I)$), where each transaction T is a set of items such that $T \subseteq I$. A frequent pattern/itemset is the subset of all items that occur in our data set of transactions with frequency higher than a given threshold. 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 [18], [19]. An association is a subset of items whose support is above a user specified minimum support [18], [19]. Algorithms like Apriori [18] can find the

frequent itemsets from all the transactions, and association rules can be found from these frequent itemsets/patterns. The Apriori algorithm is a level-wise (iterative) approach that follows a generate-and-test strategy to reduce the number of candidate itemsets [20] in each step/iteration. For example, to discover frequent itemsets of *size-3*, we first discover all itemsets of *size-2* (i.e., level 2) and then try to combine them into *3-itemsets*, and check frequencies of their occurrences.

The spatial co-location problem looks analogous to the classic association rule mining problem from market basket data mining task. However, it is significantly harder than the association rule mining problem because of the lack of transactions and continuous (not binary, as for items) character of the data. In market basket data sets, transactions represent sets of item types bought together by customers [19]. In the spatial co-location rule mining problem, transactions are not often explicit. The transactions in market basket analysis are independent of each other [19]. 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, touch, inside) with each other [19], [21]. In this dissertation 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, some algorithms have been already proposed in the literature for co-location mining in spatiotemporal databases: Topological Pattern Mining [22], Co-location Episodes [23], Mixed Drove

Co-occurrence Mining [24], Spatial Co-location Pattern Mining from extended spatial representations [25], Spatiotemporal Pattern Mining in scientific data [26], and Interval Orientation Patterns [27]. 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 past works for co-location pattern mining in spatial and spatiotemporal databases.

The task of mining topological patterns, also called *co-location patterns*, from spatiotemporal databases was introduced by Wang *et al.* in [22]. 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 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/objects 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, previously introduced in [28]. 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 [23]. 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 point 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 [18] 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 [24]. 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 [28] 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 [24].

Xiong *et al.* introduced the problem of mining spatial co-location patterns from extended spatial representations in [25]. 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 $N(f_1, \dots, f_k)$ for a feature set $E = \{f_1, \dots, f_k\}$ is defined as $\cap_{i=1}^{|E|} N(f_i)$ s.t $f_i \in E$, where $\cap$ symbolizes intersection of $N(f_i)$. 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 that cannot 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 [26]. 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 [27]. In this work, the author modeled features by taking

feature/object 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 [22] 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 [22], [23], [24], [27] or consider features as extended spatial representations [25] with temporal information but do not take feature duration into account [26]. Thus, these methods are not adequate for mining spatiotemporal co-occurrence patterns (STCOPs) on data sets with extended spatial representations that evolve over time.

2.6 Preliminaries for STCOPs Analysis

In this section, we formulate the problem of mining spatiotemporal co-occurring patterns on regions that evolve over time using spatiotemporal predicates to define the evolving regions neighborhoods.

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

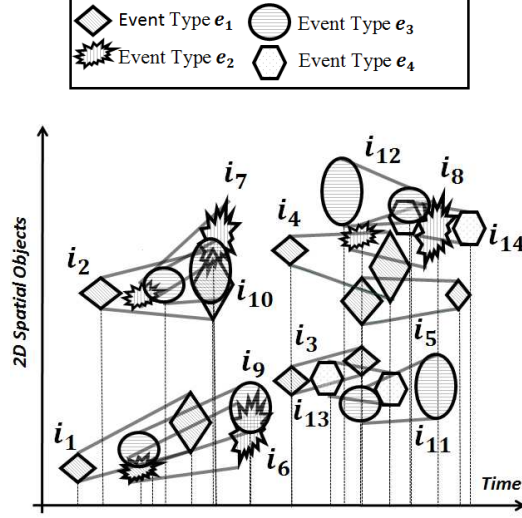


Figure 2.3: An example of spatiotemporal data

In Fig. 2.3, we show an example data set that we will use to explain the definitions in detail. In Table 2.1, 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. 2.3. This data set contains four event types. The event type e_1 has a total of five spatiotemporal instances ($i_1 \dots i_5$), e_2 has three instances ($i_6 \dots i_8$), e_3 has four instances ($i_9 \dots i_{12}$), and e_4 has two instances ($i_{13} \dots i_{14}$). For simplicity, in this example we do not show the sequence of 2D shapes that reflect the spatiotemporal evolution of our data. In 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 2.1).

Definition 1. A *size-(k)* spatiotemporal co-occurrence is denoted as $SE = \{e_1, \dots, e_k\}$, 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 .

Table 2.1: Temporal information about event instances of data shown in Fig. 2.3

<u>Instance ID</u>	<u>Event Type</u>	<u>Start Time</u> (HH:MM)	<u>End Time</u> (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 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. 2.3 and Table 2.1.

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. 2.3 and Table 2.1.

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. 2.3. We will discuss this 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) [28] 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 [28],
3. cp is the conditional probability [28] 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 probability 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 .

2.6.1 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 [29] 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.

2.6.2 More on 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 $J = \frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k)}$. The symbol J represents the *Jaccard* measure [30] (see Fig. 2.4). We use the *Jaccard* measure to capture the spatiotemporal co-occurrence as it is commonly accepted by data mining practitioners [30, 20]. Computing the *cce* for extended spatiotemporal representations such as evolving polygons is not a trivial task. In Fig. 2.4, 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. 2.4. If we assume that instances $\{i_1, i_6\}$, in our example data set (Fig. 2.3 and Table 2.1), 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 2.2 for detailed calculation of *cce*. In Table 2.2, the third column shows time instances (with $\Delta t=10$ 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.

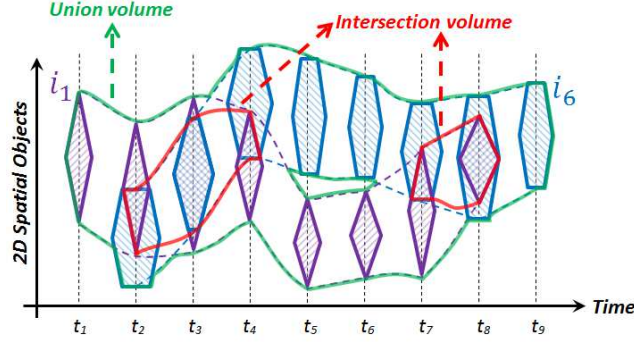


Figure 2.4: Example of size-2 co-occurrence of spatiotemporal objects.

Table 2.2: Example of *tab_instance* of $SE_i = \{e_1, e_2\}$ with calculation of *cce* from data shown in Fig. 2.4.

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

Although, we have shown calculation of our *cce* using the *Jaccard* measure, in this work we investigated alternative measures in detail. We analyze six different measures (i.e. denoted J , $OMAX$, N , D , C , and $OMIN$ in the first column of Table 2.3) to assess the strength of the spatiotemporal relation Overlap. Also, the second column in Table 2.3 gives the formula for each measure for a *size-k pattern instance* where $i_1, \dots, i_k$ denote instances of k spatiotemporal events. Moreover, each $i_j \in i_1, \dots, i_k$ 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 2.3 we specify whether a measure has the anti-monotone property. Each of the measures given in Table 2.3 can be used to assess the strength of spatiotemporal relation Overlap by comparing volumes of the objects in Overlapping trajectories of

Table 2.3: Measures evaluating spatiotemporal relation Overlap (*cce*) for pattern instances of size $k \geq 2$ [1]

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

$i_1, \dots, i_k$. In all the formulas, volumes are represented by the notation V . Note, the measure J shown in Table 2.3 matches the example calculation of *cce* shown earlier in this section.

2.6.3 Prevalence of STCOPs

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

$$\min_{j=1}^k pr(SE_i, e_j) \quad (2.1)$$

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. 2.3 and Table 2.1 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 $pr(\{e_1, e_2, e_3\}, e_3) = 2/2 = 1$. There-

fore 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*.

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,

$$\frac{|\pi_{SE_i}(tab_ins(\{SE_i \cup SE_j\}))|}{|tab_ins(\{SE_i\})|} \quad (2.2)$$

where π is the relational projection operation with duplicate elimination [28].

For example, for co-occurrence rule $e_1 \Rightarrow e_2$, from our data set in Fig. 2.3 and Table 2.1, 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 $(i_7, i_6, \text{ and } i_8)$ of event type e_2 .

2.6.4 Problem Statement

Given:

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

2. A set of N event instances $I = \{i_1, i_2, \dots, i_N\}$, which evolve over time such that $M \ll N$, and each $i_j \in I$ is a tuple $\langle \text{instance-id}, \text{spatiotemporal event type}, \text{sequence of } \langle 2D \text{ shape}, \text{matching time instant} \rangle \text{ pairs} \rangle$, where the sequence of 2D shape and matching time instant pairs reflects the evolution of the spatiotemporal 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 result set of spatiotemporal co-occurrence patterns and rules satisfying $cce > cce_{th}$, $pi > pi_{th}$, and $cp > cp_{th}$.

CHAPTER 3

SPATIOTEMPORAL CO-OCCURRENCE PATTERN DISCOVERY
ALGORITHMS

In this chapter, we present and investigate three novel algorithms to mine spatiotemporal co-occurrence patterns on data sets that have evolving regions ([1], [4], [11], [31]). 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 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. 3.1 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. 3.2 (a) shows the key columns of table instances of *size-1* for our sample spatiotemporal

Input :

- (1) E = A set of spatiotemporal event types, which can be represented as 2D shapes at each time step.
- (2) I = $\langle \text{instance-id, spatiotemporal event type, sequence of } \langle 2D \text{ shape, matching time instant} \rangle \text{ pairs} \rangle$.
- (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 :

```

1   $k=1, C_1=E, P_1 = E, P_{final} = \emptyset, R_{final} = \emptyset;$ 
2   $T_1 = \text{gen\_loc}(C_1, I, t_s);$ 
3  while ( $P_k \neq \emptyset$ ) {
4     $C_{(k+1)} = \text{gen\_candidate\_coocc}(P_k);$ 
5     $T_{(k+1)} = \text{gen\_tab\_ins\_coocc}(C_{(k+1)}, cce_{th});$ 
6     $P_{(k+1)} = \text{pre\_prune\_coocc}(C_{(k+1)}, pi_{th});$ 
7     $P_{final} = P_{final} \cup P_{(k+1)};$ 
8     $R_{(k+1)} = \text{gen\_rules\_coocc}(P_{(k+1)}, cp_{th});$ 
9     $R_{final} = R_{final} \cup R_{(k+1)};$ 
10    $k = k + 1;$ 
11 }
12 return  $P_{final}, R_{final};$ 

```

Figure 3.1: STCOPs-Miner Algorithm

data set (Fig. 2.3 and Table 2.1). 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. 3.2 (a) for simplicity.

Step 4, i.e., $C_{(k+1)} = \text{gen_candidate_coocc}(P_k)$: We generate candidate STCOPs in this step. We use an Apriori-based [18] 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. 3.2 (b) shows the candidate co-occurrence patterns of $size-2$ for our example spatiotemporal data set (Fig. 2.3 and Table 2.1). However, we generate candidate patterns of $size-(k+1)$ from all $size-k$ patterns for non anti-monotonic measures (see Table 2.3).

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

Step 5, $T_{(k+1)} = \text{gen_tab_ins_coocc}(C_{(k+1)}, cce_{th})$: 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, only 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. 3.3 (c) shows the important columns of *size*-2 table instances for our sample spatiotemporal data set from Fig. 2.3 and Table 2.1. 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. 3.3 (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., $P_{(k+1)} = \text{pre_prune_coocc}(C_{(k+1)}, pi_{th})$: 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. 3.3 (c). As seen from the Fig. 3.3 (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

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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$				$\vdots$	$\vdots$	$\vdots$				$\vdots$	$\vdots$	$\vdots$
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						
$\vdots$	$\vdots$	$\vdots$							$\vdots$	$\vdots$	$\vdots$						
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 3.3: Table instances of size-2 (c) and candidate patterns of size-3 (d).

iteration of the algorithm in Step 4 (see Fig. 3.3 (d)). However, 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 $P_{(k+1)}$ [28] for anti-monotonic measures. However, for non anti-monotonic measures we generate rules that have cp greater than cp_{th} from patterns of $P_{(k+1)}$ that have pi value greater than pi_{th} (note,

this check is necessary 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. 2.2, 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.

3.2 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 2.3 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) [32, 33]. The solar event types are *Active Region*, *Filament*, *Sigmoid*, *Sunspot*, *Flare*, and *Emerging Flux* [34]. 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 [35]. We show the ordering

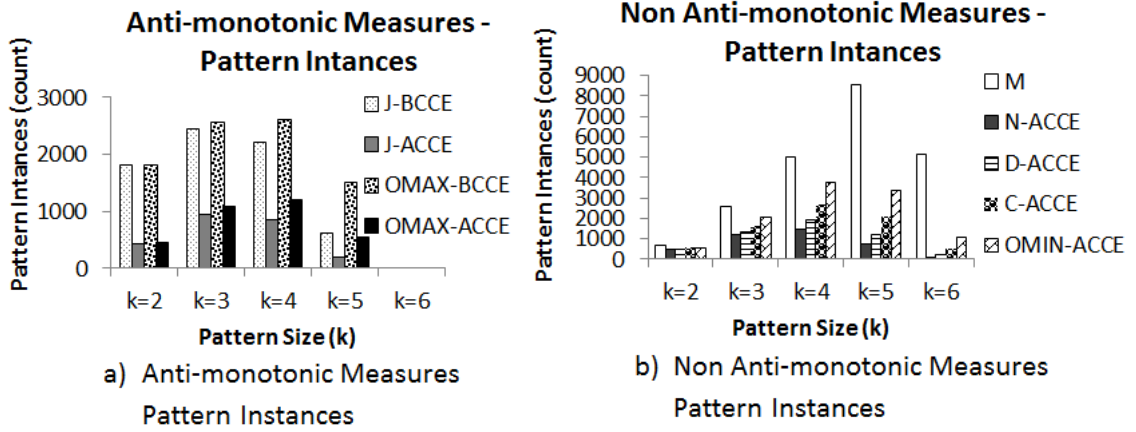


Figure 3.4: Comparison of candidate and actual pattern instances generated by anti-monotonic and non anti-monotonic measures in the STCOPs-Miner algorithm.

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.

3.2.1 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. 3.4 (a), we show the number of pattern instances used by STCOPs-Miner with anti-monotonic measures for different pattern sizes. In Fig. 3.4 (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. 3.4 (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 $OMAX$.

In Fig. 3.4 (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. 3.4 (b), M represents the number of candidate pattern instances generated (i.e., pattern instances that have $V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k) > 0$), 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. 3.4 (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 $N \leq D \leq C \leq OMIN$.

3.2.2 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. 3.5 (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. 3.5 (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. 3.5 (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. 3.5 (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 2.3). However, when the number of candidate pattern instances increases, the measure $OMAX$ uses more memory (see patterns of *size-5* in Fig. 3.5 (a)).

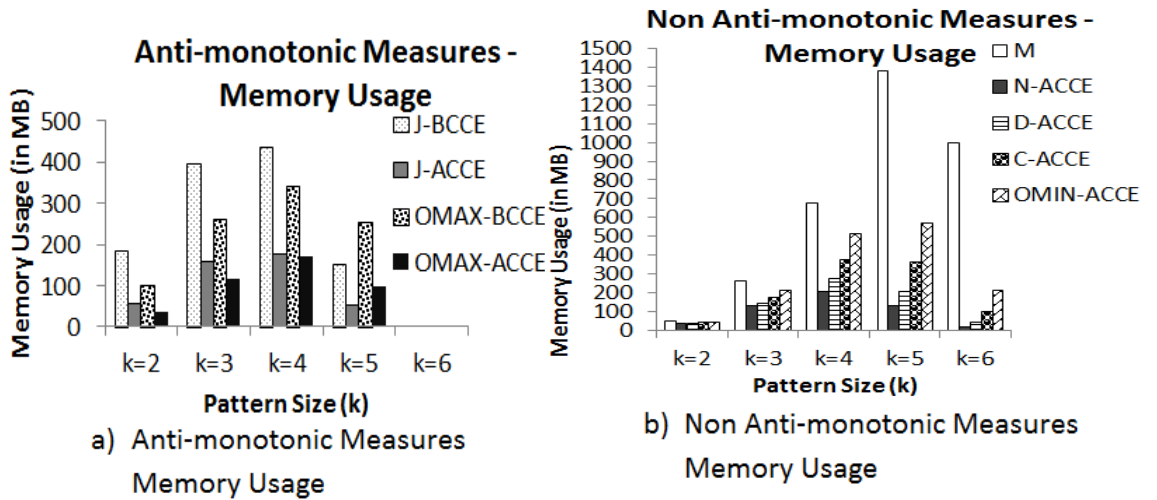


Figure 3.5: Memory usage used by candidate table instances for the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.

In Fig. 3.5 (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. 3.5 (b), M represents the memory usage of table instances for all the pattern instances generated (i.e., pattern instances that have $V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k) > 0$), 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 3.5 (a) and (b)).

3.2.3 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. 3.6. We can observe from Fig. 3.6 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.

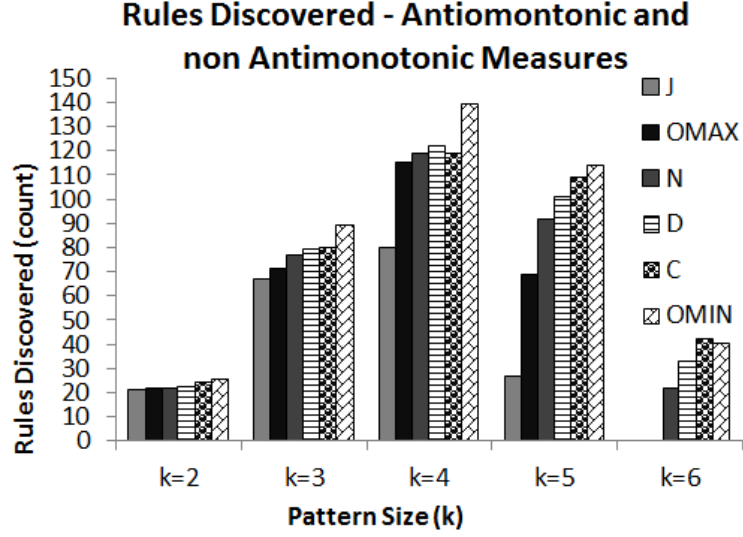


Figure 3.6: Number of rules discovered from the STCOPs-Miner algorithm using anti-monotonic and non anti-monotonic measures.

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 [30]; 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 in the area and short-lasting. Similarly, the measures $OMAX$ and N also penalize objects with smaller Intersection (common) volume. The measure $OMIN$ [30] 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 [30] 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).

3.3 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 irrelevant 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, 20, 29]. 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 [1], [4] 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.

3.3.1 Problem Statement

Input:

1. A set of spatiotemporal event types $E = \{e_1, e_2, \dots, e_M\}$ over a common spatiotemporal framework.
2. A set of N spatiotemporal event instances $I = \{i_1, i_2, \dots, i_N\}$, where each $i_j \in I$ is a tuple $\langle \text{instance-id}, \text{spatiotemporal event type}, \text{sequence of } \langle 2D \text{ shape}, \text{matching time instant} \rangle \text{ pairs} \rangle$, 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 result set of spatiotemporal co-occurrence patterns (STCOPs) satisfying $cce > cce_{th}$ and $pi > pi_{th}$.

3.3.2 Relations Between J and $OMAX$ Measures

As discussed in Sec. 2.6.2 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 $J = \frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k)}$.

We would like to point out 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 chose an alternate measure $OMAX$, defined as $\frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{\max(V(i_1), \dots, V(i_k))}$, 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 cannot 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 [36] 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:

$$\frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k)} \quad (3.1)$$

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 $i_{k+1} \notin \text{pat_instance}$. We claim the measure *J* follows the relation:

$$\frac{V(i_1 \cap i_2, \dots, \cap i_k)}{V(i_1 \cup i_2, \dots, \cup i_k)} \geq \frac{V(i_1 \cap i_2, \dots, \cap i_k \cap i_{k+1})}{V(i_1 \cup i_2, \dots, \cup i_k \cup i_{k+1})} \quad (3.2)$$

Therefore, we need to prove:

$$V(i_1 \cap i_2, \dots, \cap i_k) \geq V(i_1 \cap i_2, \dots, \cap i_k \cap i_{k+1}), \quad (3.3)$$

and

$$V(i_1 \cup i_2, \dots, \cup i_k) \leq V(i_1 \cup i_2, \dots, \cup i_k \cup i_{k+1}) \quad (3.4)$$

Since, adding one more instance of a different event type to a pattern instance can either reduce or not affect the volume of the Intersection of instance trajectories, we obtain the relation $V(i_1 \cap i_2, \dots, \cap i_k) \geq V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k \cap i_{k+1})$ from Eq. 3.3. Similarly, adding one more instance of a different event type to a pattern instance can either increase or not affect the volume of the Union of instance trajectories, we obtain the relation $V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k) \leq V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k \cup i_{k+1})$ in Eq. 3.4. Thus, our relation in Eq. 3.2 holds for all positive real numbers that represent volumes of spatiotemporal objects with evolving polygons $\square$.

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:

$$\frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{\max(V(i_1), \dots, V(i_k))} \quad (3.5)$$

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 $i_{k+1} \notin \text{pat_instance}$. We

claim the measure $OMAX$ follows the relation:

$$\frac{V(i_1 \cap i_2, \dots, \cap i_k)}{\max(V(i_1), \dots, V(i_k))} \geq \frac{V(i_1 \cap i_2, \dots, \cap i_{k+1})}{\max(V(i_1), \dots, V(i_{k+1}))} \quad (3.6)$$

Therefore, we need to prove:

$$V(i_1 \cap i_2, \dots, \cap i_k) \geq V(i_1 \cap i_2, \dots, \cap i_k \cap i_{k+1}) \quad (3.7)$$

and

$$\max(V(i_1), \dots, V(i_k)) \leq \max(V(i_1), \dots, V(i_{k+1})) \quad (3.8)$$

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. 3.7. Similarly, adding another instance of a different event type to a pattern instance cannot reduce the maximum volume of instance trajectories, so we obtain the relation $\max(V(i_1), \dots, V(i_k)) \leq \max(V(i_1), \dots, V(i_k), V(i_{k+1}))$. Thus, Eq. 3.6 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 $\frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k)} \leq \frac{V(i_1 \cap i_2, \dots, i_{k-1} \cap i_k)}{\max(V(i_1), \dots, V(i_k))}$, $\forall V \in R^+$ for $k \geq 2$.

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

$$\max(V(i_1), \dots, V(i_k)) \leq V(i_1 \cup i_2, \dots, i_{k-1} \cup i_k) \quad (3.9)$$

The maximum volume of all trajectories is always less than or equal to the volume of the 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,

$$\min_{j=1}^k pr_J(SE_i, e_j) \leq \min_{j=1}^k pr_{OMAX}(SE_i, e_j) \quad (3.10)$$

$$pi_J(SE_i) \leq pi_{OMAX}(SE_i) \quad \square \quad (3.11)$$

Since participation index is anti-monotonic as the size of the pattern increases [28], and from *Lemmas 3.1* and *3.2*, and from Eq. 3.11, we get

$$STCOP_J \subseteq STCOP_{OMAX} \quad \square.$$

3.3.3 FastSTCOPs-Miner Algorithm

In this section, we introduce the FastSTCOPs-Miner algorithm, which is more efficient than our naïve STCOPs algorithm [1], [4] 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. 3.3), to filter out candidate patterns that cannot 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 [18], are used to generate candidate STCOPs in the next iteration of the algorithm. Thus, the FastSTCOPs-Miner algorithm continuously uses a filter-and-refine strategy at each iteration of the algorithm to generate prevalent STCOPs. Even though such duplication of effort may seem unnecessary, in Sec. 3.4. 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. 3.7), and explain the algorithm with a running example using the data set already shown in Fig. 2.3 and Table 2.1.

For our FastSTCOPs-Miner algorithm shown in Fig. 3.7, the inputs and outputs are defined as in Sec. 3.3. Steps 1 and 2 of proposed algorithm initialize the data parameters and data structures, steps 3 through 11 give an iterative process to discover

Inputs :

See Sec. 3.3.

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 :

```

1   $k=1, C_k=E, PR_{Jk} = E; PR_{final} = \emptyset;$ 
2   $TR_{Jk} = gen\_loc(C_k, I, \Delta t);$ 
3  while ( $PR_{Jk} \neq \emptyset$ ) {
4     $CR_{J(k+1)} = gen\_candidate\_coocc(PR_{Jk});$ 
5     $T_{OMAX(k+1)} = gen\_tab\_ins\_coocc\_filtered(CR_{J(k+1)}, TR_{Jk}, cce_{th});$ 
6     $P_{OMAX(k+1)} = pre\_prune\_coocc\_filtered(CR_{J(k+1)}, T_{OMAX(k+1)}, p_{ith});$ 
7     $TR_{J(k+1)} = gen\_tab\_ins\_coocc\_refined(P_{OMAX(k+1)}, T_{OMAX(k+1)}, cce_{th});$ 
8     $PR_{J(k+1)} = pre\_prune\_coocc\_refined(P_{OMAX(k+1)}, TR_{J(k+1)}, p_{ith});$ 
9     $PR_{final} = PR_{final} \cup PR_{J(k+1)};$ 
10    $k = k + 1;$ 
11 }
12 return  $PR_{final};$ 

```

Figure 3.7: FastSTCOPs-Miner Algorithm

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. 3.8 (a) shows the key columns of table instances of size-1 for our sample spatiotemporal data set (Fig. 2.3 and Table 2.1). 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. 3.8 (a) for simplicity.

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 3.8: (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_{J_k} in Fig. 3.7). 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. 3.7).

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 the lemmas in Sec. 3.3. that they also cannot satisfy cce_{th} requirement for *J* measure.

For example, Fig. 3.9 (c) shows the important columns of *size-2* filtered table instances for our sample spatiotemporal data set from Fig. 2.3 and Table 2.1. 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. 3.9 (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. 3.10 (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 $pi < pi_{th}$.

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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$				$\vdots$	$\vdots$	$\vdots$				$\vdots$	$\vdots$	$\vdots$
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						
$\vdots$	$\vdots$	$\vdots$							$\vdots$	$\vdots$	$\vdots$						
i_4	i_8	11:40							i_8	i_{12}	11:40						
0.60									0.75								

d) Refined 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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	i_7	i_{10}	10:20			
i_2	i_7	10:40	i_2	i_{10}	10:40	$\vdots$	$\vdots$	$\vdots$			
						i_7	i_{10}	10:40			
0.40			0.40			0.50					

Figure 3.9: (c) Filtered (T_{OMAX2}) and (d) Refined (TR_{J2}) Table Instances of size-2.

For example, we show the pi value (See Def. 7) at the end of each table instance in Fig. 3.9 (c). As seen from the Fig. 3.9 (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. 3.10 (f). As seen from the Fig. 3.10 (f), the pattern $SE_i = \{e_1e_2e_3\}$ is a prevalent pattern if a value of 0.39 is set to pi_{th} .

e) Refined Candidate Patterns for k=3

Refined 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
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_1	i_6	i_9	10:50
i_2	i_7	i_{10}	10:00
$\vdots$	$\vdots$	$\vdots$	$\vdots$
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_2	i_7	i_{10}	10:40
0.20			

Figure 3.10: (e) Candidate Patterns of size-3 and (f) Filtered (T_{OMAX3}) and (g) Refined (TR_{J3}) Table Instances of size-3.

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, 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 the 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. 3.9 (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 table instances shown in Fig. 3.9 (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. 3.9 (c) and (d) to compare).

As another example, in Fig. 3.10 (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. 3.10 (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. 3.10 (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. 3.10 (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 satisfy the threshold $pi_{th} = 0.39$ in our example data set shown in Fig. 2.3 and Table 2.1, the algorithm would terminate at $k = 3$ for our running example.

3.4 Experimental Evaluation

In this section, we compare our FastSTCOPs-Miner algorithm against the classic Apriori-based approach [1], [4] 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) [33],[37]. The six different solar event types in our data sets are: *Active Region*, *Filament*, *Sigmoid*, *Sunspot*, *Flare*, and *Emerging Flux* [34].

The artificial data set (denoted Data Set *D*) is generated based on the works of Berkay et al. in [38]. 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. The number of events to be generated is an input parameter to dataset generator, M . We used the artificial data set to investigate the behavior of our algorithms for a larger number of event types. All of our data sets are available on-line [39] to let researchers interested in this topic reproduce our experiments, and maybe even improve on our solution.

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.

3.4.1 Memory Usage

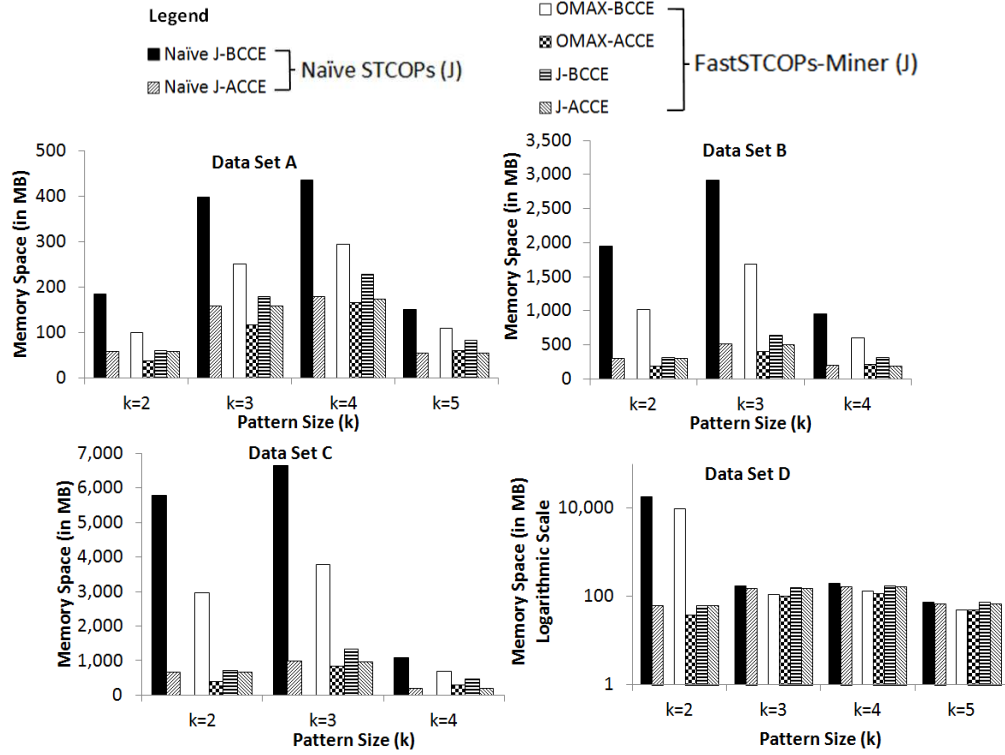


Figure 3.11: Memory usage used by candidate table instances for the FastSTCOPs-Miner and Naïve STCOPs algorithms.

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 generated (see bars in solid colors - black and white in Fig. 3.11), 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. 3.11). In Fig. 3.11 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 interpreted 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. 3.11 the *Jaccard* measure was used heavily 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. 3.11 show the 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, the 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. 3.11 Data Set A). This is because of the fact that to accomplish pruning using the *Jaccard* measure, we have to precompute and store both the Union and Intersection volumes for the co-occurring patterns, while when we use *OMAX* only the 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 the remaining steps of our algorithm. As expected, from Fig. 3.11 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. 3.11). However, generation of all J-BCCE's is necessary to discover actual STCOPs (i.e our J-ACCE's).

Also, in Fig. 3.11 OMAX-BCCE represents the memory usage of table instances for all the pattern instances generated. OMAX-ACCE represents the memory us-

age of table instances after filtering the pattern instances that do not satisfy the threshold cce_{th} in FastSTCOPs-Miner algorithm. This time the 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. 3.11 we can observe a decrease in the memory usage after the pattern instances are filtered by applying the 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).

3.4.2 Execution Time

Next, we show the execution times of our FastSTCOPs-Miner and Naïve STCOPs algorithms. Fig. 3.12 shows the execution time for patterns of different sizes. As expected, our FastSTCOPs-Miner algorithm outperforms the original Naïve STCOPs

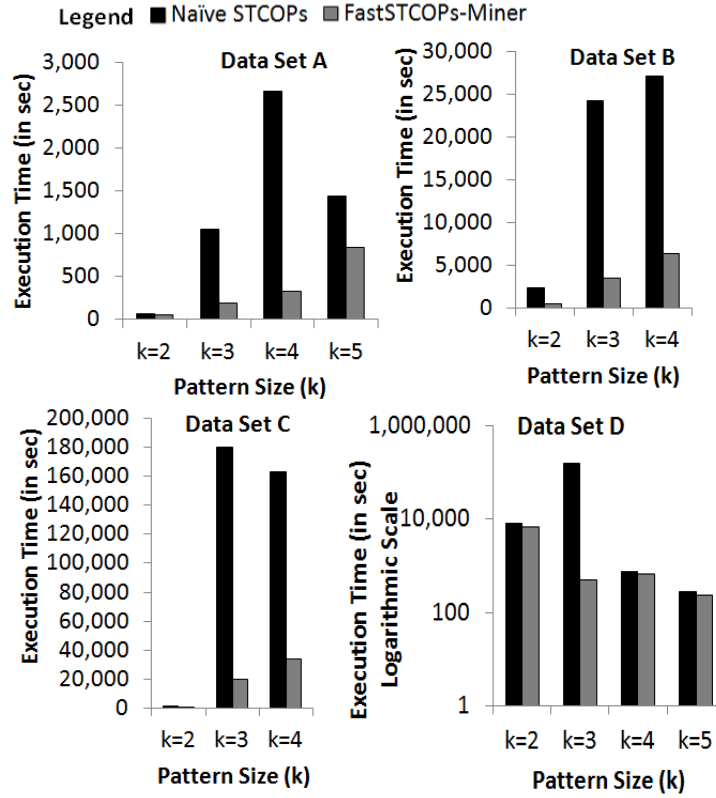


Figure 3.12: Comparison of execution time for the FastSTCOPs-Miner and Naïve STCOPs.

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 J-BCCE in Fig. 3.11 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 a smaller data set (see the bars labelled as J-BCCE in Fig. 3.11). This memory overhead causes the execution time of the Naïve STCOPs algorithm to be slower in comparison to our FastSTCOPs-Miner algorithm.

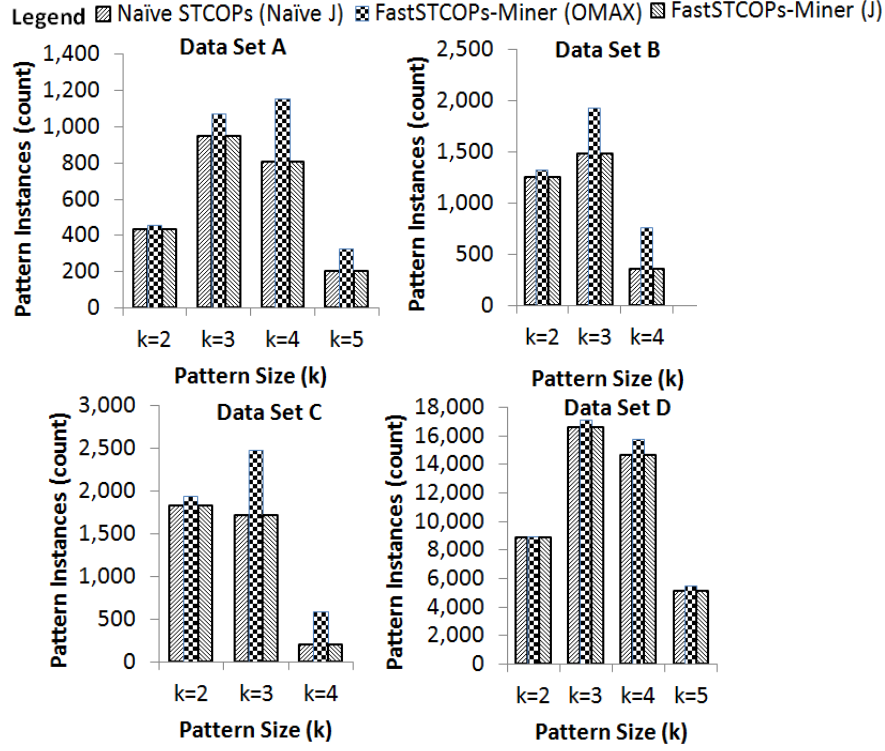


Figure 3.13: Pattern instances found (filtered with OMAX and refined with J) using the FastSTCOPs-Miner and Naïve STCOPs.

3.4.3 Pattern Instances

Fig. 3.13 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 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. 3.13 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. 3.13. The identical results between FastSTCOPs-Miner (J) and

Naïve STCOPs (J) provides evidence for the correctness of our implementation of the FastSTCOPs-Miner algorithm.

3.5 Fast-CP Miner

In this section, we provide details on the research we have done regarding discovering of STCOPs for data sets with evolving regions using a frequent pattern growth approach [31]. We first introduce our new STCOPs mining algorithm, which we call Fast-CP based on a *Candidate Pattern tree (CP-tree)* to mine STCOPs from massive data sets with evolving regions. A *CP-tree* ([40], [41], [42], [43]) is derived from an FP-tree which is the popular data structure for non-spatial association rule mining [44]. In our Fast-CP algorithm [31], we use a *CP-tree* to generate candidate STCOPs using a frequent pattern growth paradigm. Fast-CP uses a subset look-up and a filter-and-refine strategy to prune irrelevant pattern instances of STCOPs based on usage of *all-confidence (OMAX)* measure as a filtering mechanism for *Jaccard*-based analysis which is the standard measure in data mining [4], [20], [29]. Our focus on this work is to compare experimentally the candidate pruning power of our proposed *CP-tree* based Fast-CP algorithm with our Naïve [1], [4] and Fast [11] algorithms. We provide extended experimental results demonstrating the computational and memory efficiency of different algorithms on two real-life data sets and three artificial data sets.

Next, we describe our Fast-CP algorithm [31] based on a frequent pattern growth approach [44], [45], to discover candidate STCOPs using a *CP-tree* from data sets with evolving regions. Similar to the strategy used in [40], [41], [42], [43], and [46] on merging spatial points data, we first transform an input ST data to a set of ST Overlapping (STOverlapping) transactions.

Definition 10. Given a ST instance $i_j \in I$, the STOverlapping transaction of i_j is defined as a set of ST instances $\{i_j, i_l \in I | cce > cce_{th} \wedge i_j's \text{ event type} \neq i_l's \text{ event type}\}$.

Table 3.1: Instance STOverlapping Transactions.

Prefix Instance	Suffix STOverlapping Instances for $cce_{th} = 0.00$	Suffix STOverlapping Instances for $cce_{th} = 0.01$
i_1	i_6, i_9	i_6, i_9
i_2	i_7, i_{10}	i_7, i_{10}
i_3	i_{13}	i_{13}
i_4	i_8	
i_5		
i_6	i_1, i_9	i_1, i_9
i_7	i_2, i_{10}	i_2, i_{10}
i_8	i_4, i_{12}, i_{14}	i_{14}
i_9	i_1, i_6	i_1, i_6
i_{10}	i_2, i_7	i_2, i_7
i_{11}	i_{13}	i_{13}
i_{12}	i_8, i_{14}	
i_{13}	i_3, i_{11}	i_3, i_{11}
i_{14}	i_8, i_{12}	i_8

Table 3.2: Event STOverlapping Transactions.

Prefix Event	Suffix STOverlapping Events for $cce_{th} = 0.01$
e_1	e_2, e_3
e_1	e_2, e_3
e_1	e_4
e_1	
e_1	
e_2	e_1, e_3
e_2	e_1, e_3
e_2	e_4
e_3	e_1, e_2
e_3	e_1, e_2
e_3	e_4
e_3	
e_4	e_1, e_3
e_4	e_2

In Table 3.1, we show the STOverlapping transactions derived from our example data set shown in Fig. 2.3 and Table 2.1 with a threshold $cce_{th} = 0.00$ and

$cce_{th} = 0.01$, respectively. We show transactions with threshold $cce_{th} = 0.00$ to help the reader understand our approach, but will be using the transactions with threshold $cce_{th} = 0.01$ through the rest of this section. This is to keep our running example compact and simpler. In Table 3.1, the STOverlapping transaction of instance i_1 is $\{i_6, i_9\}$. All instances in the set have ST *Overlap* relation with the first instance, i_1 , which is called a prefix instance. STOverlapping transactions can give the upper bound of participation index of an STCOP ([40], [41], [42], [43]). The upper bound participation index of candidate STCOPs is used to prune STCOPs prior to calculating the true participation index of STCOPs. Thus, only the candidate STCOPs whose upper bound participation index satisfies the pi_{th} threshold are used to calculate the true participation index.

Definition 11. Given a ST instance $i_j \in I$, the event STOverlapping transaction of i_j is defined as a set of ST event types: $\{i_j's \text{ event type, and distinct event type of } i_l \in I | cce > cce_{th} \wedge i_j's \text{ event type} \neq i_l's \text{ event type}\}$.

The event STOverlapping transactions can be derived from distinct event types from the instance STOverlapping transactions. In Table 3.2, we show the event STOverlapping transactions derived from the first and third columns in Table 3.1 i.e., from our Instance STOverlapping transactions with $cce_{th} = 0.01$.

3.5.1 Candidate Pattern Trees

A *Candidate Pattern tree (CP-tree)* ([40], [41], [42], [43]) is derived from an FP-tree which is the popular data structure for non-spatial association rules mining [44]. The FP-tree is a prefix-tree structure for storing compressed information about frequent patterns. However, it is hard to represent an input ST data set with evolving regions in one FP-tree while accurately preserving ST *Overlap* relationships between all spatiotemporal event instances. Instead, we construct one tree for each event type for

storing its ST *Overlap* information. Since our trees do not represent specific instances, we can use them only to generate candidates for our pattern mining algorithm more effectively.

Definition 12. A *CP-tree* is a tree structure defined as follows: (1) It consists of one root for the individual prefix event; (2) A set of branches for suffix events that (based on Table 3.2) has ST *Overlap* with the given prefix event (listed in its root); and (3) Each node consists of three fields: event type, count, and children-pointer. Each event type denotes a set of co-occurring events represented by the portion of the path reaching this node from the root. The STOverlapping transactions in the branches reflect instances from Table 3.1 whose prefix event type is the same with the event type of the root node. Children-pointer links to the next node in the tree carrying the same prefix event type.

One *CP-tree* is constructed per event type from its event STOverlapping transactions. For example, in Table 3.2, the transactions that start with e_1 event type are used for building e_1 's *CP-tree*, as shown in Fig. 3.14. In Fig. 3.14 (a) in the first tree we show the *CP-tree* of event type e_1 , which is constructed as follows. The root node is initialized with the prefix event type e_1 . For each event STOverlapping transactions of prefix event type e_1 (rows 1 through 5 in Table 3.2), the root node's count is increased, and the transaction is compared to the prefix tree starting from the root. If there is a match with an already existing path in the tree, the count in the matching nodes is incremented; otherwise, new nodes are added for the items in the suffix of the transaction. Fig. 3.14 shows all *CP-trees* constructed from the event transactions shown in Table 3.2.

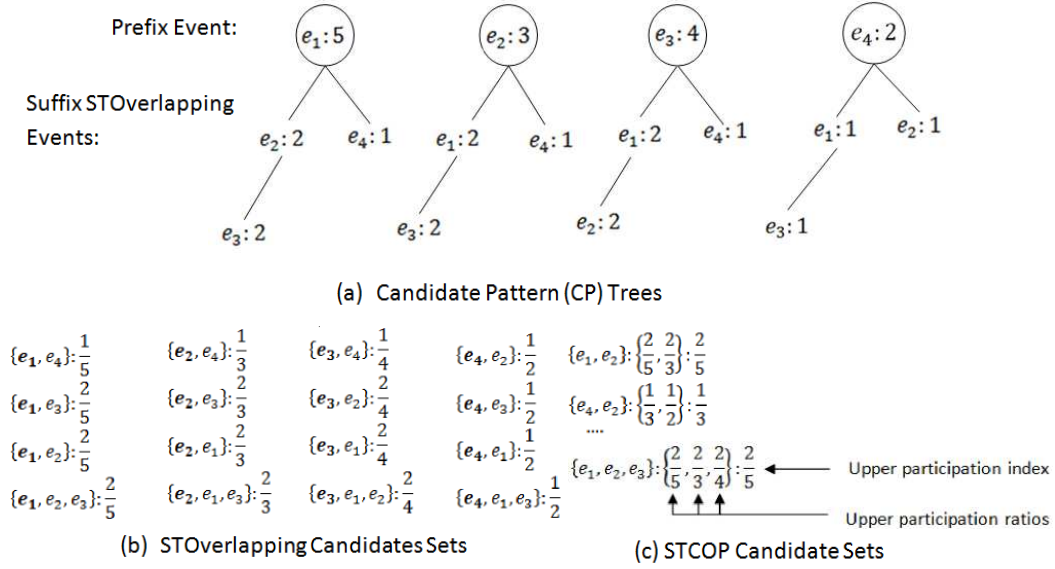


Figure 3.14: Candidate generation

3.5.2 STOverlapping Candidates

We used the FP-growth algorithm for generating STOverlapping candidate event sets from each *CP-tree*. The main difference is the FP-growth algorithm uses its tree for discovery, while *CP-trees* are used only for candidate generation. Fig. 3.14 (b) shows the event sets generated from each *CP-tree*. We call the result sets STOverlapping candidate sets since all elements of a set have a ST *Overlap* relation with its prefix element (i.e., satisfy cce_{th}). The output also gives the frequency with which its prefix event satisfy the cce_{th} value with the suffix events in the set. For example, the support value of $\{e_1, e_2, e_3\}$ shown in Fig. 3.14 (b) represents how many objects of prefix event e_1 satisfy the cce_{th} value with instances of both e_2 and e_3 . In Fig. 3.14 (b) for the pattern $\{e_1, e_2, e_3\}$ we show that two out of five instances of event type e_1 satisfies cce_{th} value with instances of both e_2 and e_3 .

After generating the STOverlapping candidate sets, we combine them for filtering candidates that can have at least one STCOP pattern instance. Fig. 3.14 (c) shows the combined candidate set, which is called an STCOP candidate set. The STCOP

candidate inherits the frequency value of each event from its STOverlapping candidate sets. For example, $\{e_1, e_2, e_3\}$ has the frequency values of e_1 , e_2 , and e_3 events, i.e., $\frac{2}{5}, \frac{2}{3}, \frac{2}{4}$ from each STOverlapping candidate set i.e., $\{\mathbf{e}_1, e_2, e_3\} : \frac{2}{5}$, $\{\mathbf{e}_2, e_1, e_3\} : \frac{2}{3}$, and $\{\mathbf{e}_3, e_1, e_2\} : \frac{2}{4}$. The value represents the upper bound of the chance (i.e., participation ratio) that each event has a ST *Overlap* relationship with other events in our STCOPs. The minimum value of the upper bound participation ratios becomes the upper bound of the participation index of an STCOP.

3.5.3 Task

Input:

1. A set of ST event types $E = \{e_1, e_2, \dots, e_M\}$ over a common ST framework.
2. A set of N ST event instances $I = \{i_1, i_2, \dots, i_N\}$, where each $i_j \in I$ is a tuple $\langle \text{instance-id}, \text{ST event type}, \text{sequence of } \langle 2D \text{ shape}, \text{time instant} \rangle \text{ pairs} \rangle$, where the sequence of 2D shape and matching time instant pairs reflects the evolution of the given ST event.
3. A user-specified ST 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 result set of STCOPs with $cce > cce_{th}$ and $pi > pi_{th}$.

3.5.4 Algorithm

In Fig. 3.15, we give the pseudo code of the proposed Fast-CP algorithm. We also explain the algorithm with a running example using the data set already shown

Inputs :

// See Sec. 3.5.3

Variables :

// See Sec. 3.5.4

- (1) STO_t : a set of all instance STOverlapping transactions (i.e., $cce > cce_{th}$) (Def. 10).
- (2) STO_{et} : a set of all event STOverlapping transactions (Def. 11).
- (3) $Tree_{e_j}$: a *CP-tree* of ST event type e_j (Def. 12)
- (4) C : a set of all candidate patterns for STCOPs derived from STOverlapping candidates of events.
- (5) k the co-occurrence size (Def. 1).
- (6) C_k : a set of candidate patterns for size- (k) STCOPs.
- (7) $T_{OMAX(k)}$: a set of filtered instances of size- (k) ST co-occurrences (Def. 3).
- (8) $TR_{J(k)}$: a set of refined instances of size- (k) ST co-occurrences (Def. 3).
- (9) P_{OMAXk} : the set of *size-k* filtered STCOPs derived from *size-k* candidate STCOPs.
- (10) $PR_{J(k)}$: a set of *size-k* prevalent STCOPs.
- (11) P_{final} : the union of all prevalent STCOPs (patterns of all sizes).// This is the final *Jaccard*-based prevalent patterns.
- (12) M : Number of ST event types.

Algorithm :

// See Sec. 3.5.4

1 $k=1$; $C_k = E$; $PR_{J(k)} = E$; $P_{final} = \emptyset$;
 2 $TR_{J(k)} = gen_loc(C_k, I, \Delta t)$;

Preprocess :

// See Sec. 3.5.5

3 $STO_t = ins_STOverlap_trans(TR_{J(k)}, cce_{th})$;
 4 $STO_{et} = event_STOverlap_trans(STO_t)$;

Candidate generation :

// See Sec. 3.5.6

5 **for** $j=1$ to M **do**
 6 $Tree_{e_j} = build_cp_tree(e_j, STO_{et})$;
 7 **end do**
 8 $C = gen_fp_candidates(Tree_{e_1}, \dots, Tree_{e_M})$;
 9 $C = calculate_upper_pi(C)$;

Mine STCOPs :

// See Sec. 3.5.7

10 $C_{(k+1)} = get_size_k_candidates(C, k+1, pi_{th})$;
 11 **while** ($C_{(k+1)} \neq \emptyset$) {
 12 $T_{OMAX(k+1)} = gen_tab_fil(C_{(k+1)}, TR_{J(k)}, cce_{th})$;
 13 $P_{OMAX(k+1)} = pre_prune_fil(C_{(k+1)}, T_{OMAX(k+1)}, pi_{th})$;
 14 $TR_{J(k+1)} = gen_tab_ref(P_{OMAX(k+1)}, T_{OMAX(k+1)}, cce_{th})$;
 15 $PR_{J(k+1)} = pre_prune_ref(TR_{J(k+1)}, pi_{th})$;
 16 $P_{final} = P_{final} \cup PR_{J(k+1)}$;
 17 $k = k + 1$;
 18 $C_{(k+1)} = get_size_k_candidates(C, k+1, pi_{th})$;
 19 } // end of while
 20 **return** P_{final} ;

Figure 3.15: Fast-CP Algorithm

in Fig. 2.3 and Table 2.1 and the STCOP candidate sets derived from it (see Tables 3.1, 3.2, and Fig. 3.14).

In the Fast-CP algorithm, we use two important pruning strategies to discover relevant pattern instances of STCOPs. In the first strategy, we use a subset look-up method to avoid the expensive candidate table instance generation. For example, before generating a candidate table instance of a *size-3* STCOP, we do a subset look-up on its *size-2* patterns. We generate a candidate table instance of *size-3* pattern only if all of the *size-2* subset patterns satisfy the given pi_{th} value. In the second strategy we use a filter-and-refine method to prune irrelevant pattern instances of STCOPs based on *all-confidence(OMAX)* measure as filtering mechanism for *Jaccard*-based analysis as shown in our recent work [11]. Our *CP-tree* algorithm is actually Apriori in nature (i.e., level-wise); however, we just use the frequent pattern growth paradigm to generate the first set of candidate STCOPs.

For our algorithm, inputs and outputs are defined as in Sec. 3.5.3. **Steps 1** and **2** of proposed algorithm shown in Fig. 3.15, initialize the data parameters and data structures. **Step 3** generates instance STOverlapping transactions (Table 3.1), and **step 4** generates event STOverlapping transactions (Table 3.2). **Steps 5** through **9** generate candidate STCOPs (Fig. 3.15). **Step 10** retrieves *size-k* candidate STCOPs satisfying pi_{th} value. **Steps 11** through **19** gives an iterative process to discover STCOPs. **Step 20** returns the union of all prevalent STCOPs (patterns of all sizes). The explanations of general functions in the algorithm are given next.

In **step 2**, we generate table instances of *size-1*. In this function, argument Δt represents the size of increment in time. The evolution of instances of our ST 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 instance at the particular moment. For

example, Fig. 3.16 shows the key columns of table instances of *size-1* for our sample ST data set (Fig. 2.3 and Table 2.1). 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. 3.16 for simplicity.

$tab_ins(e_1)$		$tab_ins(e_2)$		$tab_ins(e_3)$		$tab_ins(e_4)$	
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						

Figure 3.16: Table instances of size-1.

3.5.5 Preprocess

In **step 3**, we generate instance STOverlapping transactions. This function finds all event instance pairs that have the ST *Overlap* relation satisfying the cce_{th} value. This can be done by a ST join query on the table instances generated in **step 2**. The STOverlapping transactions are generated by grouping the STOverlapping event

instances per each event instance (see Table 3.1). In our algorithm, we first compute the cce value using the $OMAX$ measure and then filter out instance pairs that do not satisfy the threshold cce_{th} . Next we refine these filtered instance pairs by computing the J measure and deleting instance pairs that do not satisfy the threshold cce_{th} for their J value.

In **step 4**, we generate event STOverlapping transactions. In this function we generate event STOverlapping transactions derived from distinct event types in the transactions generated in **step 3** (see Table 3.2).

3.5.6 Candidate Generation

In **steps 5-9**, we generate candidate STCOPs. One CP-tree per event type is constructed from the event STOverlapping transactions of the prefix event in **step 6**. STOverlapping candidates are generated using the FP-growth algorithm [44], and STCOP candidates are generated from the STOverlapping candidates in **step 8**. In **step 9**, the upper bound of STCOP candidates is computed from the minimum value of the support values for events in the pattern. For example, we show the candidate pattern trees, STOverlapping candidates, and STCOP candidates in Fig. 3.14 (a), (b), and (c), respectively. We show the candidate STCOPs of different sizes for our example data set and their corresponding upper bound participation index in the second and fourth column of Table 3.3.

Table 3.3: STCOP Candidates

$k = 2$		$k = 3$	
Candidate	Upper pi	Candidate	Upper pi
e_2e_3	0.50	$e_1e_2e_3$	0.40
e_1e_3	0.40		
e_1e_2	0.40		
e_2e_4	0.33		
e_3e_4	0.25		
e_1e_4	0.20		

In **step 10** of the algorithm, we retrieve candidates of *size-2* satisfying the given pi_{th} value. For example the *size-2* candidate STCOPs upper bound participation index that satisfy pi_{th} value 0.39 are e_2e_3 , e_1e_3 , and e_1e_2 (see first column of Table 3.3). **Steps 11** through **19** give an iterative process to discover STCOPS from the candidate STCOPS found in **steps 5** through **9**.

3.5.7 Mining STCOPS

In **step 12**, we generate filtered table instances for candidate patterns of *size-(k+1)*. In this method we first use a subset look-up strategy to avoid the expensive candidate table instance generation. A *size-(k+1)* ($k + 1 \geq 3$) table instance is generated, only if its *size-k* subset table instances satisfy the given pi_{th} value. Pattern instances for each table instance can be generated by a ST join query (see Fig. 3.17). In this function we calculate the *cce* for each pattern instance by using the *OMAX* measure. Pattern instances that have a *cce* below the user-specified cce_{th} value are deleted from the table instances.

The STOverlapping instances of *size-2* are pattern instances of *size-2*. We show the pattern instances that satisfy the threshold $cce_{th} = 0.01$ of *size-2* STCOPS in Fig. 3.17 (see under the columns $tab_ins(e_2e_3)$, $tab_ins(e_1e_3)$, and $tab_ins(e_1e_2)$). The column denoted by $tab_ins(e_2e_3)$ represents the table instance of *size-2* co-occurrence of events e_2 and e_3 . Similarly, the other columns represent the table instance of different event types. Moreover, we only show the key columns of table instances for simplicity. Although not shown in Fig. 3.17, 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. For example, the table instance $tab_ins(e_2e_3)$ shown in Fig. 3.17, shows the rows $i_6, i_9, 10:20$ through $i_6, i_9, 10:50$ represents a pattern instance that satisfies the threshold $cce_{th} = 0.01$.

We also show the pattern instances that satisfy the threshold $cce_{th} = 0.01$ calculated using the *OMAX* measure for STCOP pattern $e_1e_2e_3$ in Fig. 3.17 (see values, under the Filtered Table Instances table).

Next we discover filtered $size-(k+1)$ STCOPs by pruning candidate patterns in $C_{(k+1)}$ that have $pi < pi_{th}$ in **step 13**. As an example, we show the pi value at the end of Filtered Table Instance $tab_ins(e_1e_2e_3)$ in Fig. 3.17. Here, the pattern $SE_i = \{e_1e_2, e_3\}$ is a prevalent pattern if a value of 0.39 is set to pi_{th} .

In **step 14** we generate 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 12; however, 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 the J measure. Pattern instances that have cce less than the user-specified cce_{th} value are deleted from the table instance.

For example, we show the $size-3$ refined table instances generated from the filtered pattern instances obtained in **step 12** in Fig. 3.17 (Refined Table Instances). In the table instance, 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 $size-3$ refined table instance $tab_ins(e_1e_2e_3)$, the pattern instances that satisfy the $cce_{th} = 0.01$ for J is $\{i_2i_7i_{10}\}$. Note that the pattern instance $\{i_1i_6i_9\}$ is dropped from the refined table instance $tab_ins(e_1e_2e_3)$ as it does not satisfy $cce_{th} = 0.01$ (see Filtered Table Instances and Refined Table Instances in Fig. 3.17 to compare).

After discovering all the refined pattern instances that satisfy cce_{th} threshold, we calculate the refined true participation index in **step 15**. For example, for the pattern $e_1e_2e_3$ the refined true participation index is 0.20 as shown in Fig. 3.17. In **step 16**, we find the union of all the prevalent patterns of $size-(k+1)$, and in **step 17** we increment the k value. **Step 18** retrieves the candidate STCOPs of the next size.

After iteratively discovering all the prevalent patterns, the algorithm returns all the prevalent patterns (i.e., patterns of all sizes) in **step 20** and terminates.

Table instances											
$k = 2$											
$tab_ins(e_2e_3)$			$tab_ins(e_1e_3)$			$tab_ins(e_1e_2)$					
e_2	e_3	timeid	e_1	e_3	timeid	e_1	e_2	timeid			
i_6	i_9	10:20	i_1	i_9	10:00	i_1	i_6	10:00			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
i_6	i_9	10:50	i_1	i_9	10:50	i_1	i_6	10:50			
i_7	i_{10}	10:20	i_2	i_{10}	10:10	i_2	i_7	10:10			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
i_7	i_{10}	10:40	i_2	i_{10}	10:40	i_2	i_7	10:40			
pi=0.50			pi=0.40			pi=0.40					
Filtered Table Instances											
$k = 3$											
$tab_ins(e_1e_2e_3)$				$tab_ins(e_1e_2e_3)$				Refined Table Instances			
e_1	e_2	e_3	timeid	e_1	e_2	e_3	timeid	e_1	e_2	e_3	timeid
i_1	i_6	i_9	10:00	i_2	i_7	i_{10}	10:00	i_2	i_7	i_{10}	10:00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_1	i_6	i_9	10:50	i_2	i_7	i_{10}	10:40	pi=0.20			
i_2	i_7	i_{10}	10:00								
$\vdots$	$\vdots$	$\vdots$	$\vdots$								
i_2	i_7	i_{10}	10:40								
pi=0.40											

Figure 3.17: Running illustration of Fast-CP Algorithm

3.6 Experimental Evaluation

In this section, we compare the candidate pruning power of our Fast-CP algorithm [31] against our Fast algorithm [11] and the classic Apriori-based approach [4]. In our experiments, we are using two real-life data sets from the solar physics domain and three artificial data sets.

In both 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, observed 01/01/2012 - 01/03/2012 (data set A), and 01/01/2012 - 01/05/2012 (data set B). We obtained our data sets from the well-known solar data repository called Heliophysics Event Knowledgebase (HEK) [33],[37].

The first artificial data set (data set C) was generated the using data set generator from [38]. We used this artificial data set to investigate the behavior of our algorithms for large number of different event types. In our experiments, we used nine different event types. The second (data set D) and third (data set E) artificial data sets were created specifically to compare the pruning power of our subset look-up (in addition to our filter-and-refine) strategy in our Fast-CP algorithm with the pure filter-and-refine approach of our Fast algorithm. For the artificial data sets D and E , we changed the pattern generation process of [38]. We first created nine different event types that have different spatiotemporal characteristics. Next, we created pattern instances for 8 co-occurrence patterns of sizes $\{2, \dots, 9\}$ and the event set of *size- k* pattern is a subset of the event set of *size- $(k+1)$* pattern.

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 paper can be found at [39].

We used our three algorithms to capture the STCOPs of the six different solar event types in the real-life data sets, and nine different made-up event types in the artificial data sets. In all of our data sets, instances of different event types are represented as evolving polygons. We compare and report the memory usage of candidate table instances (see Sec. 3.6.1), execution time of the algorithms (see Sec. 3.6.2), and the number of pattern instances found (see Sec. 3.6.3). For the two real-life data sets, for all three algorithms, the $cce_{th} = 0.01$, $pi_{th} = 0.1$, and $\Delta t = 30$ minutes leading to exactly the same set of final STCOPs. For the artificial data set C , for all three algorithms, the $cce_{th} = 0.01$, $pi_{th} = 0.05$, and the sampling time

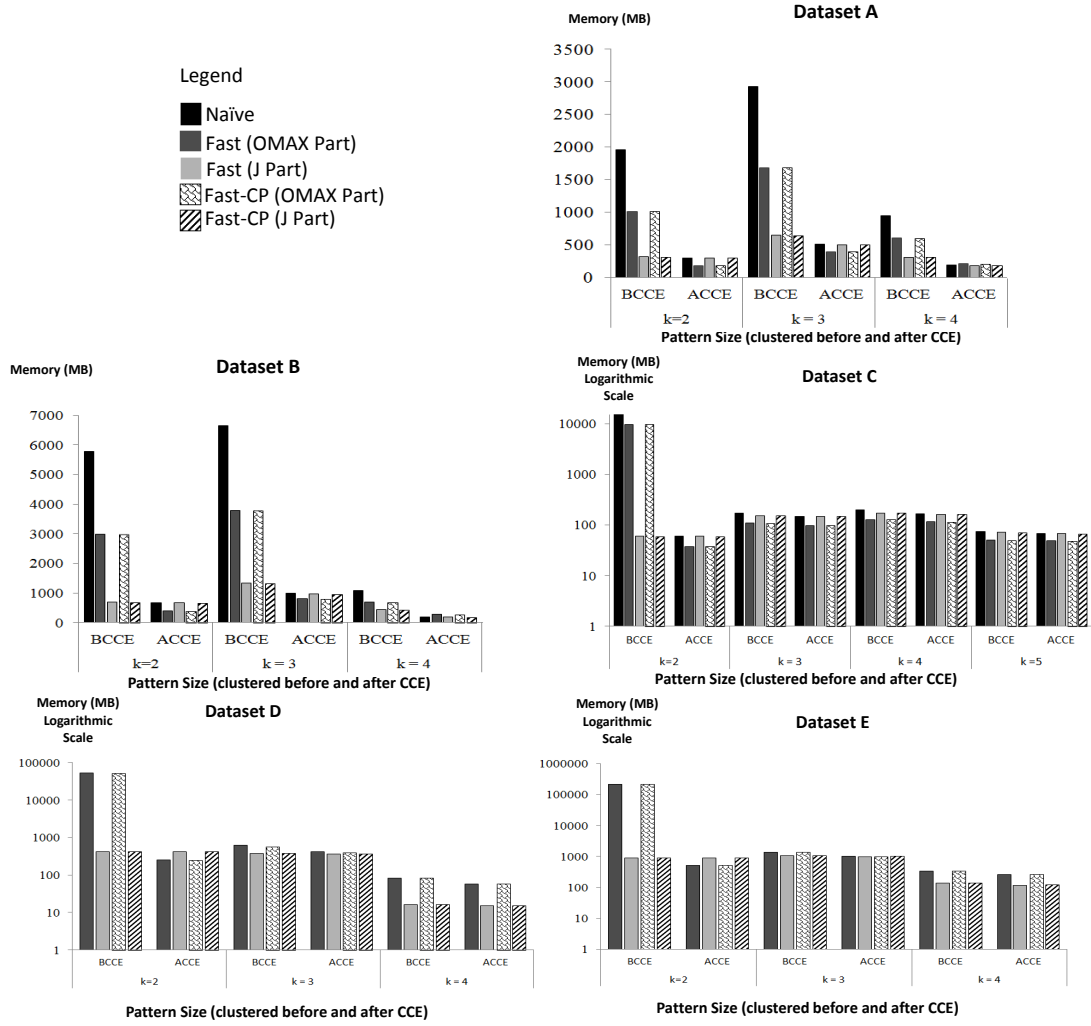


Figure 3.18: Memory usage used by candidate table instances for the Apriori-based and CP-tree algorithm.

interval $\Delta t = 3$ minutes. Since in this case we knew the actual number of generated STCOPs, this data set also served to validate our algorithms. For the artificial data sets *D* and *E*, for the Fast and Fast-CP algorithms, the $cce_{th} = 0.001$, $pi_{th} = 0.3$, and the sampling time interval $\Delta t = 3$ minutes. All experiments were performed using PostgreSQL 9.1.4 and PostGIS 1.5.4.

3.6.1 Memory Usage

We first investigated the memory usage of the Fast-CP algorithm with the Naïve and Fast algorithms for the candidate table instances generated. We report the hard-drive memory usage of candidate table instances with all the pattern instances generated (see bars clustered under BCCE (Before cce_{th} test) in Fig. 3.18), and memory usage of candidate table instances after pruning the pattern instances that do not satisfy threshold cce_{th} (see bars clustered under ACCE (After cce_{th} test) in Fig. 3.18).

In Fig. 3.18 Fast-CP (OMAX Part) clustered in BCCE represents the memory usage of table instances for all the pattern instances generated. As we can see this amount of memory is already smaller than the storage needed for candidates (compare Naïve clustered in BCCE ≈ 1951 MB vs. Fast-CP (OMAX Part) clustered in BCCE ≈ 1012 MB of *size-2* patterns in Fig. 3.18 Data Set A). This is because of the fact that to accomplish pruning with the *Jaccard* measure, we have to pre-compute and store both Union and Intersection volumes for the co-occurring patterns, while when we use *OMAX* only Intersection volumes are needed.

Fast-CP (OMAX Part) clustered in ACCE represents the memory usage of table instances after filtering the pattern instances that do not satisfy the threshold cce_{th} in Fast-CP algorithm. 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}). As expected, from Fig. 3.18 we can observe, that there is a drop in the memory usage after the pattern instances are filtered out by applying the threshold cce_{th} (see and compare Fast-CP (OMAX Part) clustered in BCCE with Fast-CP (OMAX Part) clustered in ACCE in Fig. 3.18). Moreover, Fast-CP (J Part) clustered in BCCE represents the memory usage of table instances for all the pattern instances generated from the

filtered pattern instances (that is from our Fast-CP (OMAX Part) clustered in ACCE that satisfy cce_{th}) and Fast-CP (J Part) clustered in ACCE represents the memory usage of table instances after pruning the pattern instances that do not satisfy the threshold cce_{th} in Fast-CP algorithm.

We can see the effect of our subset look-up strategy used in our Fast-CP algorithm in data sets D and E (see **step 12** in Sec. 3.5.7). For example, from the chart shown for data set E in Fig. 3.18, please compare the memory usage of pattern instances of Fast (OMAX Part) ≈ 346 with Fast-CP (OMAX Part) ≈ 332 clustered in BCCE under $size-3$ patterns. This reduction in memory space is due to the subset look-up method used in our Fast-CP algorithm. In our Fast-CP algorithm we generate table instances of a $size-(k+1)$ pattern that satisfy upper bound participation index, only when all of its $size-k$ subset patterns satisfy the given pi_{th} value. Whereas in our Fast algorithm to generate a table instance we use the Apriori principle. From the data sets D and E we experimentally show that our Fast-CP algorithm prunes pattern instances more effectively than our Fast algorithm.

3.6.2 Execution Time

In this section, we investigate the execution times of the Fast-CP with the Naïve and Fast algorithms for the candidate table instances generated. Fig. 3.19 shows the execution time for patterns of different sizes.

As expected, our Fast-CP and Fast algorithms outperform the original Naïve STCOPs algorithm, since they use 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 labeled as Naïve clustered in BCCE in Fig. 3.18 to realize how much memory overhead this process generates for data sets A , B , and C), while our Fast-CP and Fast

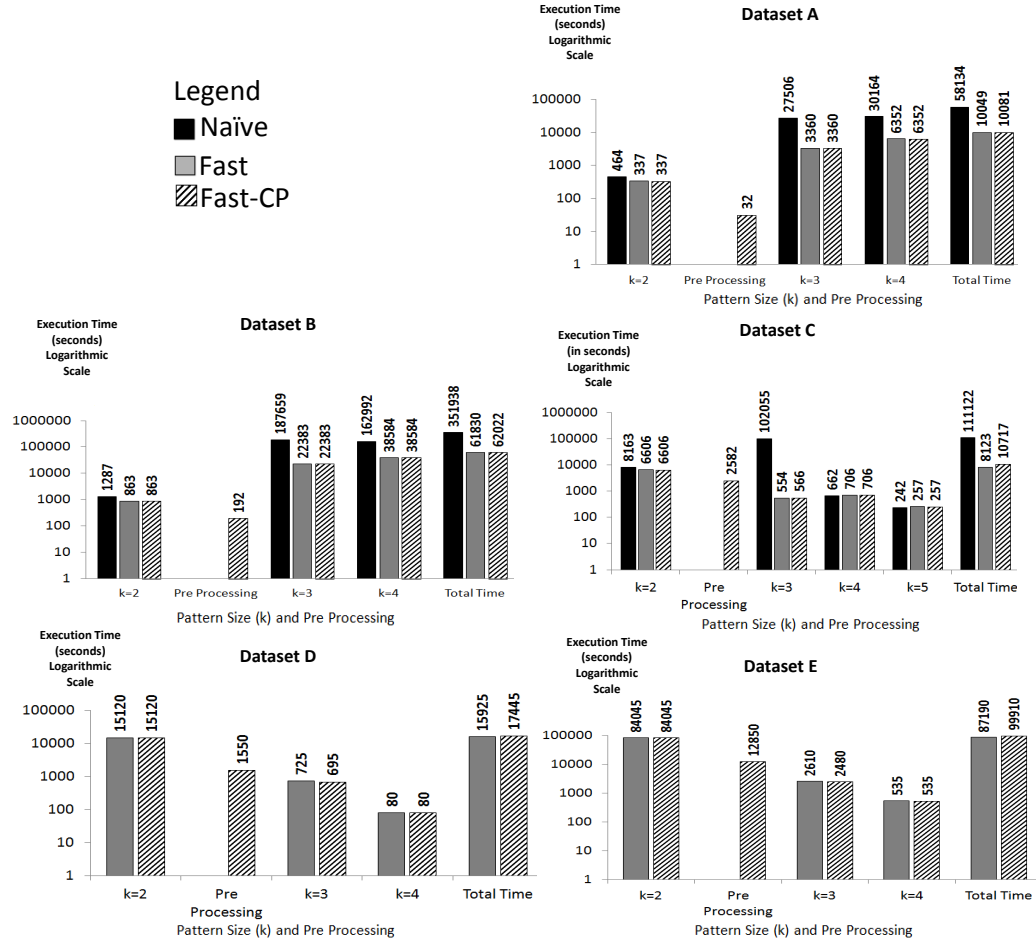


Figure 3.19: Comparison of execution time for the Apriori-based and CP-tree algorithm.

algorithm generates Union geometries for smaller data set (see the bars labeled as Fast-CP (J Part) and Fast (J Part) clustered in BCCE in Fig. 3.18). This memory overhead causes the execution time of the Naïve STCOPs algorithm to be slower in comparison to our Fast-CP and Fast algorithms.

In Fig. 3.19, we show the pre processing (see Sec. 3.5.5) time for generating the candidates through the Fast-CP algorithm separately (clustered in Pre Processing). However, to generate of instances of STOverlapping transactions, we first discover the *size-2* co-occurrences. The pre processing step is an additional overhead for our Fast-CP algorithm. This is more evident from the chart of data sets *D* and *E* in Fig.

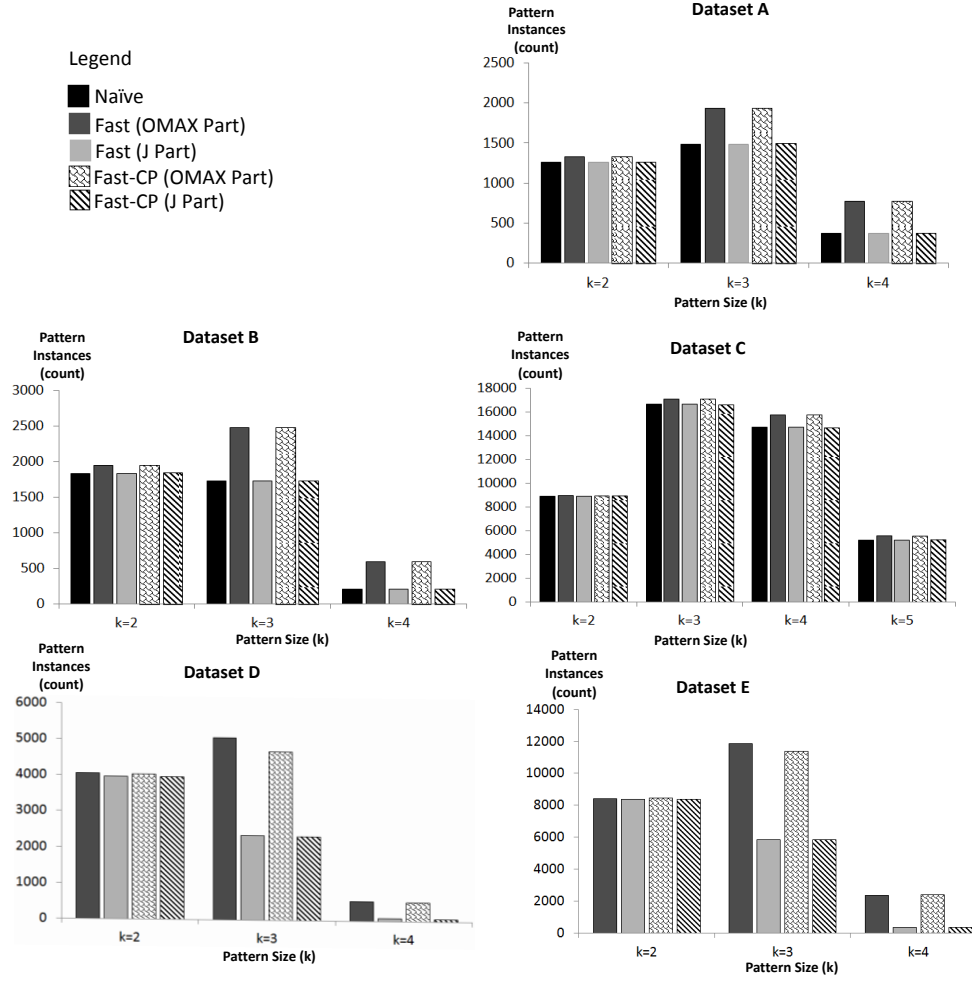


Figure 3.20: Pattern instances found using Apriori-based and CP-tree algorithm.

3.19. Even though the pruning power of our Fast-CP algorithm is better than the Fast algorithm (please compare execution time of *size-3* patterns in charts of data set *D* and *E* in Fig. 3.19). However, the pre processing time outweighs the benefit we get from pruning power of Fast-CP algorithm on these data sets.

3.6.3 Pattern Instances

Fig. 3.20 shows the counts of pattern instances that satisfy the threshold value cce_{th} . The counts of pattern instances shown in Fig. 3.20 help us to verify results and validate the ordering relation of measures *OMAX* and *J* experimentally. We

compare the counts of pattern instances satisfying the threshold cce_{th} with *OMAX* (Fast-CP (OMAX Part)) and *J* (Fast-CP (J Part)) for our Fast-CP algorithm. Our Fast-CP algorithm uses a subset look-up strategy to prune candidates, and the effect of our strategy is shown in the charts of data sets *D* and *E* in Fig. 3.20 (please compare Fast-CP (OMAX Part) with Fast (OMAX Part) of *size-3* patterns). We also compared the counts of pattern instances of the Fast and Naïve algorithms. Moreover, the count of pattern instances that satisfy the threshold value cce_{th} for the Fast-CP is identical to the results of the Naïve and Fast algorithms validating our implementation of the Fast-CP algorithm (please compare Naïve, Fast (J Part), and Fast-CP (J Part) in Fig. 3.20).

CHAPTER 4

INTERDISCIPLINARY APPLICATION - SOLAR EVENTS CO-OCCURRENCES
MINING

In this chapter, we focus on discovering spatiotemporal co-occurrence patterns from solar physics data sets using our FastSTCOPs-Miner algorithm (see Sec. 3.3.3) and the procedure used in our algorithm (see Sec. 3.1) to generate rules from the discovered patterns. We applied our methodology on the newly available metadata to six different types of solar events. These six events are: (1) Active Regions, (2) Coronal Holes, (3) Filaments, (4) Sigmoids, (5) Emerging Fluxes, and (6) Sunspots. The metadata about six solar event types are obtained from National Aeronautics and Space Administration’s (NASA’s) Solar Dynamics Observatory (SDO) and ground-based data by fully automated feature extraction modules to determine patterns and rules between six types of solar events.

We first give background information relevant to the context of our analysis. Next, we describe the data preprocessing steps, and finally we present experimental results and discussion on spatiotemporal co-occurrence patterns and rules discovered.

4.1 Background

The solar data volume generated by NASA’s SDO mission is larger than all the previous missions combined [47]. Solar physicists cannot use the same methods that have been employed for images from previous missions [48] to answer their science questions. The SDO Feature Finding Team (FFT) [48] developed automated software tools during the years 2009–2013 to analyse SDO data and generate the metadata about events that can be used by solar physicists to address their particular science

questions. Below, we briefly characterize six types of phenomena: (1) Active Regions, (2) Coronal Holes, (3) Filaments, (4) Sigmoids, (5) Emerging Fluxes, and (6) Sunspots.

(1) Active Regions on the Sun are regions of locally increased magnetic flux. When a large number of strands of magnetic flux emerge into the visible atmosphere from the solar interior, active regions are formed. The shape and X-ray brightness of the active region depends on both the rate and location of emergence of the new magnetic flux, and the manner in which it reconnects with the pre-existing magnetic fields [49]. Active Regions are the main sources of solar flares.

(2) Regions of low-density plasma on the Sun that have magnetic fields that open freely into interplanetary space are called Coronal Holes. During active periods of solar activity, coronal holes can exist at all solar latitudes; however, during times of low solar activity, coronal holes are primarily seen in the north and south polar caps. Ionized atoms and electrons flow along the open magnetic fields on coronal holes to form the high-speed component of the solar wind [49].

The FFT used the *Spatial Possibilistic Clustering Algorithm* (SPoCA) to detect active regions and coronal holes. The SPoCA algorithm produces segmentation of EUV solar images into regions that are called as Active Regions, Coronal Holes, and the quiet Sun. SPoCA algorithm was chosen by FFT because of its maturity and flexibility [48]. Readers interested in algorithm of SPoCA algorithm, please read [50, 51, 52] for details.

(3) Solar filaments are large, thin and long, strands of very dense, cool plasma, suspended in the hot and thin corona, and they are particularly visible in $H\alpha$ filtergrams [48]. Filament channels (which are the source of filaments) and filaments are known to align with photospheric magnetic polarity inversion lines (PILs) [53] and all major solar eruptions occur above PILs [48]. Moreover, filaments are known

to involve helical magnetic fields, twisted beyond their minimum-energy, current-free, and magnetic configuration [48, 54, 55, 56]. Helical magnetic fields (non potential) are invariably involved in solar eruptions and give rise to coronal mass ejections (CMEs).

Filaments themselves often erupt fully or partially into CMEs, leading to a complete or partial filament disappearance from the solar disk [48, 57, 58, 59]. The axial component of the magnetic field along the filament channel (and along the filament) can have two orientations with respect to the dominant polarities along the polarity inversion zone. The chirality (handedness) is called dextral if the observer, viewing along the filament and its channel in the direction of the axial field, finds the strip of dominantly positive polarity on the right-hand side, otherwise it is called sinistral [60]. If chirality in a filament is known before it disappears, then one has additional clues about the magnetic helicity of the CME that might be useful in assessing the CME's possible geoeffectiveness [48, 61, 62]. Filament eruptions and CMEs are also sources of helicity loss from the Sun, which is an important effect for the magnetic dynamo [63].

The Advanced Automated Filament Detection and Characterization Code (AAFDC) that was developed by the FFT, i) identifies the filament, ii) determines a filament's complete shape, spine, orientation angle, iii) if a filament is broken up into two or more pieces, it correctly identifies them as a single entity, iv) it finds the filaments magnetic chirality (sense of twist), and iv) it tracks them from image to image for as long as they are visible on the solar disk [48]. The code for AAFDC module was originally developed in [64], and it runs daily on an Applied Physics Lab (APL) server and so far has fully automatically generated a database of daily solar-filament properties from July 2000 until the present with only occasional gaps due to lack of $H\alpha$ images for specific days [48].

Readers interested in the algorithm of AAFDCC module, please read [47, 64] for details. The AAFDCC module outputs full-disk $H\alpha$ maps with contours of the identified filaments, location of filament center, filament spine, and barbs, and printed filament ID with chirality (if any is determined). In addition, for each identified filament the module provides a VOEvent entry that contains information about the filament location, outline (chain code), spine skeleton, area and length, number of barbs and in which direction they point, and finally the filament chirality based upon the barb direction analysis [48].

(4) Sigmoids [65] are discrete bright loops that often collectively form sinuous S or inverse-S (i.e., Z) shaped [66]. They are broadly recognized as signatures of unstable coronal magnetic-flux systems as they are considered to be directly or indirectly related to magnetic structures twisted above their lower energy state [48, 67, 68]. Sigmoidal structures are more likely to erupt than non sigmoidal ones, with most (about 84%) triggering solar eruptions within one or two days [66]. These eruptions typically result in the disappearance of the X-ray sigmoid and the launch of a CME that is associated with the twisted/stressed sigmoidal pre-eruption fields [48].

Sigmoids that are fainter and persistent may or may not trigger an eruption, while the brightest and more-transient sigmoids show a very robust association with eruptions [67]. Thus not all CMEs are preceded by sigmoids, and not all sigmoids are the same. However, it has been accepted that an efficient, automatic, sigmoid recognition offers an unbiased way of identifying short-term progenitors for many CMEs [48]. The FFT has produced an automatic pattern recognition “Sigmoid Sniffer” algorithm, first described in [69] to identify sigmoids. Also, a more detailed description of the algorithm can be found in [70]. The Sigmoid Sniffer algorithm has been applied to the images from the X-Ray Telescope [71] on Hinode [72] and SDO/AIA [48] data. In SDO/AIA data, the Sigmoid Sniffer module identifies sig-

moids primarily in high energy channels ($94 \text{ \AA}$ and $131 \text{ \AA}$), which correspond to flaring active regions ($10^{6.8-7.2} \text{ K}$), but also in some low energy channels ($211 \text{ \AA}$ and $335 \text{ \AA}$), which correspond to the active-region corona ($10^{6.3-6.4} \text{ K}$) [48].

(5) Emerging Fluxes are regions on the Sun where new magnetic flux is erupting [34]. The FFT uses the SWAMIS-EF module to identify emerging flux on the Sun [73].

(6) Sunspots are dark, temporary concentrations of strong magnetic field in the Sun’s photosphere. A sunspot appears dark because it is cooler than its surroundings due to the local convective energy transport having been suppressed by the spot’s strong fields. Sunspot duration varies from a few hours to a few weeks, or months for the very longest. The location and number of sunspots depends on the 11-year solar activity cycle. Sunspots usually occur in pairs or groups of opposite magnetic polarity that move in unison across the face of the Sun as it rotates. The leading spot is called the P spot; the following one is termed the F spot [34]. The FFT employs the Sunspot module developed by Zharkov [74].

4.2 Data

The metadata produced by the FFT software modules are stored (primarily) in the Heliophysics Event Knowledgebase (HEK) where they are accessible online for the rest of the world [33]. Metadata about Active Regions, Filaments, Sigmoids, Sunspots, Emerging Flux, and Coronal Holes are downloaded from HEK. Metadata about the solar events can be downloaded manually through official web interface [33]; however, after finding limitations of the original API for our purpose of large-scale data retrieval, we developed our own open source and publicly available software application named “Query HEK”, or simply QHEK (available online [75]). Using

the QHEK software, we downloaded the data for the period 01-01-2012 through 01-31-2012, with spatial region helioprojective, -5000,5000,-5000,5000 which covers the entire Sun. Moreover, in Table 4.1, we list the parameters and their values used to guide our QHEK software while downloading data from Heliophysics Event Knowledgebase (HEK) for the previously listed six different solar event types. As part of the QHEK software, some erroneous records of solar events were deleted during the data cleansing process.

Table 4.1: Parameters used to download data from HEK

	Active Regions	Filaments	Sigmoids	Sunspots	Emerging Flux	Coronal Holes
Instrument	AIA	N/A	AIA_1	HMI	HMI_FRONT2	AIA
Observatory	SDO	N/A	SDO	SDO	SDO/HMI	SDO
Channel ID	N/A	H-alpha	131_THIN	SDO/HMI	LOS Magnetograms	AIA 193
Module Name	SPoCA	AAFDCC	Sigmoid Sniffer	EGSO_SFC	Emerging flux region module	SPoCA

The default HEK web API [33] that QHEK software [75] uses, returns all the associated attributes of an event that are either required or optional. Hence from the downloaded data we extract attributes relevant to our work, and they are shown in Table 4.2. In the first column of Table 4.2 we show the attribute name, and in the second column we show the description of the attributes. We used the Stonyhurst-Heliographic coordinate system to do our analysis. The Stonyhurst-Heliographic coordinates expresses position on the Sun using longitude and latitude on the solar sphere with the origin at the intersection of the solar equator and the central meridian as seen from Earth. This means that the coordinate system remains fixed with respect to Earth while the Sun rotates beneath it [76].

4.2.1 Solar Event Instance Tracking

We use the methodology provided in [77] to track individual instances of each event type downloaded from HEK. The work proposed in [77] addresses the problem of simultaneously tracking multiple objects. Authors define a three stage hierarchical

Table 4.2: Attributes extracted from data downloaded from HEK

Attributes	Description
event_type	Type of event (e.g., Active Region, Filament)
event_starttime	Start time of event
hgs_coord	Center of event in Stonyhurst-Heliographic coordinates
hgs_bbox	Bounding box of event in Stonyhurst-Heliographic coordinates
hgs_chaincode	Contour of event in Stonyhurst-Heliographic coordinates (when available)

tracking framework with the results of the prior stage taken as input to the subsequent stages. The latter two stages formulate the problem as a maximum a posteriori problem. In the first stage of the tracking framework, a simple heuristic is used to link together event instances that have a high probability of being the same event instance at a later time.

For our experiments, differential rotation [78] of the Sun is used to predict an approximate location for our next detection of the same instance at a later time. Next, at the second level stage of the tracking framework, the track fragments obtained from the first stage are associated into longer tracks by formulating the tracking task as a maximum a posteriori problem. We use a successive shortest path algorithm to find the minimum cost flow in a network to solve the maximum a posterior problem [79]. Finally, at the third stage of the tracking framework, the resulting tracks from the previous stage are refined and grouped into longer tracks by simply adapting maximum a posterior parameters. After tracking, the obtained data is interpolated leading to the final tracking result. All our instances of solar events are interpolated in space with respect to time by using the differential rotation of the Sun [78]. Our results were validated by eye in collaboration with an expert in solar physics. [80]. Please note, the primary author of the tracking code is Dustin Kempton, who is a member of our Data Mining Lab. Although, the author of this thesis has been involved in the development of the tracking code this is not considered as his contribution and detailed presentation of the tracking algorithm is out of scope of this thesis.

In the first column of Table 4.3, we show the solar event type; in the second column we show the number of spatial instances of solar events downloaded using the QHEK software; in the third column we show the number of instances of solar events after the tracking (i.e, multiple spatial instances may be merged to represent the evolution of a single spatiotemporal event); and in the last column we show the representation (shape) of spatial instances that we used in our experiment. For example, in Fig. 4.1 we show four types of solar events, Active Regions (AR), Filaments (FI), Sigmoids (SG), and Sunspots (SS) in spatial and temporal contexts with their corresponding shapes (contours) and bounding boxes. Please note, we used bounding boxes for certain event types, because the chain code for those event types are not available in HEK.

Table 4.3: Counts and representation of solar event instances - Data period 01-01-2012 through 01-31-2012

Event Type	HEK Instance Count ¹	Spatiotemporal Instance Count ²	Representation
Active Region	1237	150	Chain Code
Filament	681	388	Chain Code
Sigmoid	303	124	Bounding Box
Sunspot	309	40	Chain Code
Emerging Flux	707	569	Bounding Box
Coronal Hole	774	149	Chain Code

4.3 Experiment and Discussion

We used our approach proposed in [11] and presented in Sec. 3.3.3 of this thesis to discover co-occurring events on the Sun. In addition to discovering the co-occurring events on the Sun, we updated the procedure given in [11] to generate rules from the discovered patterns (please see [1] for rule discovery). In our experiments, the

¹Before Tracking

²After Tracking

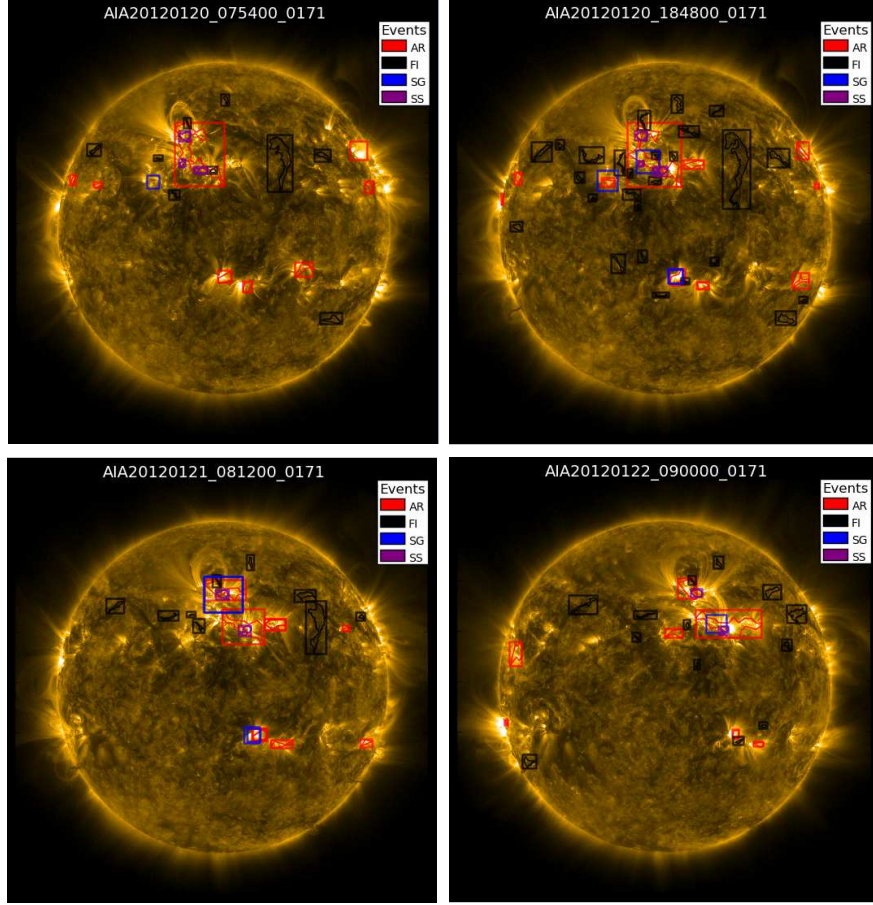


Figure 4.1: The spatiotemporal evolution of four types of solar events (NASA instrument and time stamp are printed on the top of each image).

spatiotemporal event types are the six different solar events (i.e., Active Regions, Coronal Holes, Filaments, Sigmoids, Emerging Fluxes, and Sunspots). The input to our algorithm is the tracked data set created for each solar event type obtained in step 4.2.1. For our experiment, the cce_{th} , pi_{th} , and cp_{th} values were set to 10^{-20} , and the sampling time interval Δt was set at 15 minutes. We chose Δt value based on our discussions with solar physics experts: Dr. Katharine Reeves [81], Dr. Steven Saar [82], Dr. Piet Martens [80], and the values of cce_{th} , pi_{th} , and cp_{th} were chosen so low on purpose to discover as many patterns and rules from the data as possible. All experiments were performed using PostgreSQL 9.1.4 and PostGIS 1.5.4. In this

section we report only hand full of our results, highlighting patterns and rules discovered that show known relations, surprising results, and unexpected results to solar physicists ([80], [81], [82]).

We show the complete results of all the patterns and rules discovered from the tracked solar data (see Sec. 4.2.1) in appendix B, where Table B.1 shows the STCOPs of *size-2* patterns, and Table B.2 shows the spatiotemporal co-occurrence rules (STCORs) derived from *size-2* patterns. In Table B.3, we show the STCOPs of *size-3* patterns, and in Table B.4 we show the STCORs derived from *size-3* patterns. Finally, in Table B.5 we show the STCOPs of *size-4* patterns, and in Table B.6 we show the STCORs derived from *size-4* patterns.

4.3.1 Results Discussion

In Table 4.4, we highlight some of the rules of *size-2* patterns shown in Table B.2, where a rule is denoted as Left Side $\Rightarrow$ Right Side (cp , pi).

Table 4.4: STCORs derived from *size-2* patterns (see Table B.2 in appendix B)

STCORs (Left Side $\Rightarrow$ Right Side)		Conditional Proba- bility (cp)	Participation (pi)	Index
Left Side	Right Side			
Sunspot	Active Region	0.93	0.20	
Sigmoid	Active Region	0.74	0.29	
Sunspot	Sigmoid	0.48	0.21	
Sunspot	Filament	0.28	0.04	
Sigmoid	Filament	0.07	0.02	
Emerging Flux	Sunspot	0.07	0.07	
Emerging Flux	Sigmoid	0.03	0.03	
Emerging Flux	Active Region	0.16	0.16	
Coronal Hole	Active Region	0.01	0.01	

The rules Sunspot $\Rightarrow$ Active Region (0.93, 0.2); Sigmoid $\Rightarrow$ Active Region (0.74, 0.29); Coronal Hole $\Rightarrow$ Active Region (0.01, 0.01) are known to the solar physics community ([80], [81], [82]). Our cp and pi values seem to confirm the known trends. For instance, sunspots often co-occur with active regions, while coronal holes usually

do not occur with active regions, as these two are conflicting events. The rules Emerging Flux $\Rightarrow$ Sunspot (0.07, 0.07); Emerging Flux $\Rightarrow$ Active Region (0.16, 0.16); Emerging Flux $\Rightarrow$ Sigmoid (0.03, 0.03) have lower conditional probability values than expected by the experts. This might be because the SWAMIS-EF module [73] detects a lot of small-scale emerging fluxes that do not result in sunspots, sigmoids, etc. Only large-scale emerging flux do. It is also well known that a sunspot spends more of its life slowly disintegrating than forming (i.e., emerging flux). Moreover, not all emerging flux leads to sigmoids, there are a lot of small emerging flux regions with little effect on anything studied here (e.g., forming the ubiquitous X-ray bright points).

In our earlier work [83], we analyzed the relation between filaments and sigmoids, as well as sigmoid handedness versus filament chirality for different data, and we found a weak correlation for both cases. In particular we found for the rule Sigmoid $\Rightarrow$ Filament the conditional probability was 0.31. This is much higher than the current result Sigmoid $\Rightarrow$ Filament (0.07, 0.02) which we found here. However, in our study [83] we used bounding boxes for Sigmoid and Filament, and we did not do the tracking of HEK event instances. The low conditional probability of the rule Sigmoid $\Rightarrow$ Filament is surprising to solar physics experts. Sigmoids are thought to have filaments running through them by definition ([80], [81], [82]). Possibly filaments in sigmoidal regions have their cool material evaporated and thus are not visible in H-alpha.

Other surprising results that have much higher conditional probability than the expected are the rules Sunspot $\Rightarrow$ Sigmoid (0.48, 0.21), and Sunspot $\Rightarrow$ Filament (0.28, 0.04).

In Table 4.5, we highlight some of the rules of *size-3* patterns shown in Table B.4 of appendix B.

Table 4.5: STCORs derived from *size-3* patterns (see Table B.4 in appendix B)

Rules (Left Side $\Rightarrow$ Right Side)		Conditional Probability (cp)	Participation Index (pi)
Left Side	Right Side		
Sigmoid, Sunspot	ActiveRegion	0.933	0.100
Filament, Sunspot	ActiveRegion	0.882	0.036
Sunspot, EmergingFlux	ActiveRegion	0.731	0.054
Sigmoid, EmergingFlux	ActiveRegion	0.679	0.025
Sigmoid, EmergingFlux	Filament	0.036	0.002
ActiveRegion, Sigmoid	Filament	0.035	0.010
Sigmoid, Sunspot	Filament	0.033	0.003

The rules Sigmoid, Sunspot $\Rightarrow$ ActiveRegion (0.933, 0.100); Filament, Sunspot $\Rightarrow$ ActiveRegion (0.882, 0.036); Sunspot, EmergingFlux $\Rightarrow$ ActiveRegion (0.731, 0.054); and Sigmoid, EmergingFlux $\Rightarrow$ ActiveRegion (0.679, 0.025) are all expected according to the solar physics experts ([80], [81], [82]). Similarly, the rules Sigmoid, EmergingFlux $\Rightarrow$ Filament (0.036, 0.002); ActiveRegion, Sigmoid $\Rightarrow$ Filament (0.035, 0.010); and Sigmoid, Sunspot $\Rightarrow$ Filament (0.033, 0.003) explains that if we can see the filament and it is in active region, there is likely to be a sigmoid. But apparently a lot of time we can have a sigmoid, but the filament (theoretically thought to be there along the neutral line) is not “filled” enough to be visible.

Please note, the *size-4* patterns have very low participation index. We would like to conduct a much large scale data analysis to make definitive statements. The rules derived from *size-4* patterns Filament, Sigmoid, Sunspot $\Rightarrow$ Emerging flux (1.000, 0.002); Filament, Sunspot, Emerging Flux $\Rightarrow$ Sigmoid (1.000, 0.002) are interesting to solar physics experts ([81], [82], [80]).

CHAPTER 5

CONCLUSIONS

5.1 Closing Remarks

With the advancement in technologies to capture high quality data in recent years, there is an explosive growth in the volume of data available to us. It is believed we create about 2.5 quintillion bytes of data every day [7] from various data sources, such as sensors gathering climate information, repositories of high quality images, trajectories of moving objects, and many others [84]. With this volume of data, automation is necessary to analyse the data and discover useful, interesting, and non-trivial information. Automated knowledge discovery from this huge volume of data will help us in informed decision making. Spatiotemporal data mining has emerged in recent decades with one of the main goals focused on developing the understanding of spatiotemporal characteristics and patterns. Our goal of this research is to expand the current understanding of spatiotemporal pattern mining in data sets with evolving regions.

Our research achieved this goal in several ways. First we introduced the problem of mining STCOPs from data sets with extended (region-based) spatial representations that evolve over time. Second we developed a set of novel interest measures. Third we provided a novel framework to model STCOPs. Fourth we developed STCOPs algorithms motivated by the Apriori ([1], [4], [11]) and frequent pattern growth ([31]) approaches and compared them. Fifth we used our algorithms ([11], [1]) to discover spatiotemporal co-occurrence patterns and rules from solar physics data and validated and evaluated them.

Our framework enables us to perform an in-depth study of data sets with evolving regions. To the best of our knowledge, none of the existing methods deal with the problem of mining spatiotemporal co-occurrence patterns and rules from data sets with evolving regions. We explored and evaluated Apriori and frequent pattern growth algorithmic approaches to mine such patterns for evolving extended spatiotemporal data.

5.2 Ideas for Future Work

While our framework has provided to be very useful to discover STCOPs and STCORs from data sets with evolving regions, there are a couple of areas in which it could be improved. The STCOPs mining algorithms presented here require users to provide a proper prevalence threshold value. Using a small prevalence threshold value usually generates many patterns. A large prevalence threshold value can prune a lot of significant interesting patterns, or yield no patterns at all. Moreover, a small set of highly frequent patterns may not be interesting to the user due to the reporting of patterns that are so frequent that they are already well-known to the domain experts. Also, the probability of large STCOPs is inherently smaller than that of co-occurrence of smaller size patterns. Thus, we can argue against using the same prevalence threshold value for STCOPs of all sizes. As a follow up to our work presented here, we have developed approaches that do not require users to provide prevalence thresholds to mine STCOPs [85]. Finally the framework introduced in this work would benefit from the addition of spatiotemporal data indexing strategies related to mining events [86].

REFERENCES

- [1] K. G. Pillai, R. A. Angryk, J. M. Banda, T. Wylie, M. A. Schuh. *Spatiotemporal Co-occurrence Rules*, Volume 241. Springer International Publishing, 2013.
- [2] 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.
- [3] M. Schneider. *Spatial Data Types for Database Systems: Finite Resolution Geometry for Geographic Information Systems*. Lecture Notes in Computer Science. Springer, 1997.
- [4] 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.
- [5] 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]".
- [6] T. Hey, S. Tansley, K. M. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.
- [7] IBM. <http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html>, April/n 2013.
- [8] W. D. Pesnell, B. J. Thompson, P. C. Chamberlin. The Solar Dynamics Observatory (SDO). *Solar Physics*, 275(1-2):3–15, 2012.
- [9] 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. Wiegmann, 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.
- [10] 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]".
- [11] K. G. Pillai, R. A. Angryk, B. Aydin. A filter-and-refine approach to mine spatiotemporal co-occurrences. *Proceedings of the 21th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '13, pages 114–123, New York, NY, USA, 2013. ACM.

- [12] M. Worboys, M. Duckham. *GIS: A Computing Perspective*. CRC Press, Inc., Boca Raton, FL, USA, 2004.
- [13] S. Shekhar, S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2002.
- [14] R.H. Gutting, M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.
- [15] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.
- [16] M. Erwig, M. Schneider. Spatio-temporal predicates. *IEEE Trans. on Knowl. and Data Eng.*, 14(4):881–901, July 2002.
- [17] M. Koubarakis, T. K. Sellis, A. U. Frank, S. Grumbach, R. H. Gutting, 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.
- [18] R. Agrawal, T. Imieliński, A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.
- [19] S. Shekhar, Y. Huang. *Discovering Spatial Co-location Patterns: A Summary of Results*, Volume 2121 series *Lecture Notes in Computer Science*. Springer International Publishing, 2001.
- [20] P. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
- [21] S. Shekhar, H. Xiong. *Encyclopedia of GIS*. Springer, 2008.
- [22] 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.
- [23] 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.
- [24] M. Celik, S. Shekhar, J. P. Rogers, J. A. Shine, J. S. Yoo. Mixed-drove spatio-temporal co-occurrence 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.

- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] P.J. Taylor. *Quantitative Methods in Geography: An Introduction to Spatial Analysis*. Houghton Mifflin, 1977.
- [30] C. D. Manning, H. Schutze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.
- [31] K. G. Pillai, R. A. Angryk, B. Aydin. A frequent pattern growth approach to mine spatiotemporal co-occurrences. *In Preparation*, 2014.
- [32] 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](https://arxiv.org/abs/1008.1291).
- [33] HEK. <http://www.lmsal.com/isolsearch>, June 2014.
- [34] K.R. Lang. *The Cambridge Encyclopedia of the Sun*. Cambridge University Press, 2001.
- [35] L. Egghe, C. Michel. Strong similarity measures for ordered sets of documents in information retrieval. *Inf. Process. Manage.*, 38(6):823–848, November 2002.
- [36] E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Trans. on Knowl. and Data Eng.*, 15(1):57–69, January 2003.
- [37] 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)*, pages 4349–4353, 2013.

- [38] R. A. Angryk B. Aydin, K. G. Pillai. ERMO-DG: Evolving Region Moving Object Dataset Generator. *Proceedings of the 27th International Florida Artificial Intelligence Research Society Conf. (FLAIRS'14)*, pages 321–326, 2014.
- [39] K. G. Pillai. <https://www.cs.montana.edu/~k.ganesanpillai/>, Sep 2013.
- [40] J.S. Yoo, M. Bow. Finding n-most prevalent colocated event sets. *11th International Conference on Data Warehousing and Knowledge Discovery*, pages 415–427, 2009.
- [41] J.S. Yoo, M. Bow. Mining maximal co-located event sets. *15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 351–362, 2011.
- [42] J.S. Yoo, M. Bow. Mining top-k closed co-location patterns. *Proceedings of International Conference on Spatial Data Mining and Geographical Knowledge Services*, pages 100–105, 2011.
- [43] J. Soung Yoo, M. Bow. Mining spatial colocation patterns: a different framework. *Data Min. Knowl. Discov.*, 24(1):159–194, January 2012.
- [44] J. Han, J. Pei. Mining frequent patterns by pattern-growth: methodology and implications. *SIGKDD Explor. Newsl.*, 2(2):14–20, December 2000.
- [45] M. Akbar, R. A. Angryk. Frequent pattern-growth approach for document organization. *Proceedings of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web*, pages 77–82, 2008.
- [46] Liqin. Zhang Yan. Huang, Ping. Yu. Can we apply projection based frequent pattern mining paradigm to spatial co-location mining? *Advances in Knowledge Discovery and Data Mining - 9th Pacific-Asia Conference*, pages 719–725, 2005.
- [47] William Dean; Thompson Barbara Chamberlin, Phillip; Pesnell. The Solar Dynamics Observatory (SDO). *Solar Phys*, 275:1, 2011.
- [48] 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. De Moortel, T. Wiegmann, M.K. Georgoulis, R.T.J. McAteer, R.P. Timmons. Computer Vision for the Solar Dynamics Observatory (SDO), 2010.
- [49] Paul Murdin. *Encyclopedia of Astronomy and Astrophysics*. 2001.
- [50] V. Barra, V. Delouille, J.F. Hochedez, P. Chainais. Segmentation of eit images using fuzzy clustering: a preliminary study. *The Dynamic Sun: Challenges for Theory and Observations*, pages 71–80, 2005.

- [51] V. Barra, V. Delouille, J.F. Hochedez. Segmentation of extreme ultraviolet solar images via multi-channel fuzzy clustering. *Advances in Space Research*, 42, 2008.
- [52] V. Barra, V. Delouille, M. Kretzschmar, J.F. Hochedez. Fast and robust segmentation of solar euV images: algorithm and results for solar cycle 23. *Astronomy and Astrophysics*, 505, 2009.
- [53] S.F Martin. Conditions for the formation and maintenance of filaments (invited review). *Solar Phys*, 182:107–137, 1998.
- [54] S. F. Martin, R. Bilimoria, P. W. Tracadas. *Magnetic field configurations basic to filament channels and filaments*. Kluwer Academic Publishers, Dordrecht, edition solar surface magnetism, 1994.
- [55] D. M. Rust, S. F. Martin. *A Correlation Between Sunspot Whirls and Filament Type*, Volume 68. Astronomical Society of the Pacific (ASP), 1993.
- [56] A.A. Pevtsov, K.S. Balasubramaniam, J.W Rogers. Chirality of chromospheric filaments. *Astrophys*, 595:500–505, 2003.
- [57] H.R. Gilbert, T.E. Holzer, J.T. Burkepile, Hundhausen. Active and eruptive prominences and their relationship to coronal mass ejections. *Astrophys*, 537:503–515, 2000.
- [58] A. Asai, T.T. Ishii, H. Kurokawa, T. Yokoyama, M Shimojo. Evolution of conjugate footpoints inside flare ribbons during a great two-ribbon flare on 2001 april 10. *Astrophys*, 586:624–629, 2003.
- [59] J. Jing, V.B. Yurchyshyn, G. Yang, Y. Xu, H Wang. On the relation between filament eruptions, flares, and coronal mass ejections. *Astrophys*, 614:1054–1062, 2004.
- [60] CORNELIS ZWAAN PETRUS C. MARTENS. Origin and Evolution of Filament-Prominence Systems. *Astrophys*, 558:872–887, 2001.
- [61] V.B. Yurchyshyn, H. Wang, P.R. Goode, Y Deng. Orientation of the magnetic fields in interplanetary flux ropes and solar filaments. *Astrophys*, 563:381–388, 2001.
- [62] D.M. Rust, B.J LaBonte. Observational evidence of the kink instability in solar filament eruptions and sigmoids. *Astrophys. J. Lett*, 622:69–72, 2005.
- [63] A. Brandenburg, W. Dobler. Large scale dynamos with helicity loss through boundaries. *Astron. Astrophys*, Volume 369, pages 329–338, 2001.

- [64] P.N. Bernasconi, D.M. Rust, Hakim. Advanced automated solar filament detection and characterization code: description, performance, and results. *Solar Phys*, 228:97–117, 2005.
- [65] D.M. Rust, A. Kumar. Evidence for helically kinked magnetic flux ropes in solar eruptions. *Astrophys. J. Lett*, 464:199–202, 1996.
- [66] Richard C. Canfield, Hugh S. Hudson, David E. McKenzie. Sigmoidal morphology and eruptive solar activity. *Geophysical Research Letters*, 26(6):627–630, 1999.
- [67] S.E. Gibson, Y. Fan, T. Trk, B. Kliem. The evolving sigmoid: evidence for magnetic flux ropes in the corona before, during, and after cmes. *Space Sci. Rev*, 124:131–144, 2006.
- [68] L.M. Green, B. Kliem, T. Trk, L. van Driel-Gesztelyi, G.D.R. Attrill. Transient coronal sigmoids and rotating erupting flux ropes. *Solar Phys*, 246:365–391, 2007.
- [69] B.J. LaBonte, D.M. Rust, P.N. Bernasconi. An automated system for detecting sigmoids in solar x-ray images. *Bull. Am. Astron. Soc*, 35:814, 2003.
- [70] P.N. Bernasconi, N.E. Raouafi, M.K. Georgoulis. The sigmoid sniffer. *Solar Phys*. submitted.
- [71] L. Golub, E. Deluca, G. Austin, J. Bookbinder, D. and Caldwell. The x-ray telescope (xrt) for the hinode mission. *Solar Phys*, 243:63–86, 2007.
- [72] T. Kosugi, K. Matsuzaki, T. Sakao, T. Shimizu, Y. and Sone. The hinode (solar-b) mission: an overview. *Solar Phys*, 243:3–17, 2007.
- [73] D.A. Lamb, T.A. Howard, C.E. DeForest. Non-Locality of the Small-Scale Dynamo. *In Press Astrophysics*, 2014.
- [74] S.I. Zharkov, V.V. Zharkova. Statistical analysis of the sunspot area and magnetic flux variations in 19962005 extracted from the solar feature catalogue. *Advances in Space Research*, 38(5):868 – 875, 2006.
- [75] QHEK. <https://github.com/dmlabdmlab/queryhek>, June 2014.
- [76] sunpy. http://docs.sunpy.org/en/latest/code_ref/wcs.html, 2014.
- [77] M. Hofmann, M. Haag, G. Rigoll. Unified hierarchical multi-object tracking using global data association. *Performance Evaluation of Tracking and Surveillance (PETS), 2013 IEEE International Workshop on*, pages 22–28, Jan 2013.
- [78] R.F. Howard, J.W. Harvey, S. Forgach. Solar surface velocity fields determined from small magnetic features. *Solar Physics*, 130(1-2):295–311, 1990.

- [79] D.J. Kempton, R.A. Angryk, K.G. Pillai, P. Martens. Iterative refinement multiple hypothesis tracking for solar events. *In preparation*, 2014.
- [80] Piet Martens. <http://solar.physics.montana.edu/martens/>, 2014.
- [81] Katharine Reeves. <http://helio.cfa.harvard.edu/trace/ssxg/kreeves/>, 2014.
- [82] Steven Saar. <http://www.cfa.harvard.edu/contact/hea>, 2014.
- [83] A.R. Yeates P.C. Martens, K. G. Pillai. Hemispheric patterns in filament chirality and sigmoid shape over the solar cycle, (invited review) 2014. *Proceedings of IAU Symposion 300, Nature of solar prominences and their role in Space Weather*, pages 122–125, 2014.
- [84] Shekhar S. R.R. Vatsavai, M. Celik. *Spatial and Spatiotemporal Data Mining: Recent Advances*. AAAI Press, 2008.
- [85] K.G. Pillai, R.A. Angryk, J.M. Banda, B. Aydin, D.J. Kempton, P. Martens. Mining at most top-k% spatiotemporal co-occurrence patterns in data sets with extended spatial representations. *In submission*, 2014.
- [86] B. Aydin, D.J. Kempton, V. Akkineni, S.R. Gopovaram, K.G. Pillai R.A. Angryk. Spatiotemporal indexing techniques for efficiently mining spatiotemporal co-occurrence patterns. *In Review: Proceedings of the 22nd International Conference on Advances in Geographic Information Systems*, 2014.

APPENDICES

APPENDIX A

PUBLICATIONS AND PERMISSIONS

Publications contained in this dissertation are as follows. In most cases this dissertation extends these publications.

- Spatiotemporal Co-occurrence Pattern Mining in Data Sets with Evolving Regions, Karthik Ganesan Pillai, Rafal A. Angryk, Juan M. Banda, Michael A. Schuh, and Tim Wylie, 2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW 2012), December 2012, pp. 805-812 [4]
- Spatiotemporal Co-occurrence Rules, Karthik Ganesan Pillai, Rafal A. Angryk, Juan M. Banda, Michael A. Schuh, and Tim Wylie, New Trends in Databases and Information Systems (ADBIS 2013), September 2013, Volume 241 pp. 27-35 [1]
- A Filter-and-Refine Approach to Mine Spatiotemporal Co-occurrences, Karthik Ganesan Pillai, Rafal A. Angryk, Berkay Aydin, In 2013 Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACMGIS 2013, pp. 114-123 [11].
- ERMO-DG: Evolving Region Moving Object Dataset Generator, Berkay Aydin, Rafal A. Angryk, Karthik Ganesan Pillai, In Proceedings of the 27th International Florida Artificial Intelligence Research Society Conference (FLAIRS'14), May 2014, pp. 321-326 [38].
- Hemispheric Patterns in Filament Chirality and Sigmoid Shape over the Solar Cycle", (Invited Review) 2014, Petrus C. Martens, Anthony R. Yeates, and Karthik G. Pillai, Proceedings of IAU Symposion 300, 122-125, "Nature of solar prominences and their role in Space Weather" [83].

APPENDIX B

SPATIOTEMPORAL CO-OCCURRENCE PATTERNS AND RULES

In this appendix section, we provide all the spatiotemporal co-occurrence patterns (STCOPs) and spatiotemporal co-occurrence rules (STCORs) generated for our solar data set explained in chapter 4, using our FastSTCOPs-Miner algorithm [11] and the procedure presented in our algorithm [1] to generate rules.

Table B.1: *Size-2* STCOPs

Events in STCOPs		Participation index
ActiveRegion	Sigmoid	0.29
Sigmoid	Sunspot	0.21
ActiveRegion	Sunspot	0.20
ActiveRegion	EmergingFlux	0.16
ActiveRegion	Filament	0.09
Sunspot	EmergingFlux	0.07
Filament	CoronalHole	0.06
EmergingFlux	CoronalHole	0.05
Filament	Sunspot	0.04
Filament	EmergingFlux	0.04
Sigmoid	EmergingFlux	0.03
Filament	Sigmoid	0.02
ActiveRegion	CoronalHole	0.01

Table B.2: STCORs derived from *size-2* patterns

STCORs (Left Side $\Rightarrow$ Right Side)		Conditional Probability	Participation Index (See Table B.1)
Left Side	Right Side		
Sunspot	ActiveRegion	0.93	0.20
Sigmoid	ActiveRegion	0.74	0.29
Sunspot	EmergingFlux	0.60	0.07
Sunspot	Sigmoid	0.48	0.21
ActiveRegion	Sigmoid	0.29	0.29
ActiveRegion	EmergingFlux	0.29	0.16
Sunspot	Filament	0.28	0.04
Sigmoid	EmergingFlux	0.21	0.03
Sigmoid	Sunspot	0.21	0.21
ActiveRegion	Sunspot	0.20	0.2
EmergingFlux	ActiveRegion	0.16	0.16
ActiveRegion	Filament	0.11	0.09
CoronalHole	EmergingFlux	0.11	0.05
CoronalHole	Filament	0.09	0.06
Filament	ActiveRegion	0.09	0.09
Sigmoid	Filament	0.07	0.02
EmergingFlux	Sunspot	0.07	0.07
Filament	CoronalHole	0.06	0.06
EmergingFlux	CoronalHole	0.05	0.05
EmergingFlux	Filament	0.04	0.04
Filament	Sunspot	0.04	0.04
Filament	EmergingFlux	0.04	0.04
EmergingFlux	Sigmoid	0.03	0.03
Filament	Sigmoid	0.02	0.02
CoronalHole	ActiveRegion	0.01	0.01
ActiveRegion	CoronalHole	0.01	0.01

Table B.3: *Size-3* STCOPs

Events in STCOPs			Participation index
ActiveRegion	Sigmoid	Sunspot	0.100
ActiveRegion	Sunspot	EmergingFlux	0.054
ActiveRegion	Filament	Sunspot	0.036
ActiveRegion	Sigmoid	EmergingFlux	0.025
Sigmoid	Sunspot	EmergingFlux	0.011
ActiveRegion	Filament	Sigmoid	0.010
ActiveRegion	Filament	EmergingFlux	0.004
Filament	Sigmoid	Sunspot	0.003
Filament	Sigmoid	EmergingFlux	0.002
Filament	Sunspot	EmergingFlux	0.002

Table B.4: STCORs derived from *size-3* patterns

STCORs (Left Side $\Rightarrow$ Right Side)		Conditional Probability	Participation Index (See Table B.3)
Left Side	Right Side		
Sigmoid,Sunspot	ActiveRegion	0.933	0.100
Filament, Sunspot	ActiveRegion	0.882	0.036
Sunspot, EmergingFlux	ActiveRegion	0.731	0.054
Sigmoid, EmergingFlux	ActiveRegion	0.679	0.025
Sunspot	ActiveRegion, Sigmoid	0.450	0.100
Filament, Sigmoid	ActiveRegion	0.444	0.010
Sunspot	ActiveRegion, EmergingFlux	0.425	0.054
ActiveRegion, Filament	Sunspot	0.412	0.036
ActiveRegion, Sunspot	EmergingFlux	0.386	0.054
ActiveRegion, EmergingFlux	Sunspot	0.354	0.054
ActiveRegion, Sunspot	Sigmoid	0.343	0.100
Sunspot	ActiveRegion, Filament	0.275	0.036
Sigmoid, Sunspot	EmergingFlux	0.267	0.011
ActiveRegion, Sigmoid	Sunspot	0.261	0.100
Sigmoid, EmergingFlux	Sunspot	0.250	0.011
Sunspot	Sigmoid, EmergingFlux	0.200	0.011
ActiveRegion, Sunspot	Filament	0.200	0.036
ActiveRegion, Sigmoid	EmergingFlux	0.200	0.025
Sigmoid	ActiveRegion, Sunspot	0.194	0.100
ActiveRegion, EmergingFlux	Sigmoid	0.159	0.025
Sunspot, EmergingFlux	Sigmoid	0.154	0.011
Sigmoid	ActiveRegion, EmergingFlux	0.145	0.025
ActiveRegion, Filament	Sigmoid	0.118	0.010
Filament, Sigmoid	Sunspot	0.111	0.003
Filament, Sigmoid	EmergingFlux	0.111	0.002
ActiveRegion	Sunspot, EmergingFlux	0.107	0.054
ActiveRegion	Sigmoid, Sunspot	0.100	0.100
ActiveRegion	Sigmoid, EmergingFlux	0.080	0.025
Filament, EmergingFlux	ActiveRegion	0.077	0.004
ActiveRegion	Filament, Sunspot	0.067	0.036
ActiveRegion, Filament	EmergingFlux	0.059	0.004
Filament, Sunspot	EmergingFlux	0.059	0.002
Filament, Sunspot	Sigmoid	0.059	0.003
Sigmoid	Sunspot, EmergingFlux	0.056	0.011
EmergingFlux	ActiveRegion, Sunspot	0.054	0.010
Filament, EmergingFlux	Sunspot	0.038	0.002
Filament, EmergingFlux	Sigmoid	0.038	0.002
Filament	ActiveRegion, Sunspot	0.036	0.036
Sigmoid, EmergingFlux	Filament	0.036	0.002
ActiveRegion, Sigmoid	Filament	0.035	0.010
Sigmoid, Sunspot	Filament	0.033	0.003
Sigmoid	ActiveRegion, Filament	0.032	0.010
ActiveRegion	Filament, Sigmoid	0.027	0.010
Sunspot	Sigmoid, EmergingFlux	0.025	0.011
Sunspot	Filament, EmergingFlux	0.025	0.002
EmergingFlux	ActiveRegion, Sigmoid	0.025	0.025
Sunspot, EmergingFlux	Filament	0.019	0.002
ActiveRegion, EmergingFlux	Filament	0.018	0.004
ActiveRegion	Filament, EmergingFlux	0.013	0.004
EmergingFlux	Sigmoid, Sunspot	0.011	0.011
Filament	ActiveRegion, Sigmoid	0.010	0.010
Sigmoid	Filament, EmergingFlux	0.008	0.002
Sigmoid	Filament, Sunspot	0.008	0.003
Filament	ActiveRegion, EmergingFlux	0.005	0.004
EmergingFlux	ActiveRegion, Filament	0.004	0.004
Filament	Sigmoid, Sunspot	0.003	0.003
Filament	Sunspot, EmergingFlux	0.003	0.002
Filament	Sigmoid, EmergingFlux	0.003	0.002
EmergingFlux	Filament, Sigmoid	0.002	0.002
EmergingFlux	Filament, Sunspot	0.002	0.002

Table B.5: *Size-4* STCOPs

Events in STCOPs				Participation index
ActiveRegion	Sigmoid	Sunspot	EmergingFlux	0.011
ActiveRegion	Filament	Sigmoid	Sunspot	0.003
ActiveRegion	Filament	Sigmoid	EmergingFlux	0.002
ActiveRegion	Filament	Sunspot	EmergingFlux	0.002
Filament	Sigmoid	Sunspot	EmergingFlux	0.002

Table B.6: STCORs derived from size-4 patterns

STCORs (Left Side $\Rightarrow$ Right Side)		Conditional Probability	Participation Index (See Table B.5)
Left Side	Right Side		
Filament, Sigmoid, EmergingFlux	ActiveRegion	1.000	0.002
Filament, Sigmoid, Sunspot	EmergingFlux	1.000	0.002
Filament, Sigmoid, Sunspot	ActiveRegion	1.000	0.003
Filament, Sunspot, EmergingFlux	ActiveRegion	1.000	0.002
Filament, Sunspot, EmergingFlux	Sigmoid	1.000	0.002
Filament, Sigmoid, EmergingFlux	Sunspot	1.000	0.002
Sigmoid, Sunspot, EmergingFlux	ActiveRegion	0.875	0.011
ActiveRegion, Filament, EmergingFlux	Sigmoid	0.500	0.002
ActiveRegion, Filament, EmergingFlux	Sunspot	0.500	0.002
ActiveRegion, Sigmoid, EmergingFlux	Sunspot	0.292	0.011
ActiveRegion, Filament, Sigmoid	EmergingFlux	0.250	0.002
ActiveRegion, Filament, Sigmoid	Sunspot	0.250	0.003
ActiveRegion, Sigmoid, Sunspot	EmergingFlux	0.235	0.011
Sigmoid, Sunspot	ActiveRegion, EmergingFlux	0.233	0.011
Sigmoid, EmergingFlux	ActiveRegion, Sunspot	0.214	0.011
Sunspot	ActiveRegion, Sigmoid, EmergingFlux	0.175	0.011
ActiveRegion, Sunspot, EmergingFlux	Sigmoid	0.174	0.011
Sunspot, EmergingFlux	ActiveRegion, Sigmoid	0.135	0.011
Sigmoid, Sunspot, EmergingFlux	Filament	0.125	0.002
ActiveRegion, Sunspot	Sigmoid, EmergingFlux	0.114	0.011
Filament, Sigmoid	ActiveRegion, EmergingFlux	0.111	0.002
Filament, Sigmoid	Sunspot, EmergingFlux	0.111	0.002
Filament, Sigmoid	ActiveRegion, Sunspot	0.111	0.003
ActiveRegion, Filament, Sunspot	EmergingFlux	0.067	0.002
ActiveRegion, Filament, Sunspot	Sigmoid	0.067	0.003
ActiveRegion, EmergingFlux	Sigmoid, Sunspot	0.062	0.011
ActiveRegion, Sigmoid	Sunspot, EmergingFlux	0.061	0.011
Filament, Sunspot	ActiveRegion, Sigmoid	0.059	0.003
Filament, Sunspot	Sigmoid, EmergingFlux	0.059	0.002
Filament, Sunspot	ActiveRegion, EmergingFlux	0.059	0.002
Sigmoid	ActiveRegion, Sunspot, EmergingFlux	0.048	0.011
ActiveRegion	Sigmoid, Sunspot, EmergingFlux	0.047	0.011
ActiveRegion, Sigmoid, EmergingFlux	Filament	0.042	0.002
Filament, EmergingFlux	Sigmoid, Sunspot	0.038	0.002
Filament, EmergingFlux	ActiveRegion, Sunspot	0.038	0.002
Filament, EmergingFlux	ActiveRegion, Sigmoid	0.038	0.002
Sigmoid, EmergingFlux	Filament, Sunspot	0.036	0.002
Sigmoid, EmergingFlux	ActiveRegion, Filament	0.036	0.002
Sigmoid, Sunspot	Filament, EmergingFlux	0.033	0.002
Sigmoid, Sunspot	ActiveRegion, Filament	0.033	0.003
ActiveRegion, Filament,	Sigmoid, EmergingFlux	0.029	0.002
ActiveRegion, Filament	Sunspot, EmergingFlux	0.029	0.002
ActiveRegion, Filament	Sigmoid, Sunspot	0.029	0.003
ActiveRegion, Sigmoid, Sunspot	Filament	0.029	0.003
Sunspot	Filament, Sigmoid, EmergingFlux	0.025	0.002
Sunspot	ActiveRegion, Filament, EmergingFlux	0.025	0.002
Sunspot	ActiveRegion, Filament, Sigmoid	0.025	0.003
ActiveRegion, Sunspot, EmergingFlux	Filament	0.022	0.002
Sunspot, EmergingFlux	Filament, Sigmoid	0.019	0.002
Sunspot, EmergingFlux	ActiveRegion, Filament	0.019	0.002
ActiveRegion, Sunspot	Filament, EmergingFlux	0.014	0.002
ActiveRegion, Sunspot	Filament, Sigmoid	0.014	0.003
EmergingFlux	ActiveRegion, Sigmoid, Sunspot	0.011	0.011
ActiveRegion, EmergingFlux	Filament, Sigmoid	0.009	0.002
ActiveRegion, EmergingFlux	Filament, Sunspot	0.009	0.002
ActiveRegion, Sigmoid	Filament, EmergingFlux	0.009	0.002
ActiveRegion, Sigmoid	Filament, Sunspot	0.009	0.003
Sigmoid	ActiveRegion, Filament, EmergingFlux	0.008	0.002
Sigmoid	Filament, Sunspot, EmergingFlux	0.008	0.002
Sigmoid	ActiveRegion, Filament, Sunspot	0.008	0.003
ActiveRegion	Filament, Sigmoid, EmergingFlux	0.007	0.002
ActiveRegion	Filament, Sigmoid, Sunspot	0.007	0.003
ActiveRegion	Filament, Sunspot, EmergingFlux	0.007	0.002
Filament	ActiveRegion, Sigmoid, Sunspot	0.003	0.003
Filament	ActiveRegion, Sunspot, EmergingFlux	0.003	0.002
Filament	Sigmoid, Sunspot, EmergingFlux	0.003	0.002
Filament	ActiveRegion, Sigmoid, EmergingFlux	0.003	0.002
EmergingFlux	ActiveRegion, Filament, Sunspot	0.002	0.002
EmergingFlux	ActiveRegion, Filament, Sigmoid	0.002	0.002
EmergingFlux	Filament, Sigmoid, Sunspot	0.002	0.002