5-10-2017

Discovery of Spatiotemporal Event Sequences

Berkay Aydin
Georgia State University

DISCOVERY OF SPATIOTEMPORAL EVENT SEQUENCES

by

BERKAY AYDIN

Under the Direction of Rafal Angryk, PhD

ABSTRACT

Finding frequent patterns plays a vital role in many analytics tasks such as finding itemsets, associations, correlations, and sequences. In recent decades, spatiotemporal frequent pattern mining has emerged with the main goal focused on developing data-driven analysis frameworks for understanding underlying spatial and temporal characteristics in massive datasets. In this thesis, we will focus on discovering spatiotemporal event sequences from large-scale region trajectory datasetes with event annotations. Spatiotemporal event sequences are the series of event types whose trajectory-based instances follow each other in spatiotemporal context. We introduce new data models for storing and processing evolving region trajectories, provide a novel framework for modeling spatiotemporal follow relationships, and present novel spatiotemporal event sequence mining algorithms.

INDEX WORDS: Spatiotemporal, Event, Sequence, Mining, Discovery

DISCOVERY OF SPATIOTEMPORAL EVENT SEQUENCES

by

BERKAY AYDIN

A Dissertation Submitted in Partial Fulfillment for the Degree of

Doctor of Philisopy

in the College of Arts and Sciences

Georgia State University

2017

DISCOVERY OF SPATIOTEMPORAL EVENT SEQUENCES

by

BERKAY AYDIN

Committee Chair: Rafal Angryk

Committee: Rajshekhar Sunderraman
Zhipeng Cai
Petrus Martens

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
May 2017

DEDICATION

I dedicate this work to my beautiful wife and my beloved family.

For their love, encouragement, and endless support...

ACKNOWLEDGEMENTS

This dissertation work would not have been possible without the support of many people. I want to express my gratitude to my dear advisor Dr. Rafal Angryk, for believing in me at first, staying behind me at difficult times, and guiding me with his expertise. I would also like to thank each of my committee members - Dr. Rajshekhar Sunderraman, Dr. Zhipeng Cai, and Dr. Piet Martens, for their guidance and encouragement.

I am grateful for all the help that my fellow lab mates - Karthik, Mike, Dustin, Ruizhe, Vijay, Sajitha, Ahmet, Soukaina, and Hamdi, provided over the years. I want to thank them for their great friendships, productive discussions, and well-appreciated support.

A special note of thanks to all my professors, back in Turkey at Bikent University, at Montana State University and at Georgia State University who contributed to my growth, both personally and professionally. Finally, I take this opportunity to thank all my good friends, especially Ezgi, Can, Feyyaz, Semih, Akif, Aytek, Evrim, Yasin, Oner, Fatih, and Deniz, who made my last five years more enjoyable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Discovering interesting, but implicit spatiotemporal patterns from datasets is crucial for many scientific domains such as astronomy [8, 9], ecology [10], meteorology [11], geophysics [12], and criminology [13]. The ever-growing nature of data being generated and collected from various scientific sources makes the data-driven knowledge discovery process very challenging to the researchers in these fields. An important branch of spatiotemporal pattern mining is the sequence (or sequential pattern) mining.

In traditional itemset mining, the frequent sequence (or sequential pattern) mining refers to discovering a set of attributes persistently appearing over time among a large number of objects [14]. A major category of sequences is event sequences. Event sequences represent the underlying sequential relationships among the categories of objects [15]. Event sequence mining can be useful for understanding the user behavior (by mining sequences from weblogs or system traces) [16], shopping routines of customers (by mining transaction sequences) [17], or the efficiency of business processes (by mining time-ordered managerial and operational activities) [18].

The focus of this thesis is to create and explore novel models and patterns of spatiotemporal event sequences from datasets with extended geometric representations. We can briefly define spatiotemporal event sequence mining as follows: given a dataset of spatiotemporal instances with associated event types, spatiotemporal event sequence mining identifies the frequently appearing sequences of event types whose instances spatially close-by and temporally follow each other. The spatiotemporal event sequence models form a novel analysis framework to explore interesting, useful, and non-trivial rules, and to facilitate their uses in descriptive and predictive tasks in various scientific fields.

1.1 Motivation

The spatiotemporal event sequence mining can be useful for the verification and prediction of scientific phenomena in a broad range of scientific fields including meteorology, geophysics, epidemiology, and astronomy [19]. While our primary application context is solar physics, the spatiotemporal event sequence mining is applicable to other scientific fields where moving region objects are present. The spatiotemporal event sequences can be used for modeling various scientific phenomena (*e.g.*, tornadoes, propagation of epidemics, clouds). The sequence patterns can be utilized for performing large-scale verification of current knowledge, as well as the prediction of unknown spatiotemporal relationships among different event types (*e.g.*, predicting the spread of epidemics such as cholera, malaria, and West Nile virus [6], verification of hurricane landfall precipitation models [20], discovery of the patterns in wildlife migration [21], or prediction of blastocyst formation [2]). We present three application domains where spatiotemporal event sequence mining can be used for verifying, predicting, or potentially discovering the spatiotemporal relationships and the characteristics of these relationships.

1.1.1 Solar Physics Application for Spatiotemporal Event Sequences

One important application area for spatiotemporal event sequence mining is the space weather prediction. Solar physics researchers entered the big data era with the launch of NASA's Solar Dynamics Observatory (SDO) mission, which captures approximately 60,000 high resolution images every day, and generates 0.55 petabytes of raster data each year. The big data trend in solar data is anticipated to be sustained by the ground-based DKIST telescope, which is expected to generate three to five petabytes of data each year [9]. In addition to image data, many software modules continuously work on SDO's image data, in order to detect instances of various solar event types. The detected solar events can be considered as vector-based objects with spatial and temporal attributes [8].

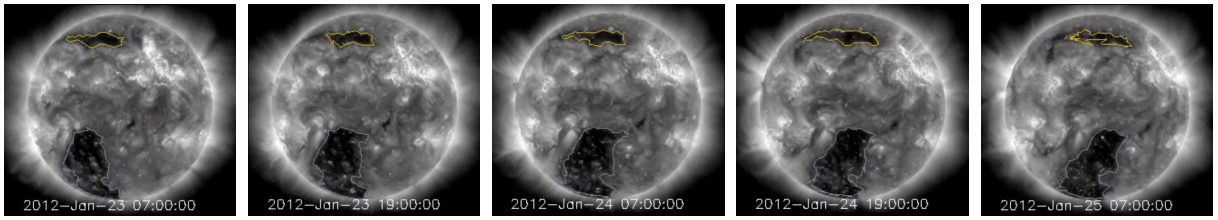


Figure 1.1: Polygon-based representations of two *coronal holes* reported to Heliophysics Event Knowledgebase (HEK) [1] between '23 January 2012 07:00' and '25 January 2012 07:00'.

Recently, a large-scale solar image dataset with labeled regions was published in [22], and a tracking algorithm was introduced in [23] (See Figure 1.1 for two tracked coronal hole instances). The solar event tracking algorithm uses the locations and corresponding image parameters [22] for linking the polygon-based evolving regions. Then, it creates spatiotemporal trajectory objects with extended geometric representations. Additionally, we introduced four spatiotemporal interpolation techniques for increasing the location accuracy of the trajectories [24]. In essence, we can access and make use of vector-based solar event data, which is in the form of spatiotemporal trajectories of continuously evolving regions.

Spatiotemporal event sequences frequently transpire among solar events such as active regions, flares, and sunspots. Identifying spatiotemporal event sequences appearing on the Sun can help us better understand the implicit spatial and temporal relationships among solar event types, and eventually lead to better modeling and forecasting of important events such as coronal mass ejections and solar flares. Coronal mass ejections and solar flares impact radiation in space, can reduce the safety of space and air travel, disrupt intercontinental communication and GPS, and even damage power grids [25].

1.1.2 Biomedical Sciences – Embryo selection prediction

In vitro fertilization (IVF) is a complex series of procedures used to treat fertility or genetic problems and assist with the conception of a child. IVF technology allowed

us to view and analyse the early events of human fertilization and embryogenesis [26]. Conventional embryo selection methods are still associated with a relatively low IVF success rate with a clinical pregnancy rate of approximately 30% per transfer [27]. This often leads to the transfer of more than one embryo at a time, which increases the risk of multiple pregnancies, and the associated neonatal complications and maternal pregnancy-related health problems [28]. Improvements in methods to select embryos for transfer would potentially enable further increases in pregnancy rates, and facilitate broader acceptance and adoption of single embryo transfer [29]. Nevertheless, the basic pathways and events of early human embryo development and the factors aiding the prediction of success and failure is not well-known [2].

Time-lapse imaging is an emerging tool that allows the identification of parameters that can potentially help predict the developmental potential of an embryo with continuous monitoring [3]. Time-lapse observation presents an opportunity for optimizing embryo selection based on morphological grading as well as providing novel kinetic parameters, which may further improve accurate selection of viable embryos [30]. Time-lapse imaging can also aid in transforming the early embryo images into spatiotemporal vector data, which can be used in spatiotemporal frequent pattern mining. In Figure 1.2 and 1.3, two illustrations of embryo cells from [2] and [3], which are tracked with an automated image analysis software.

In [29], Herrero and Meseguer present their findings on the predictive markers that influence the success rate of IVF. Those markers include spatial characteristics of the early embryo stages such as appearence (shape) of pronuclei (nucleus of sperm and egg), and temporal characteristics such as duration of first cleavage, and time interval between first and second mitotic division. Conaghan suggests that slower blastocyst formation is associated with poorer embryo viabiliy [3]. The associated markers as well as the embryo cells can be modeled as moving objects with evolving regions. The validity of these markers and predictors can be tested with spatiotemporal event sequence mining

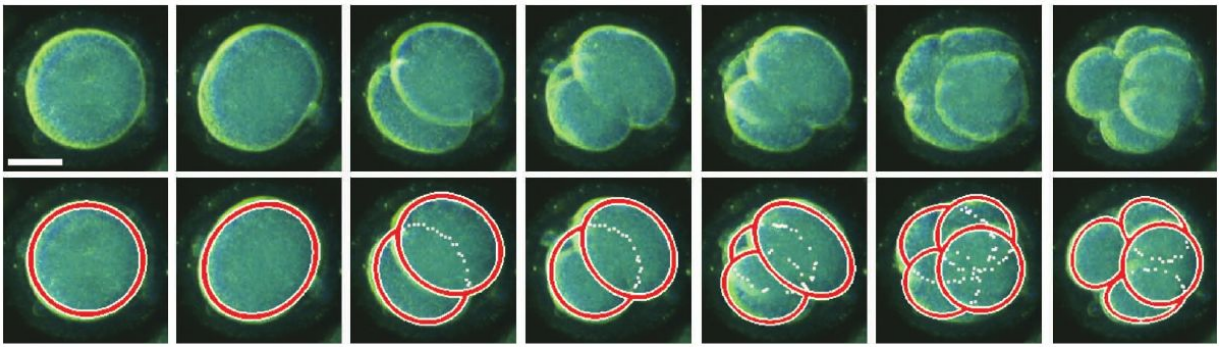


Figure 1.2: In [2], Wong et al. illustrate their cell tracking results, and compare its accuracy with manual image analysis performed by human experts. They argue that these two methods have excellent agreement. The tracking software models the embryos as a collection of ellipses with position, orientation, and overlap indices. Images in top row show the frames from original time-lapse sequence. Images in bottom row show the overlaid ellipses found after tracking. Wong et al. claims that with these models, the duration of cytokinesis and time between mitoses can be identified. (Image is copied from [2] – See Figure 1.a)

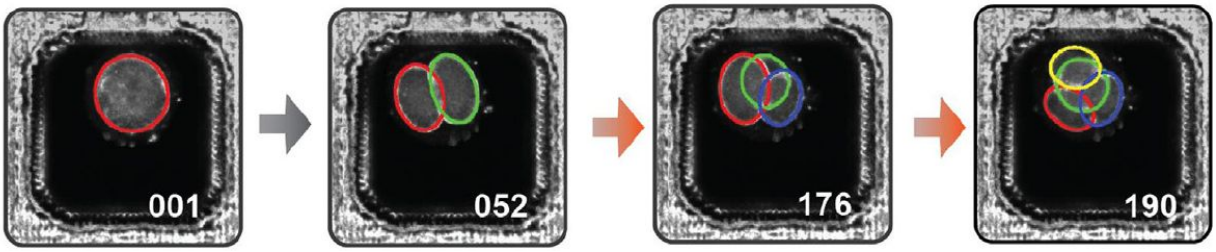


Figure 1.3: In [3], Conaghan et al. present the results of the tracking software they used. The primary features tracked by the software are the cell membranes. By using a data-driven probabilistic framework, the software generates an embryo model that includes an estimate of the number of blastomeres, as well as spatiotemporal attributes such as size, location, and shape, as a function of time. (Image is copied from [3] – See Figure 2.a)

with performing a verification task on a large scale dataset. Such data analyses can aid the scientists better comprehend the relationships among different procedures in the process of IVF.

1.1.3 Epidemiology – Prediction of malaria epidemics

It is commonly accepted that climate plays a role in the transmission of many infectious diseases, some of which are among the most important causes of mortality and mor-

bidity in developing countries [6]. The early identification of an epidemic of infectious disease is an important first step towards implementing effective interventions to control the disease and reduce the resulting mortality and morbidity in human populations. However, the epidemics are usually well advanced before the authorities are notified and epidemic control measures are prepared or deployed [31].

Malaria shows significant seasonal patterns by which the disease transmission is highest in the months of heavy rainfall and humidity [4]. The spatial distribution of disease-transmitting insects are closely related with these phenomena, where a rise in temperature accelerates the reproduction rate of insects, or humid weather conditions create desirable reproduction habitats for insects [5, 32]. Malaria demonstrates its most catastrophic effects in sub-Saharan Africa, where it is one of the largest causes of morbidity and mortality, creating a significant barrier to economic development [33].

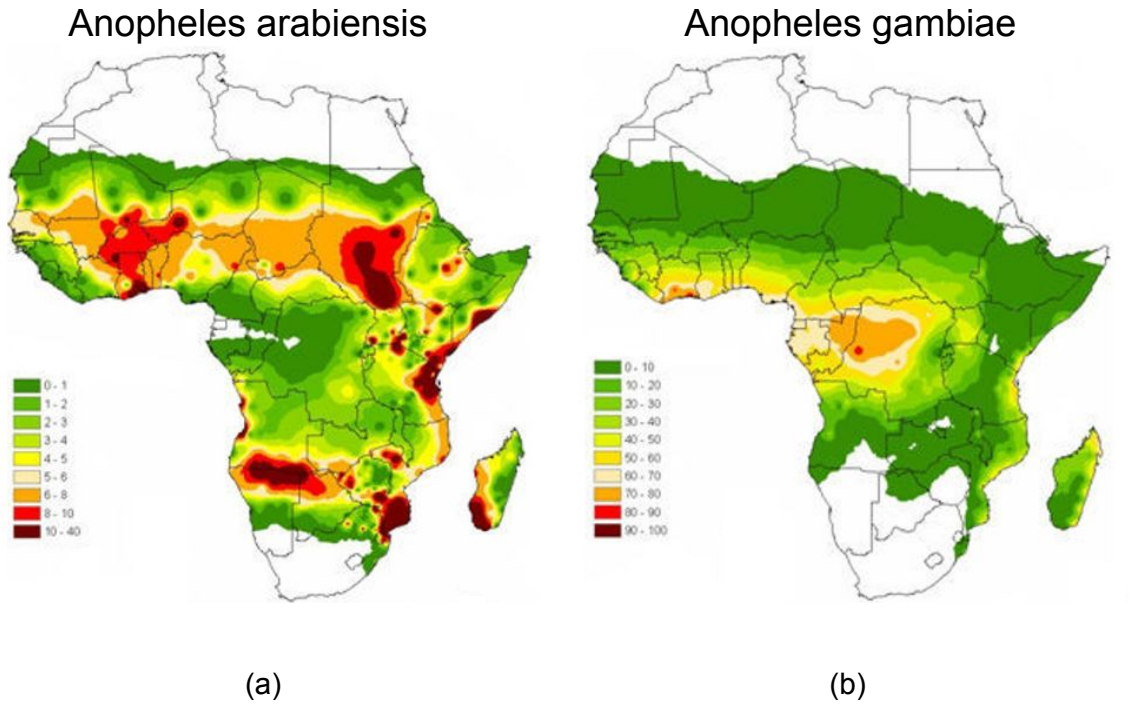


Figure 1.4: In [4], Tonnang et al. present the spatial distribution map of two important mosquito malaria vectors – *A. arabiensis* (a) and *A. gambiae* (b) under current climate conditions. (Images are copied from [4] – See Figures 1.A and 2.A)

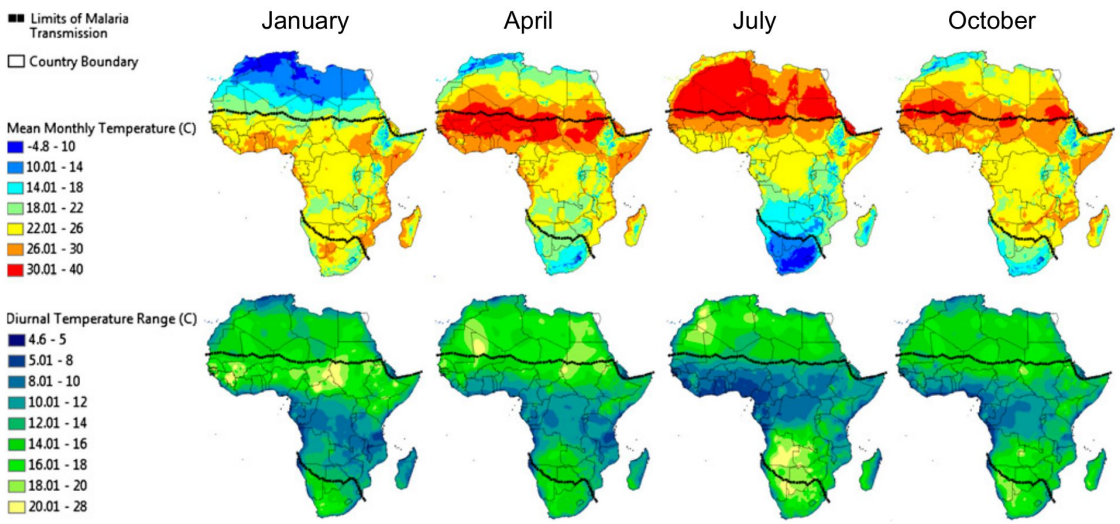


Figure 1.5: In [5], Paaijmansa et al. discuss that malaria transmission is heavily influenced by mean temperatures as well as the daily temperature fluctuations. Maps on top row show the mean monthly temperatures. Maps on the bottom row shows the diurnal temperature range. The diurnal temperature range is the difference between the daily maximum and minimum temperature. (Image is copied from [5] – See Figure 1)

Spatiotemporal event sequence mining can be helpful for prediction of epidemics by demonstrating the associations between climatic risk factors and disease outbreaks. The areas influenced by epidemics caused by mosquito vectors (See Figure 1.4), high and low temperature areas (See Figure 1.5), and rainfall anomaly zones (See Figure 1.6) can be modeled as spatiotemporal objects with extended geometric representations.

1.2 Challenges

The task of spatiotemporal event sequence discovery is challenging primarily because of the difficulty of identifying sequence forming event instances. From a theoretical standpoint, firstly, a consistent and flexible definition for spatiotemporal follow relationship is vital for the correctness and relevancy of the mining algorithms. Secondly, a meaningful significance measurement technique for the spatiotemporal follow relationship is necessary. From a practical point of view, the computational operations required for

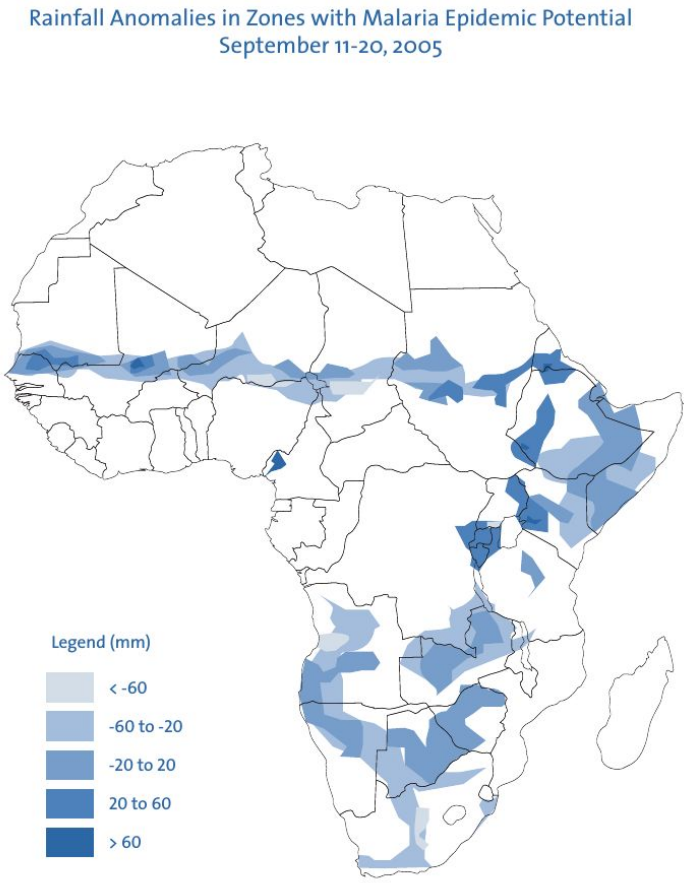


Figure 1.6: In a technical report from World Health Organization (WHO), Kuhn et al. suggest that rainfall anomaly data can be combined with epidemiology data to create predictive models [6]. Map shows the near real-time rainfall anomaly zones using the information provided by climate satellites. (Image is copied from [6] – See Figure 3)

identifying the sequence forming instances are computationally expensive due to the nested joins with complex spatial and temporal predicates.

Unlike most of the moving point object datasets used in spatial and spatiotemporal data mining literature [34], we are interested in spatiotemporal event sequence discovery from event instances with polygon-based geometries. Therefore, mining knowledge from these event datasets requires creating novel algorithms that can handle event instances with polygon-based geometries. While it seems trivial, continuously evolving and moving region objects necessitate developing new significance measures and spatiotemporal structures for two reasons: (1) Polygon-based geometries have very rich semantics (such as area, rotation, and shape) compared to the point-based ones; and (2)

storage and processing of polygon-based geometries are computationally more expensive than their point-based counterparts.

1.3 Contributions

Our controbutions can be classified into three categories: (1) Modeling the spatiotemporal event instances, (2) Creating a flexible and extensible framework for sequence generating behavior, *i.e.* designing the predicates of spatiotemporal follow relationship, and (3) developing new algorithms and data structures for effectively and efficiently mining the spatiotemporal event sequences.

The building blocks of the spatiotemporal event sequence mining are the data models created for representing the trajectories of moving region objects whose regions continuously change their location and shape. We created different models for representing the trajectories in multiple different computing environments. We used the raw trajectory data model for our event instances. It is also worth noting that our data models are extensible with temporal or non-spatiotemporal attributes.

Secondly, we designed a flexible model for the spatiotemporal follow relationships. Spatiotemporal follow relationships characterize the sequence forming behavior between event instances. In a nutshell, the predicates of spatiotemporal follow relationship checks whether an instance starts after another, and they are located close by. Our model uses the simple interval algebra for temporal starts after relationships between instances. For inspecting the closeness of locations, we use the spatiotemporal co-occurrence relationships between the heads and tails of the instance's trajectories. To understand the significance of a spatiotemporal co-occurrence, we designed a new significance measure based on the classical Jaccard measure.

Thirdly, we developed algorithms based on our spatiotemporal trajectory and follow relationship models. We will present three categories of algorithms for mining the spatiotemporal event sequences. First category of our algorithms are threshold-

based algorithms. We will present two Apriori-based and one pattern growth-based algorithm. The *NaïveApriori* and *SequenceConnect* are Apriori-based and the ESGrowth is the pattern growth-based algorithm. The second category of our algorithms are Top-*(R%, K)* spatiotemporal event sequence mining algorithms. We will present two Top-*(R%, K)* spatiotemporal event sequence mining algorithms that are: *Naïve* and *Fast* Top-*(R%, K)* -EsMiner. The last category is the bootstrap-based algorithm for explorative analysis. Our novel bootstrap-based algorithm (called Btsp-EsMiner) do not require any thresholds for the mining process. In addition to the algorithmic developments, we conducted an extended experimental evaluation. In our experimental evaluation, we check the correctness of our algorithms, compare and analyse their running time performance in different datasets, and provide a brief relevancy analysis.

1.4 Outline

This thesis is organized based on the above-mentioned contributions. In this section, we have explained the focus of our thesis, provided motivation for our research with various application areas, revealed the challanges and presented the contributions of this work. The rest of this thesis is organized as follows. In Chapter 2, we reviewed the spatial, temporal, and spatiotemporal frequent pattern mining literature on sequence patterns. In Chapter 3, we will present the background information on spatiotemporal trajectory data and provide our trajectory data models. Next, in Chapter 4, we will revisit our works on measuring the significance of spatiotemporal co-occurrences, which is the backbone of the spatiotemporal follow relationship. In Chapter 5, we will introduce the spatiotemporal follow relationship, and, later, we will present our algorithms for mining the spatiotemporal event sequences. In Chapter 6, we will present our experimental evaluation, and, lastly, we will conclude this thesis and discuss future work in Chapter 7.

2 LITERATURE REVIEW ON SPATIAL AND SPATIOTEMPORAL DATA MINING

Spatiotemporal data mining refers to the extraction of knowledge, regularly repeating relationships, and interesting patterns from spatiotemporal data [35]. In recent years, many spatiotemporal frequent pattern mining algorithms were developed for evolving region objects [36–42]. These algorithms focus on the discovery of spatiotemporal co-occurrence patterns by inspecting the spatiotemporal overlap (*i.e.*, co-occurrence of instances in spatial and temporal dimensions). We will investigate the sequences of event types whose instances frequently follow each other in spatiotemporal context. Our discussion of the current work on spatiotemporal frequent pattern mining literature starts with types of spatiotemporal knowledge to be extracted from spatiotemporal data, primarily from trajectories. Then, we will present the recent research work on temporal sequence patterns, spatial colocation patterns, and spatiotemporal co-occurrence patterns. Lastly, we will discuss the different types of spatiotemporal sequence patterns, and compare them with our work.

2.1 Types of Spatiotemporal Knowledge

There are eight categories of the spatiotemporal knowledge discovery described by Abraham et al. in [43], Roddick et al. in [44], and Shekhar et al. in [34] are: outlier, association (coupling), generalization (summarization), prediction, clustering (partitioning), hotspot, evolution rule (change), and meta-rule. Table 2.1 shows the descriptions of these knowledge types in detail with the example data mining applications in the literature. The tasks in frequent pattern discovery from spatiotemporal data require mining of multiple types of knowledge from the above-mentioned categories [19]. The examples of frequently occurring spatiotemporal patterns can be seen in various scientific

fields such as material science, epidemiology, biology, meteorology, ecology, and astronomy [10–13, 45, 46]. For instance, identification of anomalous moving objects (outlier detection) can be used in ecology for detecting outliers in bird migration. Another example is the spatiotemporal hotspot detection, which can be used for understanding the dynamics of epidemics in a geographic region.

Our pattern of interest is the spatiotemporal event sequences. The spatiotemporal event sequences fall under the category of spatiotemporal associations (or couplings). The patterns in our work (*i.e.*, the event sequences) are formed by a series of event types (feature types), whose instances frequently satisfy the spatiotemporal follow predicate. The resulting sequence patterns (event sequences) signify the relationships among the different event types and their strength in the datasets.

2.2 Temporal Sequence Patterns

Classical sequential pattern mining is concerned with discovering a set of attributes, shared across time, among a large number of objects in a given sequence dataset [14]. The sequence data contain lists of time annotated transactions, where each transaction contains a set of discrete attributes (in other words items). Notable algorithms for discovering sequential patterns are: Srikant and Agrawal's AprioriAll algorithm [17], Zaki's SPADE algorithm [14], and Pei et al.'s PrefixSpan algorithm [61]. These algorithms are primarily concerned with the time point data, where the temporal aspects of the objects in the datasets are represented as timestamps.

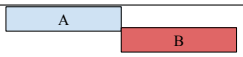
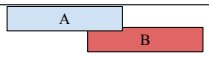
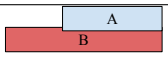
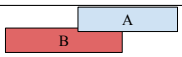
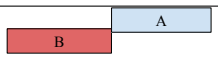
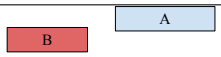
There is also a branch of sequential pattern mining, where researchers investigate the time interval patterns. Allen introduced a set of algebraic operations for temporal intervals [7]. These algebraic operations can be seen in Table 2.2. Allen's interval algebra is widely used in temporal data mining applications. In recent years, many algorithms for the discovery of sequential patterns from time interval data has been proposed. Papapetrou et al. used an enumeration tree to discover arrangements (sequences) of

Table 2.1: Types of spatiotemporal knowledge

Type	Description	Example
Outlier	Spatiotemporal outliers refers to the objects whose non-spatiotemporal attributes significantly differs from those of other objects in its spatiotemporal neighborhood	Identification of anomalous moving objects [47], discovering flow anomalies in spatial networks [48]
Association (Couplings in [34])	Patterns and association rules formed by feature types, where instances of participating types satisfies a complex or simple spatiotemporal predicate [49]	Discovering STCOPs [37], mining spatiotemporal sequential patterns [50]
Generalization (Summarization in [34])	Similar to the classical data mining counterpart [51], spatiotemporal generalization is the process of data aggregation using concept hierarchies to create a compact representation of spatiotemporal data [34, 44]	Summarization of network trajectories in K-primary corridors [52]
Prediction	Spatiotemporal prediction aims to learn a model that can predict a target variable dependent on spatiotemporal explanatory variables [34]. When the variable is categorical, the task is also referred to as classification.	Dynamic spatiotemporal models with Bayesian hierarchical framework [53], spatiotemporal autoregressive regression [53]
Clustering (Partitioning in [34])	Spatiotemporal clustering is the task of grouping similar data items based on their spatial, temporal, or spatiotemporal attributes [54]	Spatiotemporal event clustering [55], trajectory data partitioning based on their similarity [56]
Hotspots	Hotspots are special clusters (or regions) where an attribute or the number of spatiotemporal objects are unexpectedly higher within particular time intervals [34]	Discovery of emerging spatiotemporal hotspots for epidemic diseases [57]
Evolution Rule (Spatiotemporal Change in [34])	Evolution rules refer to the explicit spatiotemporal evolution actions (variations in spatial and temporal footprints), which a particular set of objects frequently performs [58].	Identification of spatial changes between snapshots using raster-based spatial footprints [59], spatiotemporal volume change patterns [60]
Meta-rule	Process of performing data mining on a set of discovered knowledge instead of datasets [44]	Tracking the differences between spatiotemporal association rules that change over different datasets [44]

interval-based events using a hybrid depth-first and breadth-first search based (H-DFS) method [62]. Winarko and Roddick introduced ARMADA, which is a projection-based efficient time interval pattern mining algorithm that utilizes an iterative candidate gen-

Table 2.2: Temporal relationships in Allen's interval algebra [7]

Relationship	Symbol	Illustration
A before B	A **<** B	
A meets B	A **m** B	
A overlaps B	A **o** B	
A finished by B	A **fi** B	
A contains B	A **di** B	
A starts B	A **s** B	
A equals B	A **=** B	
A started by B	A **si** B	
A during B	A **d** B	
A finishes B	A **f** B	
A overlapped-by B	A **oi** B	
A met-by B	A **mi** B	
A after B	A **>** B	

eration and pruning approach [63]. Wu and Chen proposed TPrefixSpan, which is a modified version of the PrefixSpan algorithm [61] for mining temporal patterns from time interval events. Patel et al. introduced the IEMiner algorithm which extends H-DFDS method [62] by extending the sequences during the discovery process. Moskovitch and Sharar proposed KarmeLego for the discovery of frequent symbolic time intervals related patterns [64]. KarmaLego uses a temporal abstraction process from raw timestamped data, and utilizes a data structure (enumeration tree) and exploits the transitivity of Allen's operations for efficient candidate sequence generation.

2.3 Spatial Colocation and Spatiotemporal Co-occurrence Patterns

Spatiotemporal co-occurrence pattern mining is conceptually similar to the classical frequent pattern mining from transactional databases. However, the implicit spatial and

temporal semantics (specifically spatial and temporal overlap) are required to be identified, and the identification of these relationships dramatically increase the complexity of the STCOP mining algorithms due to the expensive join operations with spatiotemporal predicates. In spatiotemporal frequent pattern mining, the underlying spatiotemporal semantic relationships (such as co-occurrence, sequence, or periodicity) are the main subjects of discovery. The co-occurrence relationship is originated from the significance of closeness in spatial and temporal dimensions, by asserting the instances located in spatial and temporal proximity are more related than the others [65].

One pioneering advancement in spatial data mining is the discovery of spatial colocation patterns [66]. The spatial closeness of the objects is introduced as the *colocation* relationship. Given a set of boolean spatial features (events), spatial colocation mining aims to discover the subsets of events whose instances are frequently colocated together. As a matter of course, it is often very hard to observe point-based spatial objects sharing exactly the same locations. Therefore, a neighborhood relationship (based on user specified distance thresholds) is used for defining the colocations. The spatial colocation mining algorithm in [66] uses an Apriori-based approach [67], which requires a spatial join algorithm while generating and pruning the candidate patterns. Partial-join and join-less approach for mining colocations were presented in [68] and [69]. Furthermore, statistically significant colocation patterns (SSCP) represent subsets of event types whose instances are colocated due to spatial dependency [70].

While colocation refers to purely spatial closeness of objects, the term *co-occurrence* is more frequently used for spatiotemporal closeness. Mixed-drove spatiotemporal co-occurrence patterns (MDCOP) are introduced in [71]. MDCOPs represent the subsets of spatiotemporal event types whose point-based instances are frequently occurring in spatial and temporal proximity. The aim of discovering MDCOPs is to find mixed groups (*i.e.*, of different event types) of spatiotemporal instances, which are spatially close-by and temporally persistent in time. MDCOP-mining algorithms presented in [71] can be

interpreted as a temporal extension of spatial colocation mining algorithms to spatiotemporal context. The proposed MDCOP-Miner algorithms follow a similar Apriori-based approach. Following MDCOPs, the sustained emerging (SECOP) [72], the partial (PACOP) [73], and the periodical (PECOP) [74] spatiotemporal co-occurrence patterns are introduced. Fundamentally, emerging, partial, and periodical co-occurrence relationships are quite similar to the MDCOPs. They include additional constraints for more complex spatiotemporal relations, and require new interest measures tuned for these constraints. SECOPs represent the subsets of event types whose instances are increasingly colocated in space and time. PACOPs are concerned with the discovery of spatiotemporal co-occurrences that are partially (not as frequently) present in the database. PECOPs represent the subsets of event types that are periodically co-occurring.

Spread patterns of spatiotemporal co-occurrences over zones (SPCOZ) are introduced in [75]. SPCOZs represent the subsets of event types whose instances are spreading and co-occurring over particular zones. The main purpose of the mining SPCOZs is discovering spreading structures that co-occur together both in space and time (meaning correlations among the spreading structures are mined instead of trajectories). Another instance of spatiotemporal co-occurrence pattern mining is composite spatiotemporal co-occurrence (COSTCOP) [76], where a new composite prevalence measure (using spatial and temporal dimensions together) is developed, and a pruning technique is developed for improving the performance of the mining algorithm.

The aforementioned spatiotemporal co-occurrence or colocation models are designed for event instances with point-based geometric representations. As point-based instances exhibit nearly imperceptible spatial and temporal overlap relationships among each other, the spatial and temporal neighborhoods are to be defined for characterizing co-occur-rences or colocations. However, in spatiotemporal co-occurrence pattern mining from evolving region trajectories (defined over polygon data type), it is highly likely to observe spatial and temporal coincidences (namely spatiotemporal overlap relation-

ships). Mining spatiotemporal co-occurrence patterns from datasets with evolving regions was introduced in [36]. The event instances, which are represented by polygons evolving over time, are treated as three-dimensional continuous objects. To decide whether an overlap among these three-dimensional structures form a significant co-occurrence, a spatiotemporal version of Jaccard significance measure is used. Similar to the other co-occurrence patterns, an Apriori-based algorithm (including a spatiotemporal join over spatial and temporal overlap predicates) is used. In [37], a novel filter-and-refine strategy for pruning the instances in the spatiotemporal join phase using OMAX measure is proposed. We improved this algorithm further in [38] by utilizing spatiotemporal indexing techniques for trajectory-based instances and in [41] by utilizing a frequent pattern growth-based filter. Additionally, we provided a distributed STCOP mining framework using columnar databases in [40].

2.4 Spatiotemporal Sequence Patterns

In the spatiotemporal frequent pattern mining literature, the term *sequence* (or its derivatives such as *sequence patterns*, *sequential patterns*) is used for identifying different types of knowledge from spatiotemporal data. These include sequences of locations frequently visited by spatiotemporal objects [77], sequences of event types whose instances follow each other [50], and sequences of spatiotemporal association rules [78].

Cao et al. describe the spatiotemporal sequential patterns as 'the routes which are frequently followed by objects' in [77]. Namely, a list of frequently visited locations is discovered from a dataset of spatiotemporal trajectory segments. This work is related to the movement patterns of spatiotemporal objects in the form of trajectory segments. Similarly, Giannotti et al. introduce *trajectory patterns*, and present a mining algorithm for mining trajectory patterns in [79]. Trajectory patterns represent a set of spatiotemporal objects that frequently visit similar locations with similar visiting times. While Giannotti et al.'s work is more focused on the behavioral aspect of spatiotemporal objects, the se-

quences refer to the ordered lists of visited locations. Verhein introduces the mining on complex spatiotemporal sequence patterns in [78]. Complex spatiotemporal sequence pattern mining focuses on the sequences of spatiotemporal association rules that represent frequently occurring movements of spatiotemporal objects appearing between two regions during a particular time interval. Namely, the work is interested in discovering the sequences of spatiotemporal meta-rules (movement patterns) for groups of objects.

Huang et al. presented a framework for mining sequential patterns from spatiotemporal event datasets in [50]. The sequential patterns, in [50], refer to a sequence of event types from spatiotemporal objects with event type annotations. They formally define a follow relationship between the point-based event instances of two different event types, present significance measures for sequences, and introduce two iterative pattern growth algorithms for mining task. Their algorithms create a pattern tree and expand its nodes with recursively calling the tree expansion procedures (namely, follow joins). It should be noted that sequential pattern mining in [50] considers a totally ordered event instances. In [80], a mining algorithm for partially ordered subsets of event types are presented.

Similar to [50], we are interested in sequences of event types, and our work is not applicable for discovering sequences of locations or movement behaviors. We use the term *spatiotemporal event sequence* to avoid confusion with other types of existing sequence or sequence pattern definitions. Our work focuses on discovering frequently occurring sequences of event types from the evolving region objects that are totally ordered.

Apart from those, Zhang et al. proposed the *Splitter* algorithm, which discovers fine-grained sequential patterns from semantic trajectories [81]. The algorithm firstly retrieves spatially coarse patterns and later reduces them to fine-grained patterns. The discovered patterns are sequences of categorized locations (deduced from semantic trajectories). Another example of spatiotemporal sequences, called spatio-sequences, are presented by Salas et al. in [82]. The spatio-sequence mining discovers temporal se-

quences of ordered spatial itemsets that are used for coupling geographically neighboring phenomena.

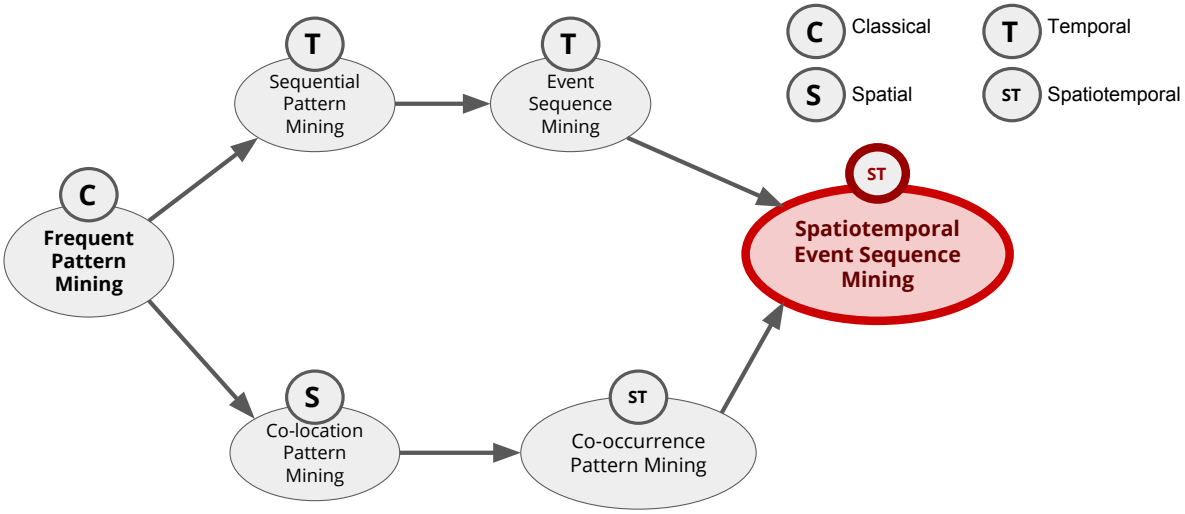


Figure 2.1: Family tree of spatiotemporal event sequence mining

Spatiotemporal event sequence (STES) mining has its roots in both spatial and temporal patterns mentioned here. In Figure 2.1, we depict an overview of the evolution of the spatiotemporal event sequences from the perspective of pattern mining. Firstly, similar to the temporal sequence patterns (*i.e.*, temporal event sequences) we are interested in finding event types whose instances temporally follow one another. In addition to that, STES mining is also interested in the spatial proximity of the instances that temporally follow each other. Unlike the earlier approaches, we created a spatiotemporal follow relationship that is based on the spatiotemporal co-occurrence relationships. We will throughly explain the follow relationship in Chapter 5–Section 5.1.

3 PRELIMINARIES ON SPATIOTEMPORAL DATA

The rapid advancements in satellite imagery technology (MODIS Terra and Aqua [83], NOAA GOES [84], NASA's SDO [85]), GPS enabled devices (mobile phones, vehicles, smartwatches), location-based web services (Google Maps, Uber, Lyft), and social networks (Facebook, Twitter, Swarm) caused a proliferation of massive spatiotemporal data sets in the last two decades. Many consumer-oriented applications from social networks to mobile services including routing and taxi services consume and generate spatiotemporal location data [86]. Furthermore, there are many massive spatiotemporal data repositories generated by scientific resources that are monitoring the moving objects. These include solar events [22], animal migrations [87], and meteorological phenomena [88].

The explosive growth in spatiotemporal data as well as the emergence of new technologies emphasize the need for automated discovery of spatiotemporal knowledge. One of the most interesting data mining applications is spatiotemporal data mining from trajectory data. Some examples of trajectory mining include destination and future route prediction based on trajectory mining [89], real-time monitoring of water quality using temporal trajectories of live fish [90], analyzing the trajectories of bird migrations [91], searching for similar trajectories in spatial networks [92], and traffic mining [93].

In this chapter, we will focus on the spatiotemporal object modeling with a focus on the moving objects with extended geometric representations. Our spatiotemporal event sequence mining algorithms primarily make use of region trajectories whose polygon-based region representations continiously evolve over time. In the rest of this chapter, we will firstly introduce the conceptual modeling of spatiotemporal trajectories and moving objects. Then, we will present the evolving region trajectories and spatiotemporal event instances which are the base data types in our mining schema.

3.1 Moving Objects and Spatiotemporal Trajectories

Spatial objects that move or change their shape over time are often referred to as moving objects. Mainly, there are two important abstractions of moving objects: moving *point* objects and moving *region* objects [94]. In [95], Guting et al. presented an abstract and a discrete data model for storing and processing moving objects. In the abstract model, geometric objects are modeled as point sets. For continuous objects such as regions, the set of points are infinite. Conceptually, the abstract model is simple, however implementation cannot be performed without transformation. Guting et al.'s discrete model is conceptually more complex, but it can be implemented practically in real-life applications.

Spatiotemporal trajectories are essentially the paths followed by the moving objects. In other words, trajectories describe the physical movement of moving objects that are changing their locations over time. For the simple case of moving point objects, the trajectories can be represented as line segments or curves that pass from the recorded locations of the moving point objects. On the other hand, for moving region objects, trajectories create a three-dimensional[1] path which can be depicted as a three-dimensional volume.

The data modeling for spatiotemporal trajectories is studied in many recent studies on spatiotemporal databases [79, 91, 95–99]. Most notably, in [91], Spaccapietra et al. state that there are two facets of a trajectory that are: geometric facet and semantic facet (See Figure 3.1). The geometric facet considers the geometric representation of the object in space over time and can be implemented using the raw trajectory data model. The raw location data of moving objects are recorded and create the trajectory. The semantic facet, on the other hand, gives a meaning or context to the movement of the object. The semantics of the trajectory refers to the application oriented meaning of the movement,

1 In this case, the space is considered as two-dimensional. For the case of three dimensional space, trajectories create a four-dimensional hyper volume path.

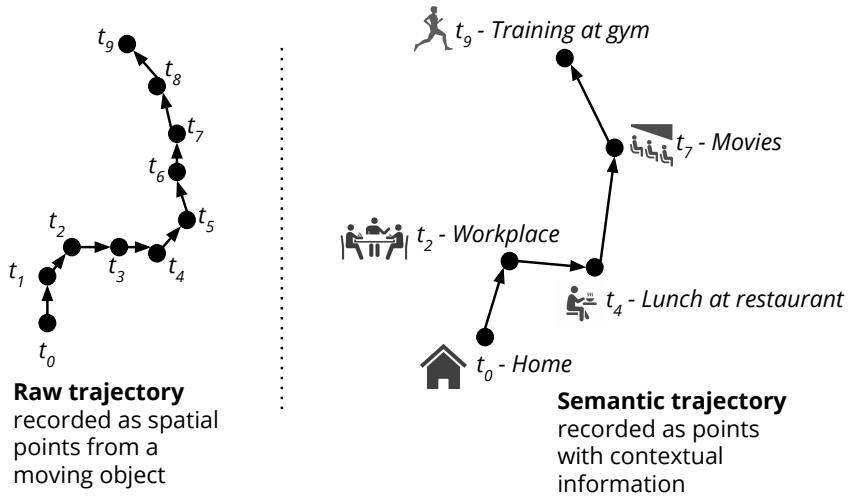


Figure 3.1: Raw trajectory (on the left) recorded as spatial locations of moving points object and the semantic trajectory (on the right), which contains the application specific contextual information

and it is linked or mapped to the real-life geographical knowledge. Semantic trajectories can be represented with structured (also referred to as symbolic in [99]) or semantic [98] trajectory data models. Adding the contextual information to the trajectories not only enriches the trajectory data model, but also help us understand the activity, and may reduce the storage requirements of the model. In Figure 3.1, we illustrate the geometric and semantic facet of a moving object using the raw trajectory data model (on the left) and the semantic trajectory data model (on the right).

3.2 Evolving Region Trajectories

Moving objects is one of the most prominent data types in spatiotemporal database research. The spatial aspect of a moving object is represented with geometric objects (such as points, lines, or polygons) that show its locations. As the name suggests, the locations of the moving objects change over time. A moving object is an abstraction representing the movement of a spatial object whose location change over time.

A category of moving objects is *moving region objects*. The locations of moving region objects are represented using polygon-based spatial data types. Thus, apart from the mere location, moving region objects also encapsulate time-dependent spatial change information such as shape, rotation, and areal evolution. It is also fair to state that not all the real-life phenomena designed as moving region objects have all the spatial evolution characteristics. In some cases, the change never happens or these evolution characteristics are not relevant to the domain. We can give the following examples:

- **Per capita income of U.S. counties in quantiles as moving region objects:** Each quantile of county per capita income can be represented as a complex moving region object (of multipolygons) that changes its complex locations over time as the per capita income ranks (its quantile) of counties change. The location and area of these regions are important for socio-economists. For instance, they can be used for showing that the wealth is concentrated on densely populated urban areas. However, the shape of these regions are primarily based on the shape of the counties, and it is not particularly interesting. Similarly, the rotation attribute is not applicable for such a model, since the fixed boundaries of counties do not rotate.

- **Epidemics as moving region objects:** The regions affected by epidemics can be represented as moving regions whose shapes change over time as the epidemics spread. The quantification of the area of affected regions, as well as the rate of spread are important factors for the epidemiologists. However, the rotation of the infected regions is not. The shape of the infected region is also not important. For instance, knowing that the epidemic region is sigmoidal or elliptic does not provide any relevant information.

- **Naval ships as moving region objects:** While in many applications ships are designed as moving point objects, the large warships such as aircraft carriers, cruisers, or destroyers can be modeled as moving regions. Unlike earlier examples, their

shape does not change and the area covered by them is not variable. However, their movement and rotation can be of great importance.

- **Tropical cyclones as moving region objects:** Tropical cyclones are very intense low-pressure wind systems, forming over tropical oceans with winds of hurricane force. A tropical cyclone can be modeled as a moving region object, and unlike the previous examples its location, area, shape, and, rotation evolve over time. Depending on the application context, all of these evolution characteristics can be important for the model.

Our algorithms for the discovery of spatiotemporal event sequences are designed for trajectories of moving objects. However, they are primarily formulated for moving region objects whose location, area, shape, and rotation continiously change over time. We model our simple trajectory data type as *evolving region trajectory*. Evolving region trajectory is the trajectory of a moving region object whose spatiotemporal characterstics such as location, area, shape, and rotation continiously evolve over time.

To model the evolving region trajectories, we use the raw trajectory data model [100], which captures the recorded locations (as polygon-based geometries) of objects over time. We model the evolving region trajectories as a list of times and locations. The basic spatiotemporal data abstraction we use is the time-geometry pairs. A time-geometry pair is denoted as tg_i, and consists of a time object (denoted as t_i) and a geometry object representing the spatial location (denoted as g_i).

$$tg_i = \langle t_i, g_i \rangle \tag{3.1}$$

The time object can either be a timestamp or a time interval. A timestamp is a single point in time dimension, which can be represented as a scalar value. On the other hand, a time interval is a time range represented with a start time and end time such that

$t_i = [t_i.start, t_i.end)$, which is a halp-open time interval that does not include its end time.

Then, the evolving region trajectories (denoted as ert_i in Eq. 3.2) are represented as a list of chronologically ordered time-geometry pairs.

$$ert_i = \{\langle t_{i_1}, g_{i_1}\rangle, \langle t_{i_2}, g_{i_2}\rangle, \ldots, \langle t_{i_k}, g_{i_k}\rangle\} \tag{3.2}$$

where $t_{i_1} < t_{i_2} < \ldots < t_{i_k}$. For the case where the time object is represented as a timestamp, the aforementioned inequality is trivial. For the time interval case, $t_{i_j} < t_{i_{j+1}}$ translates to $t_{i_j}.end \leqslant t_{i_{j+1}}.start$ as the time intervals are half-open. Time-geometry pair annotation is a discretized trajectory representation, and we consider that the object's location persists (stays the same) during the time interval shown in a particular time-geometry pair.

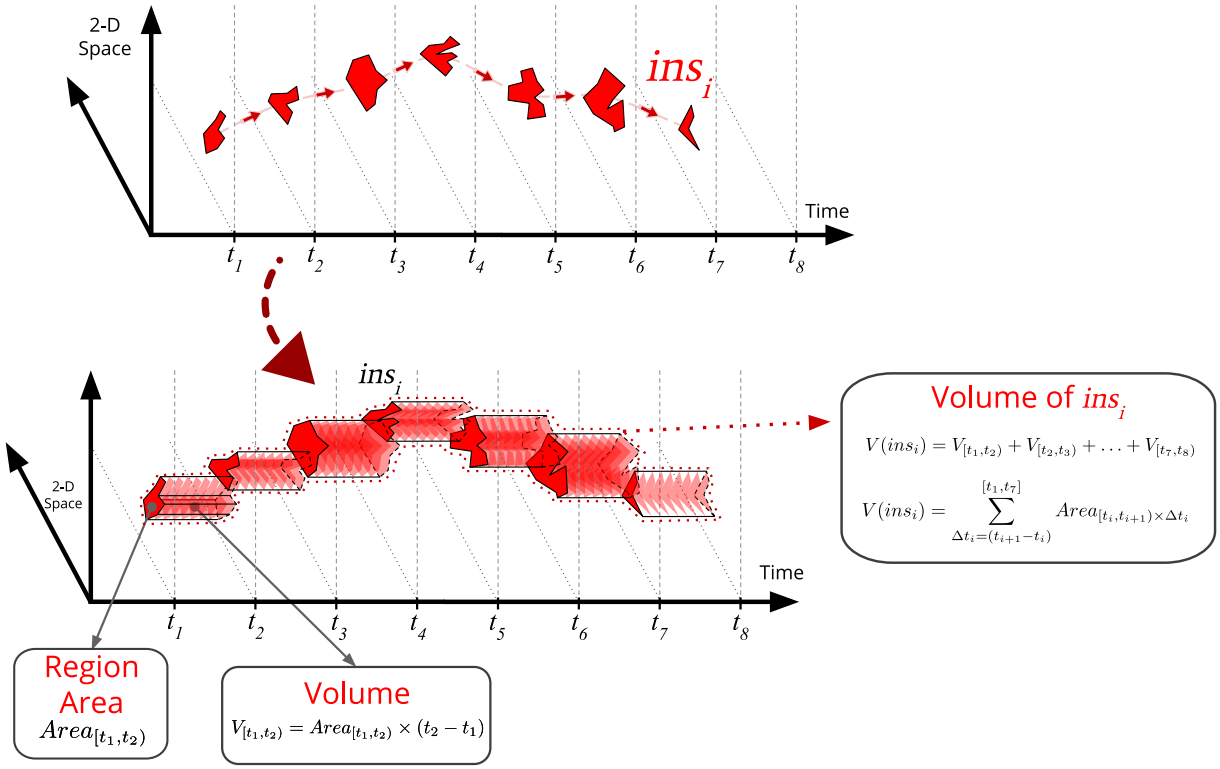


Figure 3.2: Three-dimensional modeling of a spatiotemporal event instance (ins_i) is illustrated with volume calculation from individual time-geometry pairs.

3.3 Modeling Spatiotemporal Event Instances and Examples

The spatiotemporal event instances (denoted as ins_i) are objects of a particular event type, which are the primary subjects of STES mining. An event type is the category, class, or group of the event instances. We model the instances using the evolving region trajectories. A spatiotemporal event instance consists of three attributes: a unique identifier, an event type, and an evolving region trajectory.

$$\begin{aligned} ins_i &= (id, e_i, ert_i) \\ ins_i &= (id, e_i, \{\langle t_{i_1}, g_{i_1}\rangle, \langle t_{i_2}, g_{i_2}\rangle, \ldots, \langle t_{i_k}, g_{i_k}\rangle\}) \end{aligned} \tag{3.3}$$

The instance is an abstraction of an evolving region trajectory with a unique identifier and an event type. Apart from the raw spatiotemporal data associated with the instances, we also have the *lifespan* and the *minimum bounding rectangle* of the instances. These two show the temporal and spatial boundaries of the instances. The lifespan of an instance is the time interval between the start time and end time of the instance.

$$lifespan(ins_i) = [t_{i_1}, t_{i_k}) \qquad \textit{// Following Eq. 3.3} \tag{3.4}$$

The minimum bounding rectangle (MBR) of an instance is the minimum orthogonal rectangle that encloses all the geometries of the instance's trajectory. We can find the MBR by spatially unioning all the geometries of the instance's evolving region trajectory.

$$MBR(ins_i) = \bigcup_{j=\{1,2,\ldots,k\}} g_{i_j} \qquad \textit{// Following Eq. 3.3, where } \bigcup \textit{ is spatial union operator} \tag{3.5}$$

In Figure 3.2, we demonstrate the three-dimensional modeling and spatiotemporal volume transformation of a spatiotemporal event instance. The spatiotemporal volume

of an instance is calculated by summing the volumes of time-geometry pairs during its lifespan (as shown in Eq. 3.6).

$$V(ins_i) = \sum_{\Delta t_i=(t_{i+1}-t_i)}^{[t_s,t_e)} Area_{[t_i,t_{i+1})\times\Delta t_i} \tag{3.6}$$

The volume of an individual time-geometry pair is found by multiplying the *area* of the region geometry by the duration (the length of the time interval) as shown in Eq. 3.7. Note that, for each time-geometry pair, the volume is calculated in a discrete fashion.

$$V_{[t_i,t_{i+1})}(ins_i) = Area_{[t_i,t_{i+1})} \times (t_{i+1} - t_i) \tag{3.7}$$

Note that the $Area_{[t_i,t_{i+1})}$ function returns the area of the geometry, g_i (the geometry at time interval $[t_i, t_{i+1})$).

4 SPATIOTEMPORAL CO-OCCURRENCES AND SIGNIFICANCE MEASUREMENTS

An important aspect of data mining research is the determination of the interestingness of patterns. In classical frequent pattern mining tasks (e.g. shopping basket analysis), the main goal is to identify items frequently appearing together in an itemset. While it seems trivial, such analyses require an appropriate interestingness measure to assess the strength of relationships among different types of items. Measures, such as support, confidence, correlation, and entropy, have been extensively used in frequent pattern mining [101], [102].

Spatial and spatiotemporal extensions of frequent pattern mining presents a similar challenge, where the choice of objective measure may lead to the discovery of inadvisable or uninteresting information depending on the context. Unlike classical frequent pattern mining from binary features, in both spatial and spatiotemporal pattern mining tasks, the spatial or spatiotemporal relationships among items (or instances) are not explicit. Therefore, it is considered necessary to initially transform the implicit spatial and temporal information to a transaction-like embodiment. We will mention the examples of such transformations in the literature in Section 4.1.

There has been extensive research on understanding and assessing the quality, interestingness, and appropriateness of objective measures for different tasks and domains. However, there is no prevalent agreement on selecting the right measure [103]. Selection of the interestingness measure is of great importance, because many measures create conflicting information due to their significantly different properties [104]. Many have agreed there is no universal solution for interestingness measure selection, because the appropriateness of the measures is dependent on the domain and data mining task [105].

In this chapter, we focus on measuring the strength of spatiotemporal co-occurrences in the context of spatiotemporal frequent pattern mining from evolving region trajectories. The spatiotemporal co-occurrence relationship is one of the two predicates of spatiotemporal follow relationship that characterizes the sequence generating behavior in spatiotemporal event sequence mining. We will explain the follow relationship and its relation to the spatiotemporal follow relationship in Chapter 5. The co-occurrence relationship among the event instances is characterized by spatial and temporal overlap. The significance of a co-occurrence indicates the strength of the overlap, and it is primarily utilized for filtering the spurious co-occurrences from the genuine ones.

As shown in Chapter 3, the spatiotemporal event instances (denoted as ins) are represented with evolving region trajectories, and every instance has an event type that represents the class of the instance. Evolving region trajectories are moving region objects whose spatial representations continuously evolve over time. An evolving region trajectory is comprised of a chronologically ordered collection of time-geometry pairs ($tg_i = \langle t_i, g_i \rangle$). Each time-geometry pair represents the region-based location (g_i) of the instance at a particular time (t_i). In Figure 4.1.a and Figure 4.1.b, we illustrate two example spatiotemporal co-occurrences. In Figure 4.1.a, we demonstrate the co-occurrence between two instances – ins_i and ins_j, where their regions spatiotemporally overlap during their entire lifespans. In Figure 4.1.b, we display three co-occurring instances – ins_i,

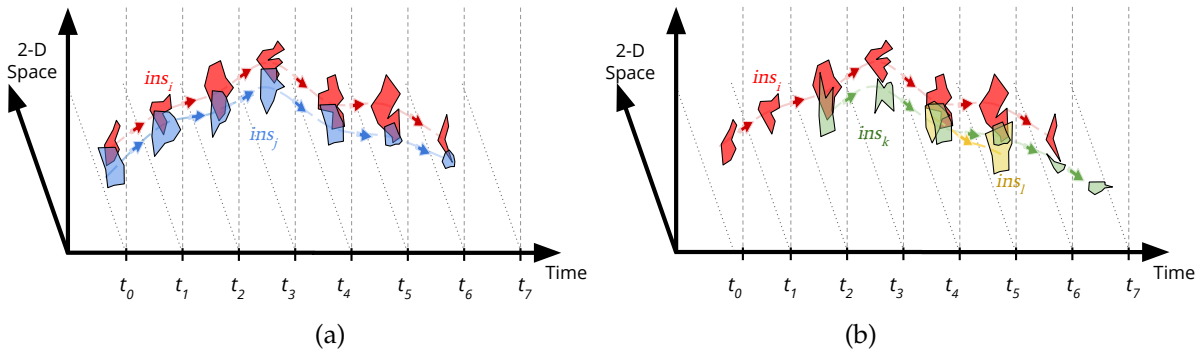


Figure 4.1: Two example spatiotemporal co-occurrences among event instances are shown in (a) and (b). In (a), ins_i co-occurs with ins_j. In (b), three instances (ins_i, ins_j, and ins_k) co-occur.

ins_k, and ins_l. Note that, in Figure 4.1.b, all three instances spatially overlap between t_4 and t_5.

The instances are considered as three-dimensional objects (with one temporal and two spatial dimensions) associated with a spatiotemporal volume. The strength (*i.e.*, significance) of a co-occurrence is measured using the co-occurrence coefficient (denoted as cce), and the co-occurrences are considered as significant, only if they pass the user-determined co-occurrence coefficient threshold (cce_{th}) . In the earlier spatiotemporal co-occurrence pattern mining studies [37], [38], the co-occurrence coefficient is calculated using a spatiotemporal version of *Jaccard* (J) or OMAX measures. Both of these measures utilize the three-dimensional volumes of the co-occurring trajectories when assessing the significance of co-occurrences. However, using the J or OMAX measure leads to unfair assessments in certain cases, which can cause the exclusion of the important co-occurrences and the inclusion of spurious ones. In this chapter, we will develop a novel and more relevant technique for significance measurement of spatiotemporal co-occurrences, which can potentially alleviate these issues.

Although, we highlight the solar event data in our examples, the disproportions of spatiotemporal data are common in nature. For instance, the proliferation and growth of cancer stem cells differs significantly based on the micro-environment in which they reside [106]. Another example is the drastic change of sizes in midget, normal, large, and giant hurricanes and the tropical storms associated with them [107].

The rest of this chapter is organized as follows. We review the related work on spatial close-by and spatiotemporal co-occurrence relationships in literature in Section 4.1. We continue our discussion with STCOP mining and a real life example from solar event data in Section 4.2. In Section 4.3 and 4.4, we explain the J^+ and J^* measures, in detail, with algorithms and their important properties. We present our experimental evaluation in Section 4.6. Finally, in Section 4.7, we will present a brief summary and present our remarks.

4.1 Related Work on Spatiotemporal Data Mining

Below, we will present the related spatial and spatiotemporal data mining studies. In these studies, the implicit spatial and temporal close by relationships are translated into composite transaction-like structures (e.g., co-locations, flocks, episodes, co-occurrences). While the mining subject of these studies are usually distinct for each study, all of them focuses on the spatial or spatiotemporal closeness of objects. We will primarily explore how they formalize the generic close by relationships in their respective studies.

4.1.1 Spatial Co-locations

Spatial association rules are association rules involving spatial relations among spatial objects [108]. Kopersky and Han introduced a *reference feature centric* model for discovering spatial association rules. In reference feature centric model, one or more user-specified reference features are selected and transactions are created based on the spatial proximity of the instances to the reference points. The spatial proximity is defined as the generic close to (`g_close_to`) relationship, which conceptually includes topological relations such as intersection, inside, or close by.

The spatial co-location patterns (or neighboring class sets) represent the subsets of features whose instances are frequently located together [109], [110], [111], [112]. To find the co-located objects, Morimoto introduced a space-driven partitioning strategy [109]. In this strategy, the space is divided into disjoint partitions and spatial instances are considered as a co-location only if they are located in the same space partition. Later, Huang et al. presented an event-centric neighborhooding strategy for co-location pattern mining [110]. The neighborhooding strategy can capture the spatial neighborhoods without specifically determining the reference features or partitioning the space. The possible criteria for forming neighborhoods include spatial (or spatiotemporal) relationships (adjacency, overlap), metric relationships (distance-based approaches), or a combination of

these two. Xiong et al. used a buffer-based event-centric neighborhooding approach for identifying the co-locations of spatial instances with extended geometric representations [111]. In the buffer-based model, given a distance d for forming a buffer, the spatial instances are considered as co-located when their buffers spatially overlap.

Yoo et al. introduced the discovery of co-evolving spatial event sets in [113]. Co-evolving spatial event sets represent the co-location patterns whose prevalence values similarly evolve over time. In this work, the distance-based event centric neighborhooding strategy (from [110]) is used to find co-location patterns. The prevalence of the spatial co-location patterns are measured by the participation index. Participation index is the minimum relative frequency of the participating event types in a pattern. Then, a spatial prevalence time sequence, which is comprised of a sequence of participation index values, is generated for each co-location pattern. To find the similarity between two spatial prevalence time sequences, normalized Euclidean distance is used.

Mining collocation episodes is introduced in [114]. A collocation episode is defined for point-based objects, and it is a sequence of spatial co-location relationships, each describing which pairs of object types are close to each other over a significant time window. The closeness of the co-location sequences are determined using an aggregate distance function defined as either maximum or average pairwise distances between the point-based objects.

4.1.2 Moving Cluster Analysis

The Relative Motion (REMO) framework was developed by Laube and Imfeld to discover the motion patterns in groups of spatiotemporal trajectories [115]. The framework builds a REMO analysis matrix from the point-based spatiotemporal trajectories using motion attributes (*i.e.*, speed, change of speed, and motion azimuth). Later, the REMO analysis matrix is analyzed to find the motion patterns such as constance, concurrence,

and propagation. For example, the concurrence pattern represents a group of objects showing synchronous motion at a time interval.

This work was extended to to include the motion patterns using the spatial neighborhood information in [116] and [117]. In other words, the motion patterns are spatially constrained based on their proximity to generate more complex spatiotemporal patterns that are track, flock, leadership, encounter, and convergence. Gudmundsson et al. formalized the spatial proximity using a circular impact range [116]. For example, the flock pattern is the spatially constrained version of concurrence pattern. Namely, a set of objects is considered to form a flock, if they are within a circular region (of radius r) and they move in the same direction. Apart from the impact range approach, Laube et al. listed three more alternative approaches for spatial proximity constraints: (1) Maximal length of cumulated distances to the mean or median center of the objects. (2) Average length of the Delaunay edges of a group forming a relative motion pattern (3) Maximal border length of the convex hull formed by a group of objects.

Kalnis et al. defined the problem of discovering moving clusters, and proposed clustering-based methods to identify moving clusters [118]. For finding spatial clusters (*i.e.*, objects that are spatially close-by at a particular time) in each spatial snapshot, they use the density-based spatial clustering algorithm, DBSCAN [56]. If there is a large enough number of common spatial objects between two clusters in consecutive time slots, such clusters are called moving clusters. The portion of common objects between two consecutive clusters is measured by a Jaccard-like integrity measure – $\frac{|c_t \cap c_{t+1}|}{|c_t \cup c_{t+1}|}$, where c_t and c_{t+1} denote two consecutive clusters (set oj objects) at times t and $t+1$.

4.1.3 Spatiotemporal Co-occurrences

Celik et al. introduced the mixed-drove spatiotemporal co-occurrence patterns (MDCOPs) in [119]. Similar to Huang et al.'s work on co-location patterns [110], Celik et al. used the distance-based event-centric neighborhooding approach to generate spa-

tiotemporal neighborhoods when mining MDCOPs [119]. In MDCOP mining, the time frames are collapsed, meaning temporal framework is divided into disjoint time frames. For each time frame (1) the event instances are considered to be in temporal neighborhood and (2) the prevalent spatial co-locations, which occur during the same time frame are found. Then, MDCOPs, which can be interpreted as temporally persistent spatial co-location patterns, are determined by checking their temporal persistence (time prevalence). In [119], the time prevalence is measured as the ratio of time frames where a co-location pattern is present to the total number of time frames. In Celik's succeeding work [120], the discovery of partial spatiotemporal co-occurrence patterns (PACOPs) are inspected. PACOPs are very similar to MDCOPs. These two works differ in finding the time prevalence of co-occurrence patterns. When finding PACOPs, the algorithm considers the partially present (*i.e.*, less frequently occurring) object types, and uses temporal participation index when determining the time prevalence. MDCOP mining uses a support-like time prevalence measure, which is based on the frequency, while PACOP mining uses temporal participation index, which is based on the relative participation (frequency).

Pillai et al. introduced spatiotemporal co-occurrence patterns (STCOP) and spatiotemporal co-occurrence rules (STCOR) from datasets with evolving regions [37] and [121]. Similar to the approach in [111], event instances are considered to form a spatiotemporal co-occurrence, if there exists a spatiotemporal overlap among these instances. In contrast to Xiong's approach, a buffer is not used and a spatiotemporal version of Jaccard (J) measure is employed for measuring the significance of the co-occurrences. In addition to the J measure, overlap measures OMIN (Overlap Minimum) and OMAX (Overlap Maximum) and spatiotemporal versions of *Dice*, *Cosine* measures are also used as a filter to the J measure.

Table 4.1: Summary of related work on spatial and spatiotemporal data mining

Pattern Type	Spatial	Temporal	Strategy	Data Type	References
Spatial Association Rules	✓	✗	Reference feature selection with a generic close to relation	Point or Region	[108]
Neighboring Class Sets	✓	✗	Disjoint spatial partitions	Point	[109]
Spatial Co-locations - 1	✓	✗	Event-centric approach based on distance	Point	[110], [112]
Spatial Co-locations - 2	✓	✗	Event-centric approach based on buffer	Point or Region	[111]
Co-evolving spatial event sets	✓	✓	Event-centric approach based on distance and normalized Euclidean for time sequences	Point	[113]
Collocation Episodes	✓	✓	Aggregate distance function (avg. or max. pairwise distance between points)	Point	[114]
Relative Motion Patterns (Flock, Convergence, Leadership etc.)	✓	✓	Relative motion parameters based on the pattern type and spatial proximity constraints (impact range, cumulated distance, Delaunay edges, length of convex hull)	Point	[115], [116], [117]
Moving Clusters	✓	✓	Density based spatial clustering and integrity threshold	Point	[118]
Mixed-drove spatiotemporal co-occurrence patterns	✓	✓	Spatial participation index in collapsed time frames and time prevalence (support-like)	Point	[119]
Partial spatiotemporal co-occurrence patterns	✓	✓	Spatial participation index in collapsed time frames and temporal participation index based on relative frequency	Point	[120]
Spatiotemporal co-occurrence patterns and rules (STCOP/STCOR) from evolving regions	✓	✓	Measure the strength of spatiotemporal overlap using the Jaccard measure	Region	[37], [121]

4.1.4 Summary

While our work is related to spatial co-location mining and discovery of moving clusters, it is primarily associated with STCOP mining. We demonstrate a summary of related studies on spatial and spatiotemporal data mining in Table 4.1. Previous works in spatiotemporal co-occurrence pattern mining can be classified into two categories based on the data types: (1) Patterns discovered from point-based spatiotemporal event instances such as MDCOPs or PACOPs [119] [120]; (2) Patterns discovered from region-based spatiotemporal event instances such as STCOPs or STCORs [37] [121]. Our proposed sig-

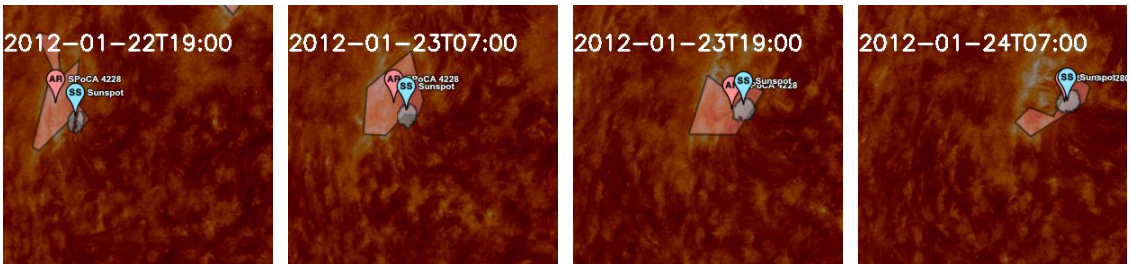


Figure 4.2: Two co-occurring solar event instances (an *active region* and a *sunspot*) reported by Heliophysics Event Knowledgebase [1] between '2012-01-22 19:00' and '2012-01-24 07:00'.

nificance measures are designated for region-based spatiotemporal event instances that form trajectories over time.

4.2 A Real-life Example

Spatiotemporal co-occurrences commonly occur among various types of solar events (or features) such as Active Regions (AR), Coronal Holes (CH), Emerging Flux (EF), Filaments (FI), Flares (FL), Sigmoids (SG), and Sunspots (SS) [122]. The spatial characteristics such as location, shape, and size of solar event instances continuously evolve over time. Figure4.2 demonstrates the spatiotemporal evolution of two co-occurring solar events (one AR and one SS). The regions covered by these solar events are represented as polygons evolving over time. Therefore, the solar events can be modeled as event instances formed by evolving region trajectories.

As mentioned earlier, spatiotemporal co-occurrences appear when there is a spatiotemporal overlap (being at the same location and at the same time) of two or more event instances, and their significance is calculated using the co-occurrence coefficient (*cce*). The *cce* is calculated as the J, OMIN, and OMAX measures [37] [121]. The J measure (shown in Eq. 4.1) is the ratio of intersection volume to the union volume of two or more overlapping instances.

$$J(ins_1, \ldots, ins_n) = \frac{V(ins_1 \cap \ldots \cap ins_n)}{V(ins_1 \cup \ldots \cup ins_n)} \tag{4.1}$$

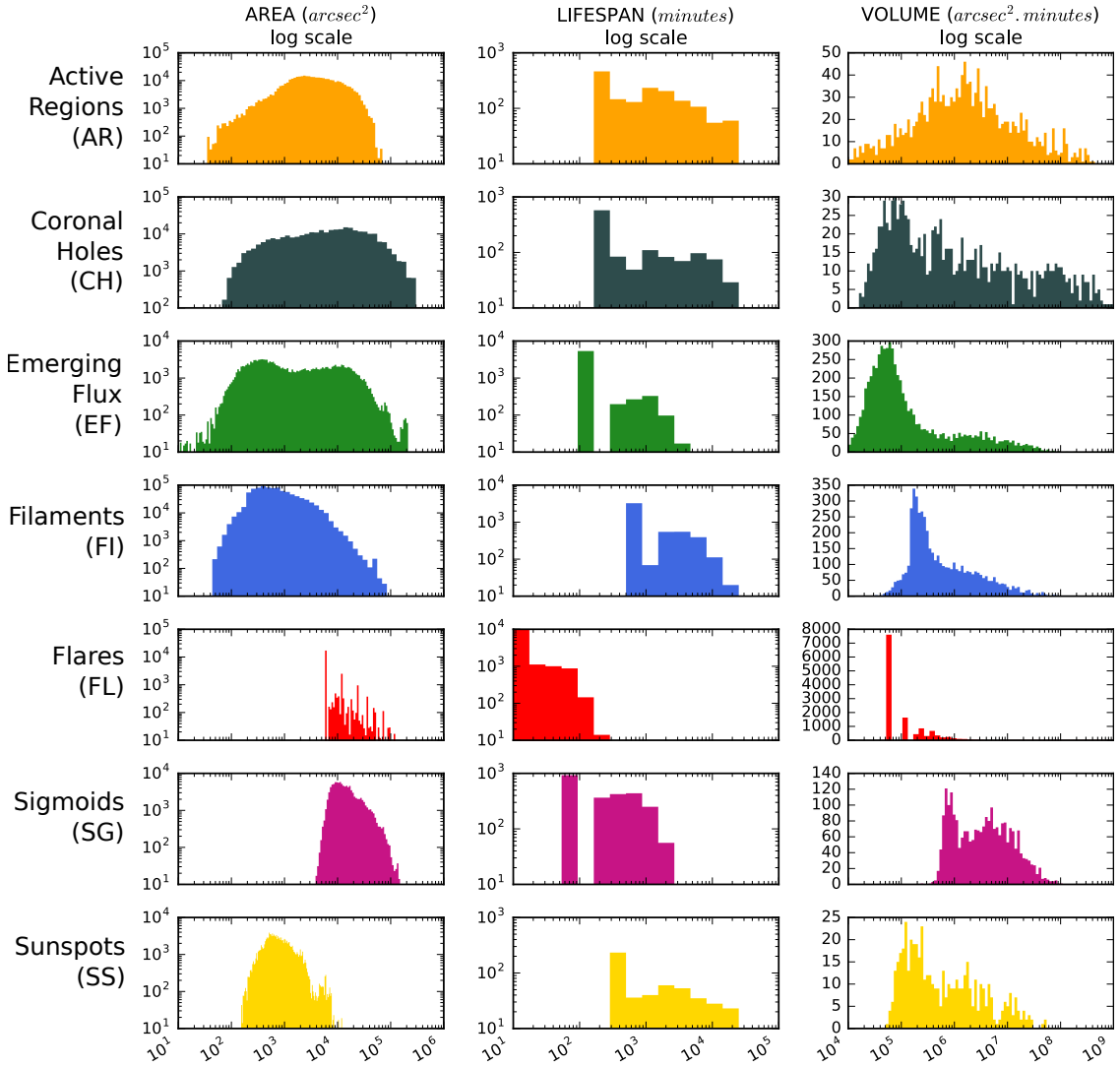


Figure 4.3: Histograms of area, lifespan, and volume for seven different types of solar event instances occurred between January 1, 2012 and December 31, 2012

The intersection volume is calculated from the areas of intersecting regions at times where they spatiotemporally overlap, while the union volume is calculated by spatially unioning the region geometries for all valid time intervals.

The overlap measures OMIN and OMAX (shown in Eq. 4.2 and Eq. 4.3) are calculated as the ratio of intersection volume to the maximum and minimum volume of the instances.

$$\mathrm{OMAX}(ins_1, \ldots, ins_n) = \frac{V(ins_1 \cap \ldots \cap ins_n)}{\max(ins_1, \ldots, ins_n)} \tag{4.2}$$

$$\text{OMIN}(ins_1, \ldots, ins_n) = \frac{V(ins_1 \cap \ldots \cap ins_n)}{\min(ins_1, \ldots, ins_n)} \tag{4.3}$$

The OMIN, OMAX, and J measures output a value on $[0, 1]$ range. The value 0 means there is no co-occurrence, and 1 means the co-occurring trajectories are equal[1] [2].

Figure 4.3 demonstrates the histograms of area, lifespan, and volume of seven different solar event types registered by NASA's Solar Dynamic Observatory telescope, and reported by the Heliophysics Event Knowledgebase [123]. The *area* refers to the region's area for individual time-geometry pairs, while the *lifespan* refers to the time duration between start and end times of the instances.

It can be observed from Figure 4.3 that lifespans, volumes, and areas of the instances exhibit drastic variability. In Figure 4.3, the horizontal axes are in logarithmic scale and shared across the rows in the same column. Events such as sunspots and filaments have very long lifespans, while flares, sigmoids, and emerging flux events have very short ones. On the other hand, the volumes of coronal holes and active regions are very large compared to flares and emerging flux events. The spatiotemporal co-occurrence of large volume event instances with smaller ones leads to very large union volumes. However, the intersection volume is limited to the volume of the smaller event (See the J and OMAX measure in Eq. 4.1). Furthermore, the events with longer lifespans and larger areas have higher chances of co-occurring with other events. These situations breed anomalies when assessing the strength of spatiotemporal co-occurrences.

Consider the following example scenarios to see the need for a different approach when calculating the co-occurrence coefficient in spatiotemporal co-occurrence mining.

Example 1 – Coverage anomaly: In Figure 4.4.a, two event instances (ins_A and ins_B) are demonstrated. ins_A has small area and shorter lifespan, and ins_B has larger area

1 The equality of trajectories means that all of the time-geometry pairs in trajectories are equal.
2 If an instance is completely covered by another one the OMIN outputs 1

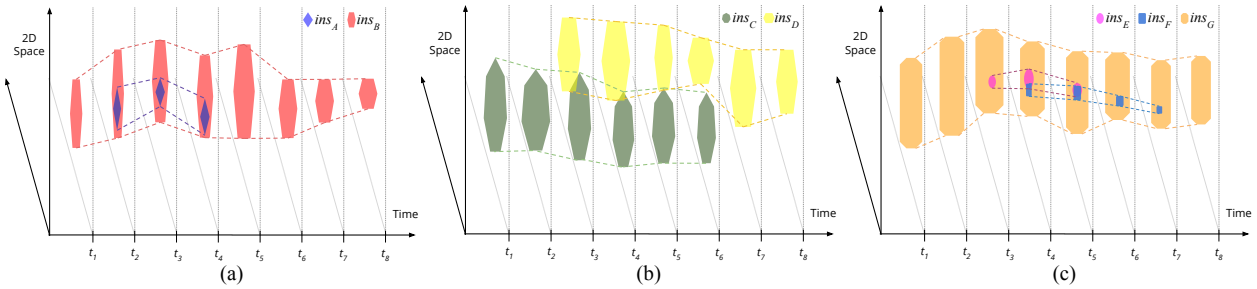


Figure 4.4: The illustration of three possible scenarios for spatiotemporal co-occurrences of instances that can occur among event instances with unbalanced characteristics.

and longer lifespan. Since ins_A completely covers ins_B, we cannot have a stronger spatiotemporal overlap given the state of those two instances. However, the J or OMAX values for ins_A and ins_B are unfairly affected by the large union volume caused by ins_B, even though ins_A strongly overlaps with ins_B throughout its entire lifespan.

Example 2 – Large volume bias: In Figure4.4.b, two event instances (ins_C and ins_D) are depicted. ins_C and ins_D have both large areas and long lifespans. Their region geometries slightly overlap only at three timestamps, which is a weaker co-occurrence when compared to the examples in Fig 4.4.a and Fig 4.4.c. Even though their spatiotemporal co-occurrence is limited to a small portion of their longer lifespan, the J or OMAX values for these two instances will tend to be larger because of two reasons: (1) in a fixed spatial and temporal window, instances with larger areas or longer lifespans have higher chances of spatiotemporal overlap, and (2) J and OMAX values tend to go higher as the intersection volume of these two instances is likely higher.

Example 3 – Favoring the similar: In Figure4.4.c, three event instances (ins_E, ins_F and ins_G) are demonstrated. ins_E and ins_F have both small areas and relatively shorter lifespans. On the other hand, ins_G has moderate area but longer lifespan. For the co-occurrence of ins_E and ins_F, the J value is higher as their union volume is not high. However, for the co-occurrence of ins_E, ins_F, and ins_G, the J value stays small because of larger union volume caused by ins_G. Similarly, the OMAX value is also small because of the large volume of ins_G. Nevertheless, both ins_E and ins_F strongly co-occur with ins_G.

A similar problem is also present for *Example 2*, the instances with similar spatiotemporal characteristics are unfairly favored by the J or OMAX measures especially when a fixed threshold is used for pruning supposedly unimportant co-occurrences.

It can be seen that the J and OMAX measures tend to favor the event instances with similar characteristics. Small volume instances have higher chances of having a strong co-occurrence with each other. Similarly, large volume instances will have higher chances of having a strong co-occurrence with large volume instances. Given the unbalanced nature of solar data (or more generally, scientific data) and possibly disregarded but important co-occurrences among these instances, it is necessary to develop novel techniques for the significance assessment of co-occurrences.

4.3 Evolution of Spatiotemporal Jaccard Measure

4.3.1 Preliminaries

The support (denoted as *supp* in Eq. 4.4) measure for an association rule in classical frequent itemset mining is the fraction of transactions that includes all the participating item types (denoted as I_i) in the entire database [124]. Support is usually used for assessing the significance of a pattern or an association rule and it represents the joint probability of two or more item types in a sample dataset.

$$\text{supp}(I_1, I_2, \ldots, I_n) = P(I_1 \cap I_2 \cap \ldots \cap I_n) \tag{4.4}$$

The Jaccard similarity coefficient has been extensively used for measuring the similarity among item types (in shopping basket analysis) [125], documents (in text mining) [126] [127], or spatial feature types and objects [128] [36], [37], [38]. Following the

item type representation in Eq. 4.4, the Jaccard similarity coefficient ($\mathtt{Jaccard}$) for itemsets is calculated as follows:

$$\mathtt{Jaccard}(I_1, I_2, \ldots, I_n) = \frac{P(I_1 \cap I_2 \cap \ldots \cap I_n)}{P(I_1 \cup I_2 \cup \ldots \cup I_n)} \tag{4.5}$$

The generalized version of Jaccard similarity coefficient (in Eq. 4.5) can be expressed as Steinhaus index [129]. Given a measurable space, and a measurement function μ, Steinhaus index is defined as follows:

$$\mathtt{Steinhaus}(I_1, I_2, \ldots, I_n) = \frac{\mu(I_1 \cap I_2 \cap \ldots \cap I_n)}{\mu(I_1 \cup I_2 \cup \ldots \cup I_n)} \tag{4.6}$$

For the case of classical Jaccard similarity coefficient (in Eq. 4.5), the cardinality of a given sample set is the measurement function. In STCOP mining [37], a spatiotemporal version of Jaccard measure (*i.e.*, the J measure) is used for assessing the strength of a spatiotemporal co-occurrence. The J is a version of Steinhaus index, where the measurement function (μ) is the volume function (V) presented in Eq. 3.6. In Eq. 4.1, the measurement function, V, calculates the intersection and union volumes of trajectory-based event instances.

4.3.2 *Intermediate Form: J^+ Measure*

Three problems associated with the J measure are mentioned in Section 4.2 with example scenarios. One intuitive solution for alleviating the problems addressed in Section 4.2 is to modify the measurement function (μ) to eliminate the segments of trajectories when calculating the cce. The criterion for elimination that we employ is the existence of spatiotemporal co-occurrence (*i.e.*, spatiotemporal overlap relationship) among the instances. Using the overlap-based criterion can help us focus on segments of trajectories, where co-occurrences appear.

Let J^+ be an extended version of Jaccard measure. We define J^+ as follows:

$$J^+(ins_1, \ldots, ins_n) = \frac{V_{til^{co}}(ins_1 \cap \ldots \cap ins_n)}{V_{til^{co}}(ins_1 \cup \ldots \cup ins_n)} \tag{4.7}$$

Here, the measurement function of J (that is V in Eq. 4.1), is replaced by an interval volume function – $V_{til^{co}}$. $V_{til^{co}}$ measures the volume of intersection and union at times where there exists a spatiotemporal overlap among all the instances.

Definition 1. Interval volume function, V_{til}, calculates the volume of given trajectory-based geometries only for the time intervals given in a time interval list, denoted as til.

For a given trajectory-based instance ins_i, interval volume function is calculated using V_{til} (Eq. 4.8). It calculates the volume of the trajectory-based instance only for the intervals specified in the time interval list (til).

$$V_{til}(ins_i) = \sum_{[\tau_k, \tau_{k+1}) \in til} Area_{\tau_k}(ins_i) \times (\tau_{k+1} - \tau_k) \tag{4.8}$$

Definition 2. Time interval list (til) is a list of ordered time intervals. Each time interval is defined by a pair of timestamp values (t_i, t_j), where $t_i < t_j$; for each i, j; $1 \leqslant i < j \leqslant n$

$$til = \{(t_1, t_2), (t_3, t_4), \ldots, (t_{n-1}, t_n)\} \tag{4.9}$$

Definition 3. For a given set of event instances S (where $S = \{ins_1, \ldots, ins_n\}$ and $n \geqslant 2$), the co-occurrence time interval list (denoted as til^{co}) contains time intervals where there exists a spatiotemporal overlap among *all* the instances in S.

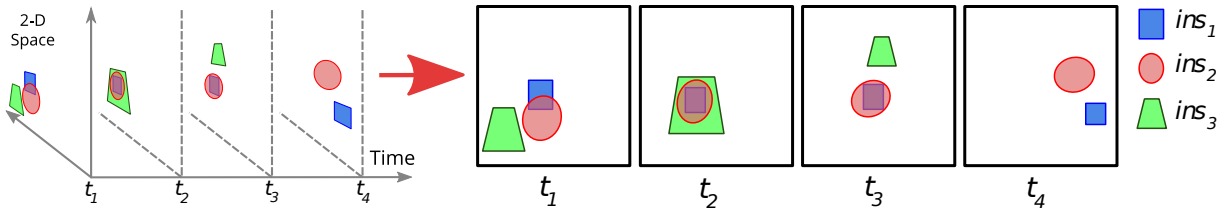
In Figure 4.4, we illustrated three example co-occurrences. To better explain the concept, we will present the co-occurrence time intervals for each of these co-occurrences.

- In Figure 4.4.a, the til^{co} for ins_A and ins_B is $[t_2, t_4]$ as these two instances overlap between these time intervals.
- Similarly, in Figure 4.4.b, the til^{co} for ins_C and ins_D is $[t_3, t_5]$.
- In Figure 4.4.c,
  - til^{co} for ins_E and ins_F is $[t_4, t_5]$.
  - til^{co} for ins_E and ins_G is $[t_3, t_5]$.
  - til^{co} for ins_F and ins_G is $[t_4, t_7]$.
  - til^{co} for ins_E, ins_F, and ins_G is $[t_4, t_5]$ as between t_4 and t_5 all three of ins_E, ins_F, and ins_G spatiotemporally overlap. This is essentially the intersection of til^{co}'s of (ins_E, ins_F), (ins_E,ins_G), and (ins_F, ins_G).

For the J^+ measure, the amount of geometric calculations (*i.e.*, determining the union and intersection of instances, calculating the areas and volume) is limited to the co-occurrence time intervals. Space requirement is also reduced, because for each co-occurrence the intersection and union geometries are included only if there exists a spatiotemporal overlap among all participating instances. However, the J^+ measure has potential drawbacks regarding the filtering mechanism (*i.e.*, elimination of particular segments of spatiotemporal instances).

For a spatiotemporal co-occurrence which has three or more participating instances, co-occurrence time interval list only includes the time intervals in which trajectories of *all* participating instances overlap. Nevertheless, any event of co-occurrence between two instances is a region of interest, and should be considered. These regions can be disregarded when calculating the J^+. Another problem stemming from the same issue is the antimonotonic property of J^+. STCOP mining algorithms efficiently employ downward closure property, and require the significance measures to carry antimonotonic property.

Lemma 1. J^+ *measure is not antimonotonic.*



Area	t_1	t_2	t_3	t_4
ins_1	75	80	70	70
ins_2	80	90	85	95
ins_3	100	120	60	N/A
$(ins_1 \cap ins_2)$	10	80	70	0
$(ins_1 \cap ins_3)$	0	80	0	N/A
$(ins_2 \cap ins_3)$	0	90	0	N/A
$(ins_1 \cup ins_2)$	145	90	85	165
$(ins_1 \cup ins_3)$	175	120	130	N/A
$(ins_2 \cup ins_3)$	180	120	145	N/A
$(ins_1 \cap ins_2 \cap ins_3)$	0	80	0	N/A
$(ins_1 \cup ins_2 \cup ins_3)$	245	120	145	N/A

Figure 4.5: An example co-occurrence of three spatiotemporal instances with the area values at particular timestamps.

Proof. We will present a proof by contradiction. Assume that J^+ measure is antimonotonic. In Figure4.5, the locations of three spatiotemporal instances for four different timestamps are demonstrated, as well as the corresponding area values of instances, their intersections, and unions for each timestamp. Let the time difference between each timestamp be (τ=t_i–t_{i-1}). Then, the J^+ value of any spatiotemporal co-occurrence of two instances (*i.e.*, $J^+(ins_1, ins_2)$, $J^+(ins_1, ins_3)$, $J^+(ins_2, ins_3)$) must be greater than or equal to the J^+ value of any spatiotemporal co-occurrence of three instances (*i.e.*, $J^+(ins_1, ins_2, ins_3)$).

$$J^+(ins_1, ins_2) = \frac{\tau(10+80+70)}{\tau(145+90+85)} = 0.5$$

$$J^+(ins_1, ins_2, ins_3) = \frac{\tau(80)}{\tau(120)} \simeq 0.66$$

$$J^+(ins_1, ins_2) < J^+(ins_1, ins_2, ins_3)$$

This contradicts with the earlier assumption of J^+ measure being antimonotonic as the value of the measure for this particular example increased as the cardinality of the co-occurrence increase. Therefore, the J^+ is not an antimonotonic measure. □

4.4 J^* Measure

A problem with the J^+ is that it does not consider the spatiotemporal co-occurrences appearing in the subsets of participating instances, but only considers the intervals that *all* participating instances spatiotemporally overlap. For the example shown in Figure 4.5, the J measure would incorporate all the geometries (from t_1 to t_4) for all instances when calculating the union volumes (whether there exists a co-occurrence or not). On the other end of spectrum, the J^+ measure only calculates the union volume for the co-occurrence among all the participating instances, but does not reflect any information about the co-occurrences among the subsets of the participating instances. Accordingly, to eradicate the problems regarding the J measure and avoid the neglect of the subset co-occurrences, we introduce the concept of *cross co-occurrence*s, which provides foundation for the antimonotonic J^* measure.

Definition 4. A set of *cross co-occurrence*s (xco) in a spatiotemporal co-occurrence is the spatiotemporal overlap relationships, which occurred among the 2-subsets of participating instances.

Definition 5. For a given set of event instances S (where $S = \{ins_1, \ldots, ins_n\}$ and $n \geqslant 2$), cross co-occurrence time interval list (denoted as til^{xco}) contains time intervals where there exists a spatiotemporal overlap among *at least two* instances in S. Alternatively, let SubS be a 2-subset of S such that $SubS = \{ins_{i_1}, ins_{i_2}\}$, where $1 \leqslant i_1 < i_2 \leqslant n$. Then,

cross co-occurrence time interval list for S is the temporal union of the co-occurrence time interval lists of each 2-subset of P.

$$\mathrm{til}^{\mathrm{xco}} = \bigcup_{\mathrm{SubS} \subseteq \mathrm{S}} \mathrm{til}^{\mathrm{co}}(\mathrm{SubS}) \tag{4.10}$$

where $\mathrm{til}^{\mathrm{co}}(\mathrm{SubS})$ denotes the co-occurrence time interval list of instances in SubS.

Let J^* be a significance measure for spatiotemporal co-occurrences. We define the J^* measure using the interval volume function (in Def. 1) and the cross co-occurrence time interval list (in Def. 5) as follows:

$$J^*(ins_1, \ldots, ins_n) = \frac{V_{\mathrm{til}^{\mathrm{xco}}}(ins_1 \cap \ldots \cap ins_n)}{V_{\mathrm{til}^{\mathrm{xco}}}(ins_1 \cup \ldots \cup ins_n)} \tag{4.11}$$

While the J, J^+ and J^* measures might seem similar in notation, they are considerably different because of the variations stemming from the interpretations of *co-occurrence* and *cross co-occurrence* time interval lists in their respective volume functions. For clarification, we present an example following the instances shown in Figure 4.5. The J^* and J^+ values are equal for size-2 co-occurrences because cross co-occurrence time intervals are the same with co-occurrence time intervals. However, the J^* for size-3 spatiotemporal co-occurrence of ins_1, ins_2, and ins_3 is different from J^+ and thus, calculated as follows. Firstly, the co-occurrence time intervals ($\mathrm{til}^{\mathrm{co}}$) for each 2-subset are determined.

- For ins_1 and ins_2 – $\mathrm{til}^{\mathrm{co}}(ins_1, ins_2) = [t_1, t_4)$
- For ins_1 and ins_3 – $\mathrm{til}^{\mathrm{co}}(ins_1, ins_3) = [t_2, t_3)$
- For ins_2 and ins_3 – $\mathrm{til}^{\mathrm{co}}(ins_2, ins_3) = [t_2, t_3)$

Then, the cross co-occurrence time interval list $\mathrm{til}^{\mathrm{xco}}(ins_1, ins_2, ins_3)$ is

$$\mathrm{til}^{\mathrm{xco}}(ins_1, ins_2, ins_3) = \mathrm{til}^{\mathrm{co}}(ins_1, ins_2) \cup \mathrm{til}^{\mathrm{co}}(ins_1, ins_3) \cup \mathrm{til}^{\mathrm{co}}(ins_2, ins_3) = [t_1, t_4)$$

For the cross co-occurrence time interval, $[t_1, t_4)$,

- The intersection volume is $V_{\{[t_1,t_4)\}}(ins_1 \cap ins_2 \cap ins_3) = \tau(0 + 80 + 0)$
- The union volume is $V_{\{[t_1,t_4)\}}(ins_1 \cup ins_2 \cup ins_3) = \tau(245 + 120 + 145)$.
- Then, the J^* value is $J^*(ins_1, ins_2, ins_3) = \dfrac{V_{\{[t_1,t_4)\}}(ins_1 \cap ins_2 \cap ins_3)}{V_{\{[t_1,t_4)\}}(ins_1 \cup ins_2 \cup ins_3)} = \dfrac{80\tau}{510\tau} \simeq 0.16$

Earlier, we calculated the J^+ value for the same instances as 0.66. The intersection volume remains unchanged; however, as co-occurrence time interval is $[t_2, t_3)$, the union volume for J^+ is calculated as 120τ, which is different from 510τ for J^*.

The J^* measure, unlike the J measure, does not particularly favor the instances, which carry similar volume characteristics. It acknowledges the spatiotemporal co-occurrence of all participating instances as the main event of interest, while also considering the cross co-occurrences appearing among the subsets of participating instances. Regardless of unbalanced characteristics instances may have, the regions of interest for J^* measure is only limited to co-occurrence and cross co-occurrences. Therefore, the J^* can be considered less biased when handling the coverage anomalies and bias created in the co-occurrences by small or large volume instances.

Another important aspect of the problem is the storage requirements and computational complexity. For all measures derived from Jaccard (J, J^+, J^*), the numerator in the ratio is the volume of the spatiotemporal intersection of all instances. However, the denominator, which calculates union volume, changes drastically. Especially, for long-lasting events, storage of the union volumes may create huge storage overhead. From a practical point of view, storing only cross co-occurrences can greatly reduce storage requirements. The geometric calculations for determining unions and intersections are typically very expensive operations. Theoretically, the upper bound of geometric calculations for the J, J^+, and J^* measures are the same (Consider the case where all instances overlap at all time intervals).

4.4.1 *Algorithms for J* Calculation*

We introduce two algorithms for calculating the J* measure. Our first algorithm (shown in Algorithm1) is designed for calculating J* measure for two event instances. The second one is the generalized algorithm (shown in Algorithm2). It calculates the J* measure for two or more event instances. In both of our algorithms, we consider instances are modeled as a list of timestamp-geometry pairs [100]. The list of auxiliary functions used in J* calculation algorithms are listed, and their descriptions are demonstrated in Table 4.2.

Table 4.2: Auxiliary functions used in J* calculations

Function	Description
`FindCoexistence(`ins_1`,`ins_2`)`	The function returns a coexistence time interval list (til^{ce}) that contains intervals where the lifespans of two given instances (ins_1, ins_2) temporally overlap.
`FindCrossCooccurrence(`$instances$`)`	The function returns the cross co-occurrence time interval list (til^{xco} - See Def. 5) of a set of instances.
⟨ins⟩`.GetGeometryAt(`ivl`)`	This function is applied to a spatiotemporal instance (ins). It returns the region geometry of the instance at the given time interval (ivl).
⟨ins⟩`.GetTimeIntervals()`	The function returns the set of uniformly sampled time intervals of the instance (ins).
⟨Collection⟩`.Insert(`$item$`)`	The function inserts an $item$ to a collection.
`Intersection(`$geometries$`)`	The function returns the spatial intersection geometry of a given collection of geometries.
`Union(`$geometries$`)`	The function returns the spatial union geometry of a given collection of geometries.
`Intersects(`g_1`, `g_2`)`	The function returns *true* if two given geometries (g_1, g_2) spatially intersects; otherwise, returns *false*.
`Area(`$geometry$`)`	The function returns the area of the given geometry.

In Algorithm 1, two spatiotemporal instances are given as input. Initially, intervals where the lifespans of two instances overlap are discovered (*i.e.*, coexistence time interval list – til^{ce}). Then, for each interval in til^{ce}, we find the intersection area. If the geometries intersect ($iArea > 0$) at a given time interval, we calculate the union area. Later, we

increase the intersection and union volumes using the intersection and union areas. If there is no spatiotemporal intersection between two instances, the algorithm returns 0; else, it returns the ratio between intersection and union volumes.

Algorithm 1: J^* calculation for two spatiotemporal instances

Input: Two spatiotemporal instances – ins_i, ins_j (Instances are assumed to have the same sampling intervals and phases.)
Output: J^* value for ins_i and ins_j – $J^*(ins_i, ins_j)$

```
1  Algorithm J*_2(ins_i, ins_j)
2      iVolume ← 0 ; uVolume ← 0 ;
3      til^ce ← FindCoexistence(ins_i, ins_j)
4      foreach ivl in til^ce do
5          g_i ← ins_i.GetGeometryAt(ivl) ;
6          g_j ← ins_j.GetGeometryAt(ivl) ;
7          iGeom ← Intersection(g_i, g_j) ;
8          iArea ← Area(iGeom);
9          if iArea > 0 then // calculate union volume if geometries intersect
10             uGeom ← Union(g_i, g_j);
11             uArea ← Area(uGeom);
               // calculate and add intersection and union volumes
12             iVolume = iVolume + iArea * ivl.length;
13             uVolume = uVolume + uArea * ivl.length;
14     if iVolume = 0 then
15         return 0
16     else
17         return iVolume/uVolume
18 Procedure FindCoexistence(ins_i, ins_j)
       // til_i and til_j are the valid time interval sets of ins_i and ins_j
19     til_i ← ins_i.GetTimeIntervals();
20     til_j ← ins_j.GetTimeIntervals();
21     return til_i ∩ til_j
```

In Algorithm2, an initial cross co-occurrence time interval list (til^{xco}) is found using the procedure, `FindCrossCooccurrence`. This procedure iterates over each 2-subsets of given instances. Firstly, for every possible pair, the procedure finds the coexistence time intervals, and later discovers the co-occurrence time intervals by checking the spatial overlap (See Def. 3 - til^{xco}). The union of all co-occurrence time intervals gives the

cross co-occurrence time intervals. After discovering $\mathtt{til}^{xco}$, the algorithm iterates over the intervals in $\mathtt{til}^{xco}$ for volume calculations. In each iteration, intersection and union areas are found, and intersection and union volumes are increased accordingly. If there is no spatiotemporal intersection among the instances, the algorithm returns 0; else, it returns the ratio between intersection and union volumes.

In a nutshell, our algorithms initially determine the temporal co-existence, and later check for the spatial overlap between individual geometries. Therefore, we eliminate the computationally expensive spatial intersection and union operation when they are not necessary. Both of our algorithms effectively calculate the intersection and union volumes at cross co-occurrence time intervals. In Algorithm 1, for two instances, co-occurrence and cross co-occurrence time intervals are the same, and the volumes are simultaneously discovered. In Algorithm 2, the cross co-occurrence time intervals are discovered in advance. Later, the intersection and union volumes are calculated.

4.4.2 *Key Properties of J^**

In this section, two key properties of J^*, which are related to spatiotemporal frequent pattern mining, will be discussed. The first one is the antimonotonic property, which is vital for efficiency and correctness of STCOP mining. The second one is the containment property, which shows the relation between the J, J^+, and J^* measures.

Antimonotonic Property

Downward closure property (i.e. antimonotonicity) is the fundamental aspect of many objective measures used in frequent pattern mining. Similar to the seminal frequent pattern mining approaches [130] [131], in STCOP mining (which are based on the Apriori algorithm [130]) the antimonotonicity plays a significant role for efficiently and correctly mining the co-occurrences.

Lemma 2. *J^* is an antimonotonic measure.*

Algorithm 2: Generalized J^* Calculation

Input: A collection of k event instances – $\mathbb{I} = \{ins_1, ins_2, \ldots ins_k\}$
Output: J^* value for instances in $\mathbb{I}$ – $J^*(ins_1, ins_2, \ldots ins_k)$

```
1  Algorithm J*(𝕀)
2      iVolume ← 0 ; uVolume ← 0 ;
3      til^xco ← FindCrossCooccurrence(𝕀);
4      foreach ivl in til^xco do
5          geometries ← {};
6          foreach ins in 𝕀 do
7              geometries.Insert(ins.GetGeometryAt(ivl)) ;
8          iGeom ← Intersection(geometries) uGeom ← Union(geometries);
9          iArea ← Area(iGeom) uArea ← Area(uGeom)
            iVolume = iVolume + iArea * ivl.length;
10         uVolume = uVolume + uArea * ivl.length;
11     if iVolume = 0 then
12         return 0
13     else
14         return iVolume/uVolume
15 Procedure FindCrossCooccurrence(𝕀)
16     til^xco ← {} ;
       // For any instance pair combination (i.e. 2-subset) (ins_i, ins_j) of 𝕀
       // find co-occurrence time intervals (See Def. 3)
17     foreach (ins_i, ins_j) in 𝕀 do
18         til ← FindCoexistence(ins_i, ins_j) ;
19         foreach ivl in til do
20             g_i ← ins_i.GetGeometryAt(ivl) ;
21             g_j ← ins_j.GetGeometryAt(ivl) ;
22             if Intersects(g_i, g_j) then // if spatially intersects, then add
                interval
23                 til^xco ← til^xco ∪ ivl ;
24     return til^xco
```

Proof. Let S be the set of participating instances of a spatiotemporal co-occurrence ($S = \{ins_1, \ldots, ins_n\}$). Let ins_{n+1} be another instance that forms a co-occurrence with all the instances in S. Then $S' = \{ins_1, \ldots, ins_n, ins_{n+1}\}$ and $S \subset S'$. Then, $J^*(S) \geqslant J^*(S')$, because

1. $V_{til^{xco}}(ins_1 \cap \ldots \cap ins_n \cap ins_{n+1}) \leqslant V_{til^{xco}}(ins_1 \cap \ldots \cap ins_n)$. The intersection volume can only decrease or stay the same with the addition of a new instance to the participating instance set.

2. $V_{til^{xco}}(ins_1 \cup \ldots \cup ins_n \cup ins_{n+1}) \geqslant V_{til^{xco}}(ins_1 \cup \ldots \cup ins_n)$. The union volume can only increase or stay the same with the addition of a new instance to the participating instance set. Note that, the cross co-occurrence time interval list of S' includes at least the cross co-occurrence time interval list of S, and it can potentially include cross co-occurrences between ins_n and the instances in S ($S.til^{xco} \subseteq S'.til^{xco}$). Therefore, the union volume for S' is greater than or equal to the union volume of S.

J^* value for a co-occurrence decreases or stays the same with the addition of a new spatiotemporal instance, as the intersection volume can only decrease or stay the same and the union volume can only increase or stay the same. Hence, J^* is an antimonotonic measure. □

Containment Property

The interestingness of discovered patterns is an important aspect of the data mining research. The concept of interestingness for a pattern includes characteristics such as conciseness, generality, surprisingness and novelty [103]. The J measure can be considered to provide generality as a large fraction of discovered knowledge matches the well-known patterns. However, its ability to discover unexpected or obscure co-occurrences is limited due to the shortcomings we have addressed in Section 4.2. With the J^+ and J^* measures, we aim to achieve novelty and possible surprisingness (meaning not known before, or contradicting the existing knowledge), while preserving generality.

The containment relationship between the J, J^*, and J^+ measures dictates that for any spatiotemporal co-occurrence, the value of the J is always less than or equal to the value of J^* and the value of the J^* is always less than or equal to the value of J^+. The con-

tainment relationship is important, as it helps to maintain the desired generality related characteristics of J measure. The property can be described as follows: Given a particular co-occurrence coefficient threshold, if a spatiotemporal co-occurrence is assessed as significant based on J value, it is also significant based on J^* value. Similarly, if a spatiotemporal co-occurrence is assessed as significant based on J^* value, it is also significant based on J^+ value

Lemma 3. J^+ *measure contains* J^* *measure, and* J^* *measure contains* J *measure.*

Proof. Let S be the set of participating instances of a spatiotemporal co-occurrence ($S = \{ins_1, \ldots, ins_n\}$). The J value for a particular co-occurrence can never be greater than J^* value, and the J^* value can never be greater than J^+ value ($J^+(S) \geqslant J^*(S) \geqslant J(S)$) and , because:

1. $V_{til^{co}}(ins_1 \cap \ldots \cap ins_n) = V_{til^{xco}}(ins_1 \cap \ldots \cap ins_n) = V(ins_1 \cap \ldots \cap ins_n)$. Intersection volumes calculated for both measures are equal, as til^{co} and til^{xco} include the interval of the co-occurrence of all the participating instances.

2. $V_{til^{xco}}(ins_1 \cup \ldots \cup ins_n) \leqslant V(ins_1 \cup \ldots \cup ins_n)$. Union volume calculated for J^* is less than or equal to the union volume calculated for the J measure, since the $V_{til^{xco}}$ function only calculates the union volume for the time intervals specified in til^{xco}, and the intervals in til^{xco} is a subset of the intervals specified by the lifespans of all participating instances.

3. Similarly, $V_{til^{co}}(ins_1 \cup \ldots \cup ins_n) \leqslant V_{til^{co}}(ins_1 \cup \ldots \cup ins_n)$. Union volume calculated for J^+ is less than or equal to the union volume calculated for the J^* measure, since the til^{xco} is a superset of til^{co}.

For any spatiotemporal co-occurrence, the $J^+ \geqslant J^* \geqslant J$ because intersection volumes are the same for all the measures and the union volumes have the following relationship $V_{til^{co}} \leqslant V_{til^{xco}} \leqslant V$. Hence, J is contained by J^* and J^* is contained by J. □

4.5 Algorithms for J, J^+, OMAX and OMIN Calculations

In this section, we will present the algorithms for J, J^+, OMAX and OMIN. The presented algorithms are for the generalized version of the measures, which is suited for k instances ($k \geqslant 2$).

4.5.1 J Calculation Algorithm

We present the generalized J calculation algorithm (for k instances) in Algorithm 4.1. The algorithm initially determines the union time interval list (til^{union}) (union of the time intervals for all the instances) and coexistence time intervals. Then, for each interval in til^{union}, the geometries of all the instances are collected in *geometries* list. Then, union and intersection area and volumes are found for a particular interval. The total intersection and union volume is increased accordingly. Note that the intersection volume calculation is filtered using the coexistence time intervals to create a more efficient algorithm.

Algorithm 4.1 Generalized J Calculation

Input: A collection of k spatiotemporal instances – $\mathbb{I} = \{ins_1, ins_2, \ldots ins_k\}$ (Instances are assumed to have the same sampling intervals and phases.)
Output: J value for instances in $\mathbb{I}$ – $J(ins_1, ins_2, \ldots ins_k)$

```
25 Algorithm J(𝕀)
26     iVolume ← 0 ; uVolume ← 0 ; til^union ← {}   foreach ins_i in 𝕀 do
27         til^union ← TimeIntervalUnion(til^union, ins_i.GetTimeIntervals())
28     til^ce ← FindCoexistence(𝕀)
29     foreach ivl in til^union do
30         geometries = []   foreach ins_i in 𝕀 do
31             geometries.Insert(ins_i.GetGeometryAt(ivl))
32         if ivl in til^ce then
33             iVolume ← iVolume + Area(Intersection(geometries)) * ivl.length
34         uVolume ← uVolume + Area(Union(geometries)) * ivl.length
35     if iVolume = 0 then
36         return 0
37     else
38         return iVolume/uVolume
```

Algorithm 4.2 Generalized J^+ Calculation

Input: A collection of k spatiotemporal instances – $\mathbb{I} = \{ins_1, ins_2, \ldots ins_k\}$
Output: J^+ value for instances in $\mathbb{I}$ – $J^+(ins_1, ins_2, \ldots ins_k)$

```
39 Algorithm J+(I)
40     iVolume ← 0 ; uVolume ← 0   til^ce ← FindCoexistence(I)
41     foreach ivl in til^ce do
42         geometries ← []   foreach ins_i in I do
43             geometries.Insert(ins_i.GetGeometryAt(ivl))
           /* Check spatial overlap for finding co-occurrence time intervals          */
44         if Intersects(geometries) then
45             iVolume ← iVolume+ Area(Intersection(geometries)) *ivl.length  uVolume ←
                uVolume + Area(Union(geometries)) * ivl.length
46     if iVolume = 0 then
47         return 0
48     else
49         return iVolume/uVolume
```

4.5.2 J^+ Calculation Algorithm

We present the generalized J^+ calculation algorithm (for k instances) in Algorithm 4.2. The algorithm initially determines the coexistence time intervals til^{ce} (temporally overlapping time intervals) for all the instances. Then, for each interval in til^{ce}, spatial overlap is among the geometries are checked to determine the co-occurrence time intervals. If the geometries of all participating instances at a particular time interval spatially overlaps, then we calculate intersection and union volumes for that interval, and add them to the total intersection and union volumes. Lastly, the ratio between the total intersection and union volume is returned.

4.5.3 OMIN and OMAX Calculation Algorithms

We present the generalized OMIN and OMAX calculation algorithms (for k instances) in Algorithm 4.3 and Algorithm 4.4. The algorithms initially determine the coexistence time intervals til^{ce} for all the instances and find the intersection volume. Then, for OMAX, maximum volume (of participating instances) and for OMIN minimum volume

Algorithm 4.3 Generalized OMIN Calculation

Input: A collection of k spatiotemporal instances – $\mathbb{I} = \{ins_1, ins_2, \ldots ins_k\}$
Output: OMIN value for instances in $\mathbb{I}$ – $OMIN(ins_1, ins_2, \ldots ins_k)$

```
50 Algorithm OMIN(𝕀)
51     iVolume ← 0 ; minVolume ← 0   til^ce ← FindCoexistence(𝕀)
52     foreach ivl in til^ce do
53         geometries ← []   foreach ins_i in 𝕀 do
54             geometries.Insert(ins_i.GetGeometryAt(ivl))
           /* Check spatial overlap and calculate intersection volume        */
55         if Intersects(geometries) then
56             iVolume ← iVolume+ Area(Intersection(geometries)) ∗ivl.length
57     foreach ins_i in 𝕀 do
58         minVolume ← MinOf(minVolume, ins_i.GetVolume())
59     if iVolume = 0 then
60         return 0
61     else
62         return iVolume/uVolume
```

Algorithm 4.4 Generalized OMAX Calculation

Input: A collection of k spatiotemporal instances – $\mathbb{I} = \{ins_1, ins_2, \ldots ins_k\}$
Output: OMAX value for instances in $\mathbb{I}$ – $OMAX(ins_1, ins_2, \ldots ins_k)$

```
63 Algorithm OMAX(𝕀)
64     iVolume ← 0 ; maxVolume ← 0   til^ce ← FindCoexistence(𝕀)
65     foreach ivl in til^ce do
66         geometries ← []   foreach ins_i in 𝕀 do
67             geometries.Insert(ins_i.GetGeometryAt(ivl))
           /* Check spatial overlap and calculate intersection volume        */
68         if Intersects(geometries) then
69             iVolume ← iVolume+ Area(Intersection(geometries)) ∗ivl.length
70     foreach ins_i in 𝕀 do
71         maxVolume ← MaxOf(maxVolume, ins_i.GetVolume())
72     if iVolume = 0 then
73         return 0
74     else
75         return iVolume/uVolume
```

is determined. Lastly, the OMAX returns the ratio of intersection volume to maximum volume and the OMIN returns the ratio of intersection volume to minimum volume.

4.6 Experimental Evaluation of Significance Measures

In this section, we will evaluate the relevancy and efficiency of the J, J^+, J^*, OMIN, and OMAX measures. The J, OMAX, and OMIN measures are previously used for STCOP mining in [37], [121], [38].

We have conducted our experiments on for real-life solar event datasets and four artificial datasets with varying spatiotemporal characteristics. The artificial datasets are generated using the random dataset generator [132]. The solar event data is partitioned into four datasets, each corresponding to three-month periods (quarters) in 2012. For relevancy analysis, we used the solar event datasets. For efficiency analysis, we used both artificial and solar event datasets. In our experiments, we have enumerated all the size-2 and size-3 spatiotemporal co-occurrences among the instances of all different event types. We reported the J, J^+, J^*, OMIN, and OMAX values and running times for each spatiotemporal co-occurrence.

Table 4.3: Datasets used in the experiments

	Dataset	Tag	Start Date	End Date	#of Polygons	#of Instances
Solar	Quarter 1 - 2012	Q1	01/01/2012	03/31/2012	439,512	6,498
Event	Quarter 2 - 2012	Q2	04/01/2012	06/30/2012	537,078	7,911
Datasets	Quarter 3 - 2012	Q3	07/01/2012	09/30/2012	570,875	8,527
	Quarter 4 - 2012	Q4	10/01/2012	12/31/2012	503,001	6,854

Artificial Datasets

Dataset	Tag	# of Vertices per Polygon	# of tpgs per Instance	#of Polygons	#of Instances
Low Vertex Short Lifespan	LVSL	20	20	20,000	1,000
Low Vertex Long Lifespan	LVLL	20	100	100,000	1,000
High Vertex Short Lifespan	HVSL	100	20	20,000	1,000
High Vertex Long Lifespan	HVLL	100	100	100,000	1,000

4.6.1 *Experimental Settings*

Our real-life solar event datasets are obtained from Heliophysics Event Knowledgebase [123]. The individual recordings of the solar events are tracked and interpolated using the algorithms in [133] and [24]. The solar event datasets include the instances of seven solar event types that are: (1) Active Regions (AR), (2) Coronal Holes (CH), (3) Emerging Flux (EF), (4) Filaments (FI), (5) Flares (FL), (6) Sigmoids (SG), and (7) Sunspots (SS). The artificial datasets have two event types. The details of these datasets can be found in Table 6.1. The significance measures are implemented in the Java programming language. We used Algorithm 1 and Algorithm 2 for J^* calculations. The algorithms for the J, J^*, OMAX, and OMIN measures are shown in Algorithm 4.1, Algorithm 4.2, Algorithm 4.4, and Algorithm 4.3, respectively.

4.6.2 *Relevancy Analysis*

In this part of our discussion, we will discuss the relevancy of our new measures, J^+ and J^*. In Section 4.2, we have outlined the possible anomalies created due to unfair assessments of the J measure. We will analyze each of the mentioned anomalies for solar event data, compare the measures with OMAX and OMIN measures, and how the J^* measure addresses these particular problems.

In Figure4.6, we demonstrate boxplots showing the value distributions of J, J^+, J^*, OMIN, and OMAX measures for size-2 and size-3 co-occurrences in Q1 dataset. The results from other solar event datasets can be found in [134]. The co-occurrences are grouped by the event types of participating instances. In size-2 co-occurrences, the J^+ and J^* values are merged as we have observed that they were the same. Additionally, the medians and means of each distribution are represented as red lines and magenta dots.

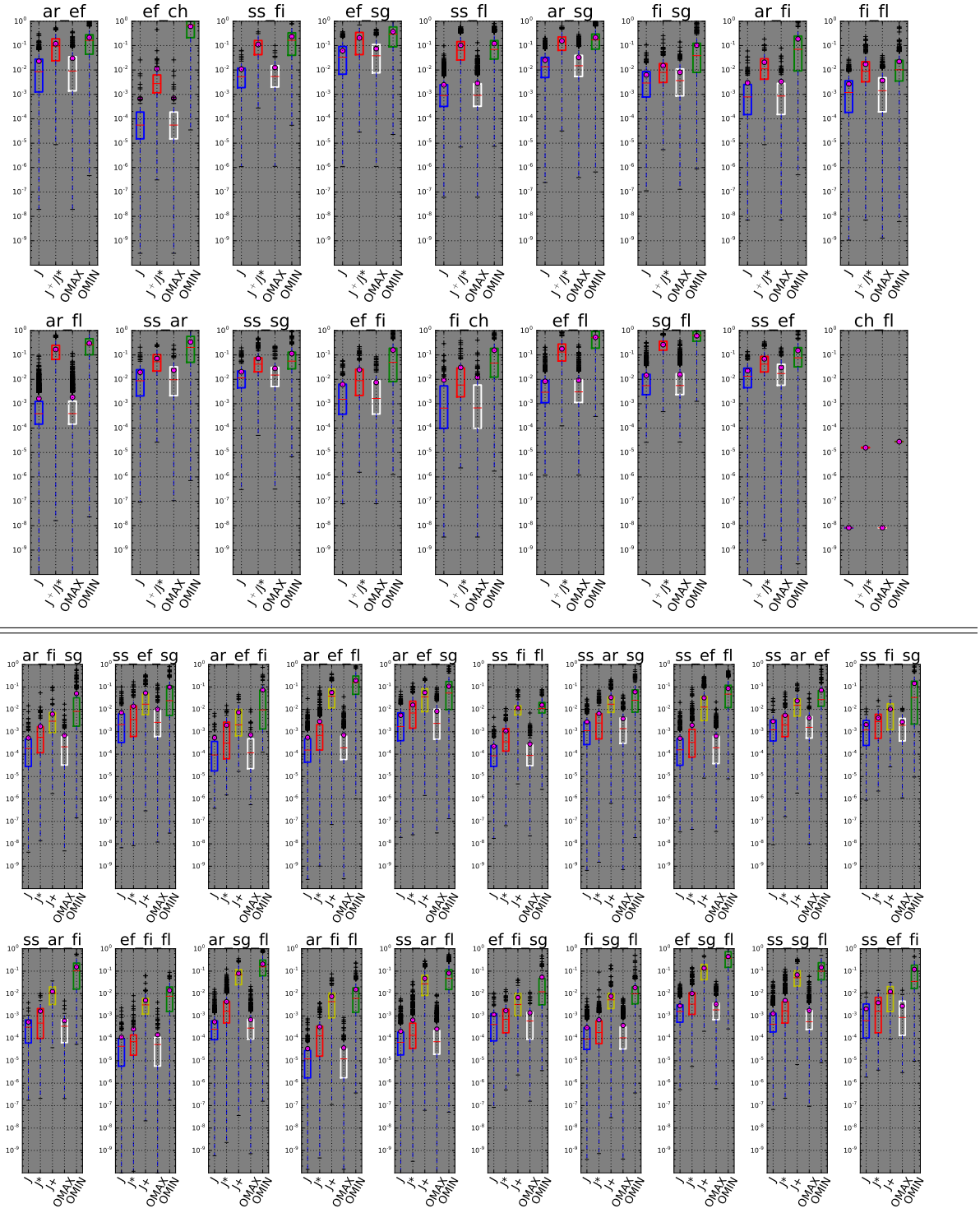


Figure 4.6: The boxplots showing the distribution of J, J^+, J^*, OMIN, and OMAX values (in log scale) for size-2 and size-3 spatiotemporal co-occurrences in the *Q1* dataset. Each sub-figure shows co-occurrences between different event types. For size-2 co-occurrences J^+ and J^* are joined as they are the same. J, J^+,J^*, OMIN, and OMAX values are represented with blue, yellow, red, white, and green boxes respectively.

From Figure 4.6, by analyzing the mean and median points and the confidence intervals of the distributions, we can see the following:

- OMAX and J measures have similar distributions.
- J^* values are generally higher than J and OMAX values, and lower than J^+ and OMIN measures.
- The OMIN and J^+ measures have the highest values.
- The confidence interval for J^+ measure is usually smaller than the others.
- When compared to the J^* or J^+ values, the J and OMAX values have higher chances of being an outlier within their own data series. This situation is more noticeable particularly for the co-occurrences that involves the event instances of Flare (FL).
- OMIN measure creates more outliers when spatiotemporal characteristics of the event types are similar.
- The greater variations among the value distributions are from the co-occurrences between the instances of event types with very different characteristics such as EF-CH, SG-FL, or AR-SG-FL.
- There are no theoretical containment relationship between our new measures (J^+ or J^*) and overlap measures (OMAX and OMIN). However, for both mean and median values, we see the following trend $J < OMAX < J^* \leqslant J^+ < OMIN$.

In Figure 4.6, we can observe that the co-occurrences of EF-SG, SS-AR, AR-SG, SS-SG, and SG-FL have relatively higher J and OMAX values. On the other hand, the co-occurrences between AR-EF, EF-SG, AR-SG, AR-FL, EF-FL, and SG-FL have higher J^* (or J^+) and OMIN values. We can suggest that the significance assessments with J^+, J^*, or OMIN measures increase the likelihood of a small volume instance (such as EF or FL) to be involved in a significant co-occurrence given a particular cce threshold value.

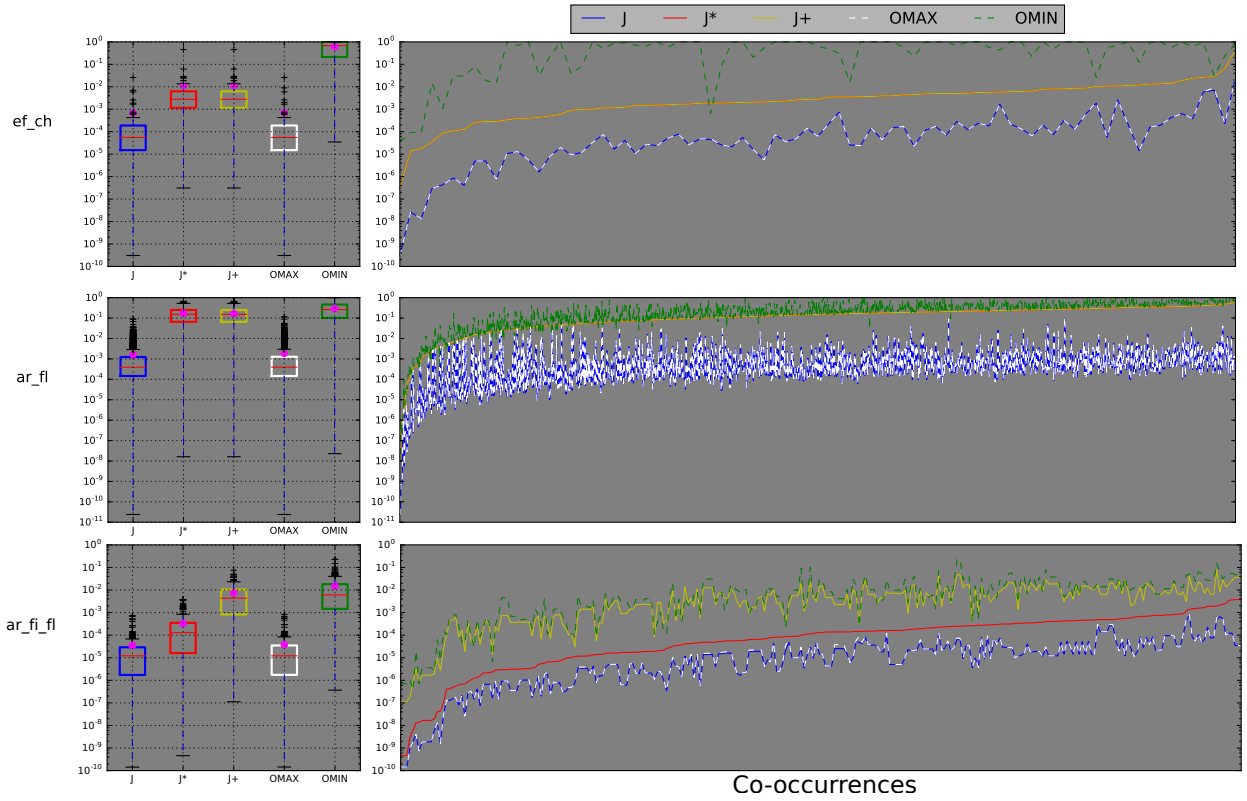


Figure 4.7: The value distributions of significance measures for EF-CH, AR-FL, and AR-FI-FL co-occurrences. The boxplots showing the distribution of the values are demonstrated on the left. On the right, the value comparison plots for individual co-occurrences are shown. The value plots are sorted on the J* measure.

Next, we will analyze the anomalies that we have mentioned in Section 4.2, and how the J* addresses these anomalies.

Coverage Anomaly

In Figure 4.3, we have demonstrated the spatiotemporal characteristics of different event types. Instances of event types such as AR and CH have relatively higher volumes. On the other hand, instances of FL or EF event types have smaller volumes. Additionally, instances of FI and SS event types have longer lifespans. The coverage anomaly refers to the unfair significance assessments of co-occurrences between very large and very small volume instances.

Figure 4.7 exhibits typical coverage anomaly problems between the large volume AR and CH instances and small volume EF and FL instances. Flares can occur anywhere on the Sun's surface, from active regions to the the boundaries of the magnetic network of the quiet Sun [135]. However, large area flares have preferred locations. They occur *inside* the large active regions showing a complex geometry of the 3D magnetic field [136]. Similarly, flux tubes (*i.e.*, emerging flux) are also observed to be emerging into coronal holes [137]. The individual OMAX and J values for above-mentioned co-occurrences in Figure 4.7 are very similar. We can suggest that small volume EF and FL instances are mostly covered, because maximum volume (denominator in OMAX) and the union volume (denominator in J) of co-occurring instances are very similar.

It is apparent that the large volumes of AR and CH instances unfairly decrease the J and OMAX values. Mean J value for AR-FL is two orders of magnitude smaller than the J^* value. Likewise, mean J values for EF-CH and AR-FI-FL are more than one order of magnitude smaller than the J^* value.

Favoring the similar

In Figure 4.8, we demonstrate three different types of size-2 co-occurrences. In Figure 4.8, the boxplots for measures and individual value comparisons for SS-FL, SS-AR and FI-CH co-occurrences are shown.

The sunspot (SS) instances appear inside the active regions (AR), and a flare (FL) is essentially an intense burst of radiation coming from the release of magnetic energy associated with sunspots [138]. Sunspots can last as long as two months, while lifespan of the flares are between mere minutes to several hours. Therefore, AR and SS instances both have relatively longer lifespans, and their volumes are larger, while FL instances have very short lifespans and small volumes. On the other hand, CH instances have large areas, longer lifespans, and large volumes, and FI instances have medium areas, but longer lifespans. To remind the readers, the J and OMAX values of co-occurrences

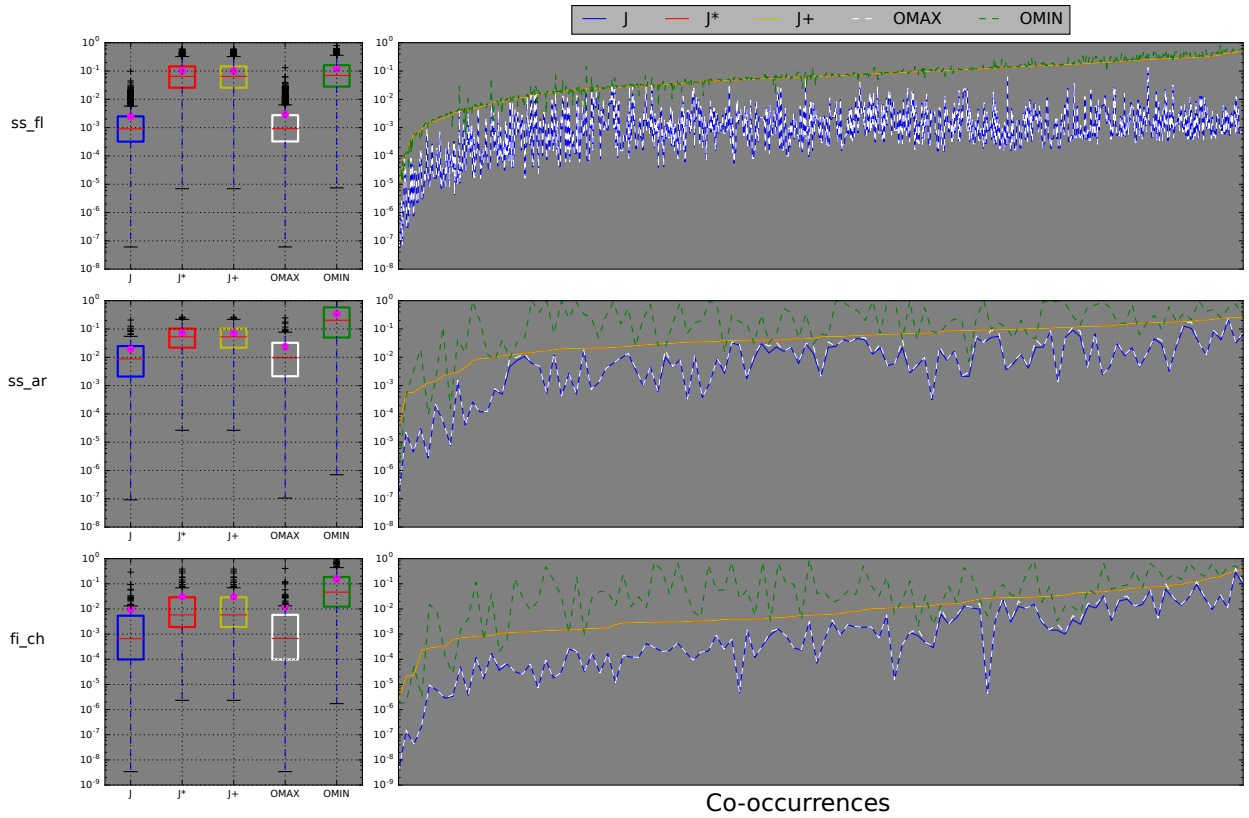


Figure 4.8: The value distributions of significance measures for SS-FL, SS-AR, and FI-CH co-occurrences. The boxplots showing the distribution of the values are demonstrated on the left. On the right, the value comparison plots for individual co-occurrences are shown. The value plots are sorted on the J^* measure.

between instances with similar spatiotemporal characteristics tend to be higher, while the co-occurrences of instances having highly different spatiotemporal characteristics are more likely to have lower J values.

From Figure 4.8, we can observe that for event types with similar spatiotemporal characteristics (in SS-AR or FI-CH) the J and OMAX values are not significantly increased with J^* or J^+. However, the contrasting spatiotemporal characteristics carried by SS and FL instances drastically affects the J values. A similar situation can be observed for AR and FL instances from Figure 4.7. The J^* significantly increases the co-occurrence coefficient values for instances with contrasting spatiotemporal characteristics such as AR-FL or SS-FL. Therefore, it can be used for alleviating the favoring the similar problem caused by the J measure.

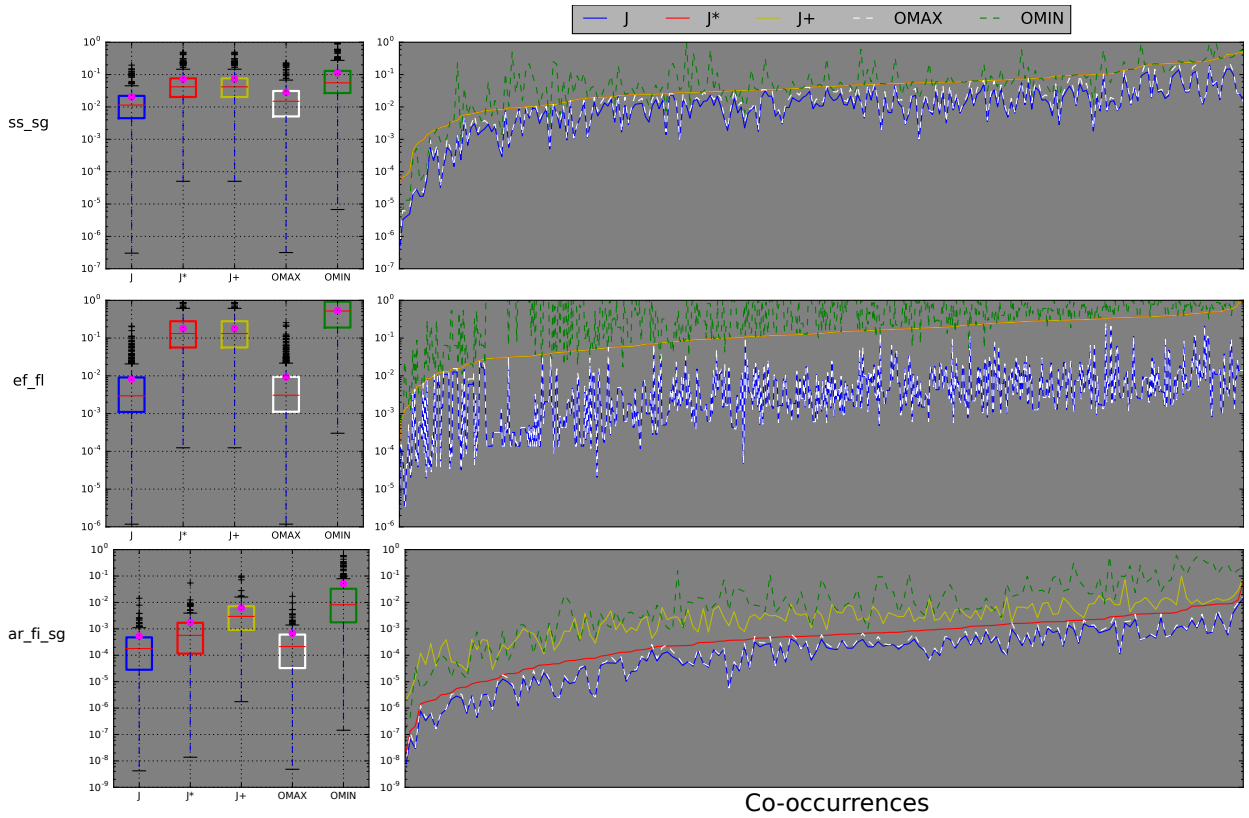


Figure 4.9: The value distributions of significance measures for SS-SG, EF-FL, and AR-FI-SG co-occurrences. The boxplots showing the distribution of the values are demonstrated on the left. On the right, the value comparison plots for individual co-occurrences are shown. The value plots are sorted on the J* measure.

Large volume bias

In Figure 4.8, we demonstrated the boxplots and value comparisons of J and J* for SS-FL, SS-AR and FI-CH co-occurrences. In Figure 4.9, we show the boxplots and value comparisons for SS-SG, EF-FL, and AR-FI-SG co-occurrences. AR, SS, and SG instances have relatively larger volumes, while FI instances have moderate, and EF and FL instances have relatively smaller volumes. SS and FI instances have long lifespans, while SG and AR have larger areas. The large volume bias is created because instances with larger volumes are more likely to have higher J values in a fixed spatial and temporal framework.

From Figure 4.8 and Figure 4.9, we can observe that the J values for co-occurrences between instances with large volumes are higher. For both SS-SG or AR-FI-SG co-occurrences, J^* does not significantly affect the co-occurrence coefficient. See the J^* measure's mean and medians for SS-SG and AR-FI-SG. Contrarily, for co-occurrences involving smaller volume instances, the J values are lower, and the J^* measure decidedly increases the co-occurrence coefficients.

Flares are our solar system's largest explosive events and they produce high energy particles and radiation that are dangerous to living organisms. Empirical studies show that flares are associated with emerging flux loops [139] [140] [141]. Using the J measure, many emerging flux and flare (EF-FL) co-occurrences can be unfairly assessed as insignificant as they have small volumes. The J^* measure can mitigate these problems, and can help find more relevant spatiotemporal co-occurrences.

4.6.3 Efficiency Analysis

In this part of our evaluation, we discuss the running time requirements of the J, J^*, J^+, OMAX, and OMIN measures for different types of co-occurrences. In Figure 4.10 and Figure 4.11, we demonstrate the boxplots of running times of the measures for individual spatiotemporal co-occurrences among the instances of different event types. For brevity, we show the results from the Q1 solar event dataset. The results from Q2, Q3, and Q4 datasets are presented in [134].

Implementation details

The J^*, J, J^+, OMAX, and OMIN calculation algorithms are presented in Algorithm 1 and Algorithm 2 (for J^*), Algorithm 4.1, Algorithm 4.2, Algorithm 4.4, and Algorithm 4.3 (for others, respectively). In our experimental runs, we stored the spatiotemporal instances in main memory. Our running time comparison solely includes the time required for procedures to calculate values of measures from given instances. For providing a fair

comparison, we implemented the measures in a similar fashion. We first find the intersection volumes, then union (J, J*, and J$^+$), minimum (OMIN), or maximum (OMAX) volumes. All the experiments are repeated for five times for both artificial and solar event datasets, and average running times are reported.

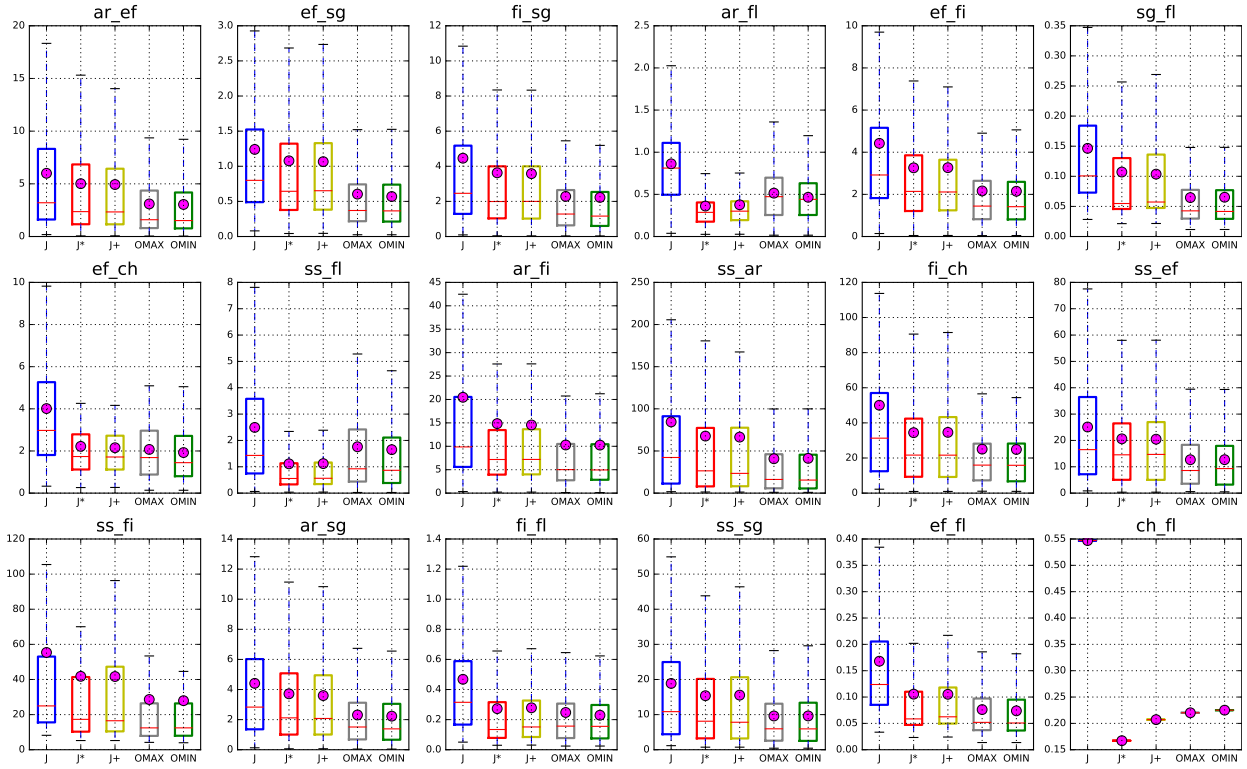


Figure 4.10: The boxplots showing the running times (in milliseconds) for J, J$^+$, J*, OMIN, and OMAX in size-2 co-occurrences in *Q1* dataset.

Comparison of Running Times for Size-2 Co-occurrences

In Figure 4.10, the boxplots of running times for size-2 co-occurrences in Q1 dataset are demonstrated. The running times are grouped by the event types of instances involved in the co-occurrences. Outliers are not demonstrated. Mean values are demonstrated with magenta dots, and median values are shown as red lines inside the boxes.

It can be observed from Figure 4.10 that for size-2 co-occurrences, the J* and J$^+$ calculations require less running time when compared to the J calculations for all different types of events. As expected, J* and J$^+$ measures, and OMAX and OMIN measures

have very similar running times. The longer running times are reported for pairs of long lifespan event instances (such as AR, SS, FI, and CH instances). Specifically, AR-SS pairs have the most computationally expensive calculations for all the measures. On the other hand, the co-occurrences involving short lifespan instances (such as EF, FL, and SG instances) require significantly less time. This situation is more noticeable when SS-AR and AR-FL co-occurrences are compared. AR-FL co-occurrences require two orders of magnitude less running times for measure calculations than SS-AR co-occurrences. This is an anticipated result because when measures are calculated the intersection volumes (and union volumes for J^* and J^+) are determined by filtering the coexistence time intervals, and the filtering operation significantly reduces the computational load for short lifespan instances.

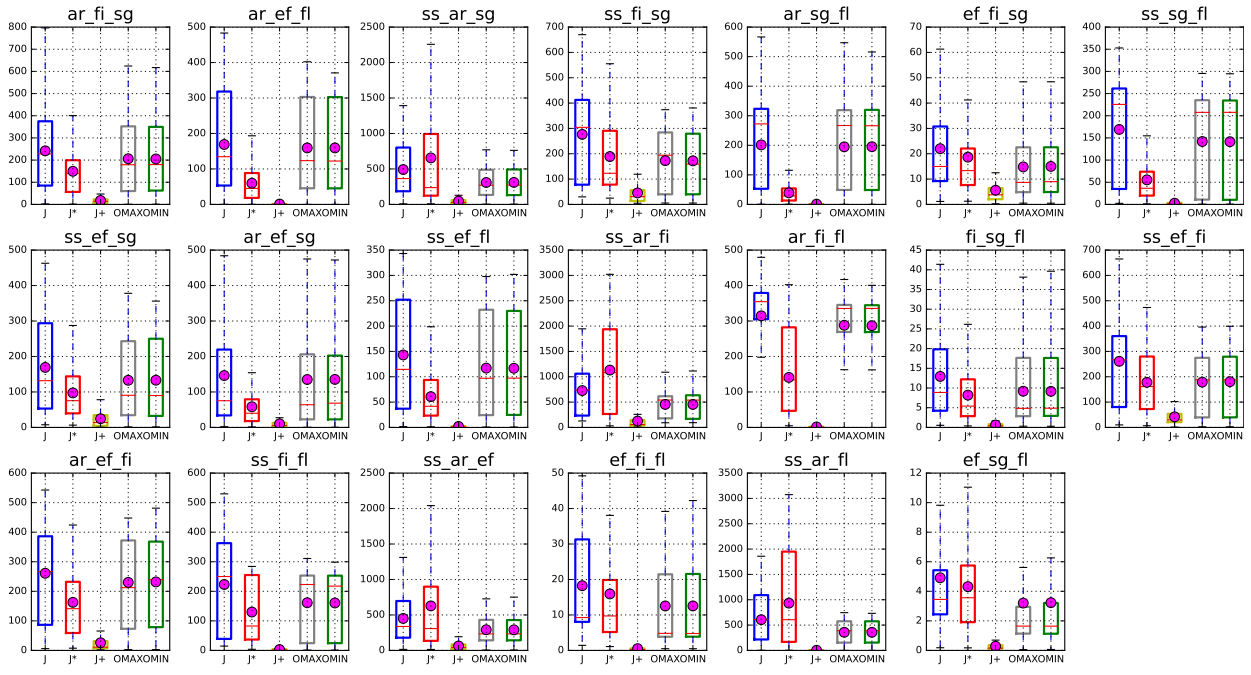


Figure 4.11: The boxplots showing the running times (in milliseconds) for J, J^+, J^*, OMIN, and OMAX in size-2 co-occurrences in *Q1* dataset.

Comparison of Running Times for Size-3 Co-occurrences

In Figure 4.11, the boxplots of running times for size-3 co-occurrences are demonstrated. Similar to Figure 4.10, the running times for the measures are grouped together based on different event type triples. Note that not all the event types co-occur with each other. For instance, we did not identify any co-occurrences of AR-CH-SG or CH-EF-SS instances in *Q1* dataset.

From Figure 4.11, we can observe that average running time (mean) for OMAX and OMIN values are very close to each other. Among all five measures, the J^+ is the most efficient one. For many size-3 co-occurrences, the J^* is more efficient than the J measure. The J is more efficient than J^* for SS-AR-SG, SS-AR-FI, SS-AR-EF, and SS-AR-FL co-occurrences. This can be explained by long co-occurrence time intervals between SS and AR instances. For many of size-3 co-occurrences that include SS and AR instances, the cross co-occurrence time intervals are very long (as sunspots occur inside the active regions), which creates an overhead for J^* calculations.

Similar to size-2 co-occurrences, the shortest calculation times are observed for co-occurrences among short lifespan instances such as EF, SG, or FL. See the boxplots for EF-SG-FL, FI-SG-FL, and EF-FI-FL in Figure 4.11. The measure calculation times are significantly higher for co-occurrences between long lifespan instances. Particularly for SS-AR-FI co-occurrences, the average J^* calculation time is greater than 1000 ms, while the mean J^* calculation time for EF-SG-FL is 4 ms.

Comparison of Running Times for Artificial Datasets

In Figure 4.12, the boxplots of running times for artificial datasets (*HVLL, HVSL, LVLL,* and *LVSL*) are demonstrated. All the artificial datasets include two artificial event types, each having 1,000 instances. The area values for the region polygons in the artificial datasets are the same. However, for observing the effect of spatial operations (*i.e.*, union, intersection, intersects, and area), we change the number of vertices to create more com-

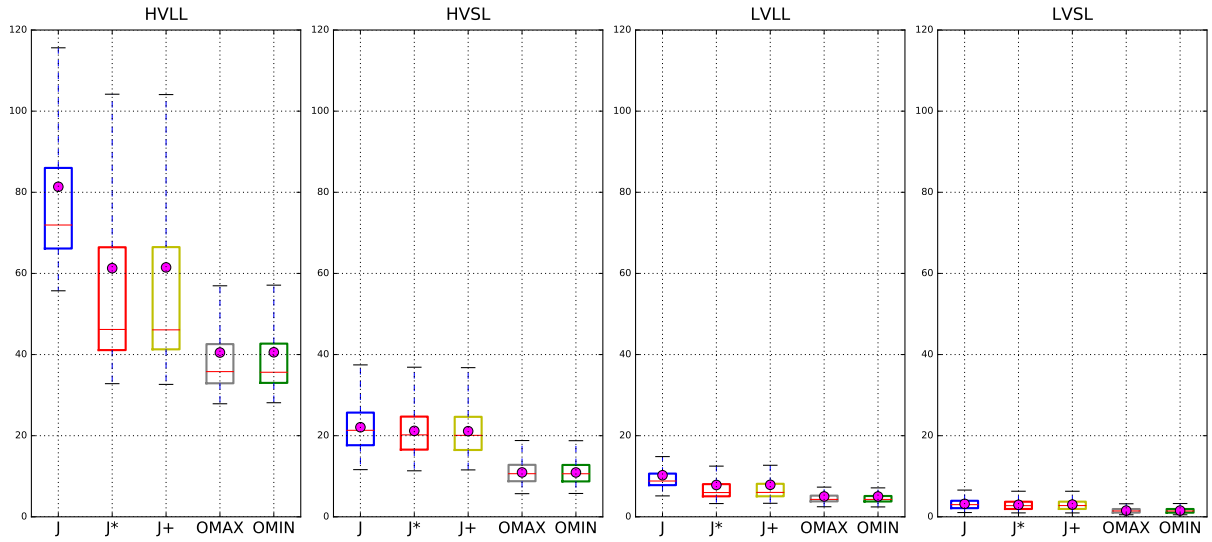


Figure 4.12: The boxplots showing the running times for J, J$^+$, J*, OMIN, and OMAX in artificial datasets.

plex region geometries. *LV_* and *HV_* prefixes denote low and high vertex counts in the region geometries represented as polygons. On the other hand, we altered the lifespans of the instances to observe the impact of temporal complexity. *_SL* and *_LL* suffixes denote short and long lifespans of the instances.

From Figure 4.12, we can make following observations. In short lifespan datasets (LVSL and HVSL), the running times of J is closer to J*'s. However, for long lifespan datasets (LVLL and HVLL), J* takes noticeably less time than J. Additionally, OMIN and OMAX are the most efficient measures for artificial datasets. Lastly, as expected, spatially more complex HVSL (with short lifespans) dataset requires more running time for all the measures when compared to temporally complex dataset LVLL (with low vertex counts).

Remarks on Running Times

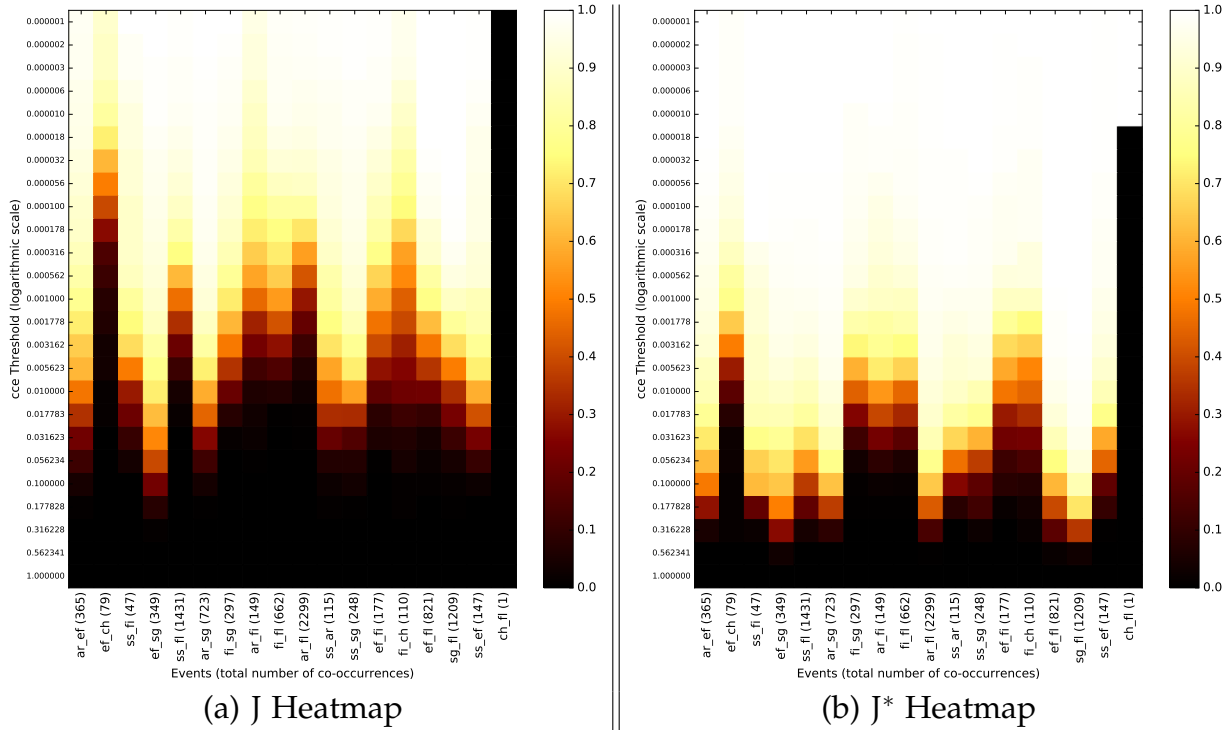
From the running time analysis of size-2 and size-3 co-occurrences from solar event datasets (Figure 4.10 and Figure 4.11) and artificial datasets (Figure 4.12), we can outline our remarks as follows:

- The OMIN and OMAX are the most computationally efficient measures for size-2 co-occurrences.
- The J^+ is the most efficient measure for size-3 co-occurrences.
- For size-2 co-occurrences, the J^* consistently takes less time than the J.
- For size-3 co-occurrences, the J^* usually takes less time than the J. In four types of co-occurrences involving long lifespan SS and AR instances, the J calculation takes less time on average.
- Increasing spatial complexity (more complex polygons) impacts the running time more than increasing the temporal complexity (longer lifespans).

The main difference between the J, J^+ and J^* calculation procedures is the identification of cross co-occurrence or co-occurrence time intervals. The J^* calculation procedure for size-2 co-occurrences has a shortcut for direct identification of cross co-occurrence time intervals. For larger size co-occurrences, the cross co-occurrence time intervals are found by examining the co-occurrence time intervals of each instance pair. This operation is expensive as it essentially checks whether a spatial overlap occurs between each geometry of each instance pair. On the other hand, the J^+ calculation is relatively less complex. The co-occurrence time intervals are found using a temporal coexistence filter, which reduces the search space. While determining the co-occurrence or cross co-occurrence time intervals creates overhead, the J^+ and J^* measures calculate the union volume for only co-occurrence and cross co-occurrence time intervals, and remaining time intervals are not considered. Contrarily, the J measure includes the union volumes of all instances.

In summary, for larger size co-occurrences, J^* calculation can be less efficient as it needs to initially identify the cross co-occurrence time intervals. For spatiotemporal frequent pattern mining applications (such as STCOP mining [142]), initial identification

of cross co-occurrence time intervals is not usually performed, as the larger size co-occurrences are found by using smaller size co-occurrences, and the cross co-occurrence time intervals can be easily identified. On the other hand, the J^* can be more efficient for co-occurrences with certain characteristics (such as long lifespan instance vs. short lifespan instance - see running times of FI-AR-EF in Figure 4.11), as it only calculates the union volumes at cross co-occurrence time intervals.

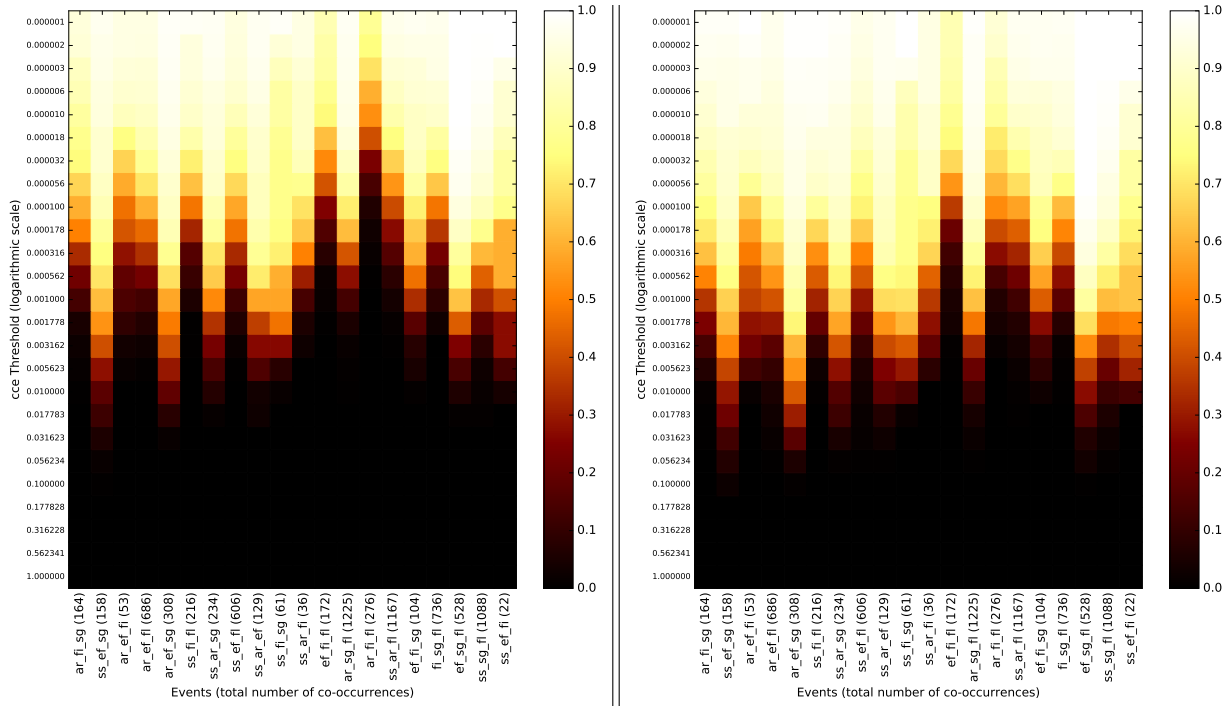


(a) J Heatmap (b) J* Heatmap

Figure 4.13: Heatmaps of the J (in (a)) and J^* (in (b)) values of size-2 co-occurrences for the Q1 dataset. The heatmaps demonstrate the ratio of the significant size-2 co-occurrences for different co-occurrence coefficient (cce) thresholds and event types of co-occurring instances, on x and y axes respectively.

4.6.4 Suitability for STCOP Mining

In this part of the experiments, we will discuss the measures from the perspective of STCOP mining. It should be noted that STCOP mining algorithms require antimonotonic measures for the correctness. Therefore, OMIN and J^+ measures, which does not carry antimonotonic property cannot be used in the context of current STCOP mining



(a) J Heatmap (b) J* Heatmap

Figure 4.14: Heatmaps of the J (in (a)) and J* (in (b)) values size-3 co-occurrences for the Q1 dataset. The heatmaps demonstrate the ratio of the significant size-3 co-occurrences for different co-occurrence coefficient (*cce*) thresholds and event types of co-occurring instances, on x and y axes respectively.

algorithms. We also showed that OMAX and J values are very similar in Section 4.6.2. Therefore, here, we will compare the J measure, which is currently used for ultimately determining the significance of co-occurrences, with our newly proposed J* measure.

In Figure 4.13 and Figure 4.14, the heatmaps of J and J* values for size-2 and size-3 co-occurrences are demonstrated. Essentially, all the heatmaps encode information regarding the ratio of significant co-occurrences to all discovered co-occurrences (of particular event types) for specific co-occurrence coefficient (*cce*) threshold values. In Figure 4.13 and Figure 4.14, the threshold values exponentially decrease from 1.0 to 0.000001 (10^{-6}) by $\sqrt[4]{10}$.

In STCOP mining, a key challenge is to determine a meaningful co-occurrence coefficient threshold for the recognition of strong co-occurrences [38]. The co-occurrence coefficient threshold essentially implies the level of significance for the STCOP mining

scheme. In other words, for a particular *cce* threshold, the STCOP mining algorithm identifies the patterns whose instances have a strong co-occurrence based on that *cce* threshold. Apart from the anomalies of J measure in Section 4.6.2, for the spatiotemporal instances with highly unbalanced characteristics (e.g. solar event datasets), it is difficult to determine a threshold or a set of thresholds using the J measure for STCOP mining analysis. The reason for that is the fluctuation of J values happens in a very limited interval. We can observe from Figure 4.13 that for the J values, the variability on the ratio of significant co-occurrences can only be observed in threshold interval from 0.1 to 0.0001. Conversely, much of the variability for J^* can be observed in threshold interval from ~0.316 to 0.001.

Another problem with using the J is the exclusion of important co-occurrences or inclusion of spurious co-occurrences. We have mentioned the strong association between flares and active regions in Relevancy Analysis (Section 4.6.2). For including a very optimistic 30% of AR-FL co-occurrences in Q1 dataset, the *cce* threshold should be set to less than 0.000562. Such a mining schema would consider almost all the identified co-occurrences as significant, which makes the significance assessment procedures pointless because it would include all the co-occurrences including possibly spurious ones. Conversely, using the J^* measure in STCOP mining, with a *cce* threshold set to 0.1 or 0.0562, would include 60% to 80% of AR-FL co-occurrences, while preserving variations among the other co-occurrences.

4.7 Summary on Significance Measurements

We presented two novel significance measures, J^+ and J^*, which are specifically designed for determining the strength of spatiotemporal co-occurrences appearing among event instances. We have initially presented shortcomings of the currently used OMAX and J measures with example anomaly scenarios that can appear among the spatiotemporal co-occurrences. These anomalies can lead to unfair significance assessments that can

impact the applicability of data mining algorithms to real life datasets. As a solution to these anomalies, we introduced the J^+ and J^* measures. Both J^+ and J^* are extensions to the J measure. Our measures limit the volume calculations to specific regions of interests that are co-occurrence (for J^+) and cross co-occurrence (for J^*) time intervals. We have also presented novel algorithms for J^+ and J^* value calculations, which uses temporal coexistence filtering. We have provided proofs for antimonotonicity and containment properties of J^*. For demonstrating the effects of using our new measures for assessing the strength of co-o ccurrences, we have conducted our experiments with four solar event datasets and four artificial datasets. In our experiments, we have compared our measures with J, OMIN, and OMAX measures, and shown that J^* and J^+ can solve the anomalies created by the J and OMAX measures. As a result of our experiments, we have confirmed that J^* measure is more efficient than J measures, and can be utilized for discovering more meaningful spatiotemporal co-occurrences.

5 SPATIOTEMPORAL EVENT SEQUENCE MINING

Spatiotemporal event sequences (STESs) are the ordered sequences of event types, which frequently follow each other in spatiotemporal context. Formally, given a dataset of event types ($\mathbb{E} = \{e_1, \ldots, e_m\}$) and spatiotemporal event instances ($\mathbb{I} = \{ins_1, \ldots ins_n\}$, where each ins_i is a spatiotemporal event instance defined by an evolving region trajectory and is associated with an event type (e_j)), the purpose of STES mining is to discover sequences of event types in the form $(e_{j_1} \rhd e_{j_2} \rhd \ldots \rhd e_{j_k})$ such that the instances of participating event types *temporally follow* each other and *spatially located close-by* at certain locations where sequence forming behavior is observed. These two conditions define the spatiotemporal follow relationship.

A spatiotemporal follow relationship occurs between two event instances. Two event instances, ins_i and ins_j, which have a follow relationship (denoted as $\blacktriangleright$ for instances) between each other, form the simplest form of instance sequence, that is a length-2 se-

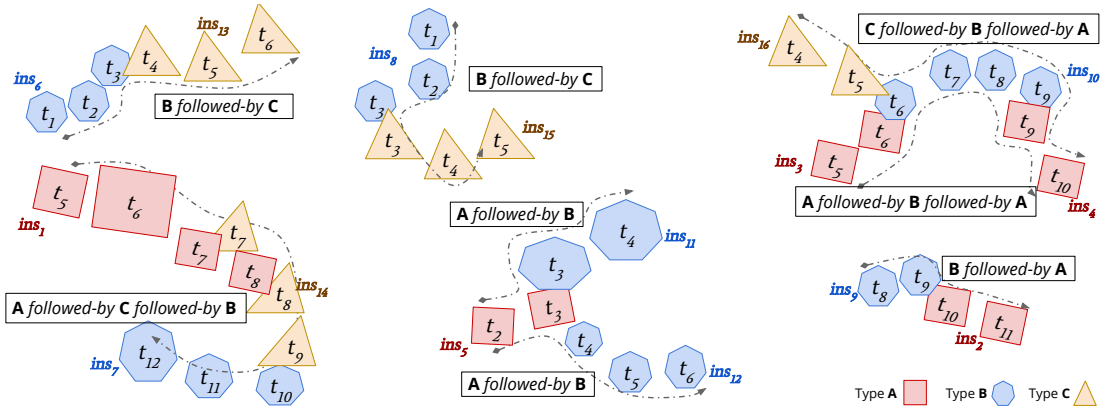


Figure 5.1: An example dataset of spatiotemporal instances $\mathbb{I}$ with 3 event types A, B, and C. The spatiotemporal instances are evolving region trajectories. The timestamps are displayed on the geometries. The dataset includes five instances of event type A ($ins_1, \ldots ins_5$), seven instances of event type B ($ins_6, \ldots ins_{12}$), and four instances of event type C ($ins_{13}, \ldots ins_{16}$). The figure also illustrates spatiotemporal follow relationships between the instances.

quence (*i.e.*, $ins_i \blacktriangleright ins_j$). Multiple follow relationships observed in consecutive instances form longer instance sequences. For example, if there is a follow relationship between ins_i and ins_j, and another one between ins_j and ins_k, they form a length-3 sequence, $(ins_i \blacktriangleright ins_j \blacktriangleright ins_k)$.

To illustrate the problem better, in Figure 5.1, we depict an example dataset of sixteen instances ($\mathbb{E} = \{ins_1, \ldots ins_{16}\}$) from three different event types ($\mathbb{E} = \{A, B, C\}$). The times are marked on the region polygons of the instances, and their shapes are different for each instance. We indicate the spatiotemporal follow relationships among the instances with dashed arrows. For example, there are two instances of event type B, which are followed by an instance of event type C (forming $(B \rhd C)$, see ins_8 is followed-by ins_{15}). It is possible to see the longer length sequences, as well as the ones with repetitions. An example for longer length sequences with repetitions is A followed-by B followed-by A (forming $(A \rhd B \rhd A)$, see ins_3, ins_{10}, and ins_4). Similarly, the same instance can be followed by more than two seperate instances. For example ins_5 is followed by ins_{11} and ins_{12}, and there are no sequence forming relationship between ins_{11} and ins_{12}.

The goal of the spatiotemporal event sequence mining is to find frequently occurring spatiotemporal follow relationships among the instances of different event types and create event sequence patterns from these individual relationships. In our example dataset shown in Figure 5.1, we observe two $(B \rhd C)$ sequences, and three $(A \rhd B)$ sequences. However, we do not see any $(C \rhd A)$ sequences. While we count the number of relationships to highlight our point in this example, we use a relative frequency based measure (prevalence index) to measure the frequency of the spatiotemporal event sequences.

In this chapter, we will focus on modeling the spatiotemporal event sequences, and the algorithms for mining the spatiotemporal event sequences. We will explain the model we developed for the follow relationship in Section 5.1 and Section 5.2, as well as the preliminary concepts of mining. In Section 5.3, we will present two Apriori-based STES mining algorithms. In Section 5.4, we will present our pattern growth-based algorithms.

Lastly, in Section 5.6, we will present a new technique for mining spatiotemporal event sequences without thresholds.

5.1 Modeling Spatiotemporal Event Sequences

Spatiotemporal event *instance*s are evolving region trajectories with a unique identifier and an associated event type. Our evolving region trajectory model is described thoroughly in Chapter 3. To remind the readers, the spatiotemporal event instances are formed by evolving region trajectories. In our trajectory data model, we use a chronologically ordered list of time-geometry pairs for representing moving region objects (instances) that create the trajectories. Each time-geometry pair represents the location of the instance at a particular time (point or interval). Event instances are identified by a unique identifier. Moreover, each event instance is associated with an event type. The event types signifies the class of its associated instances. The event type of an instance is represented with $ins_i.\mathcal{E}$.

We denote the set of instances as $\mathbb{I} = \{ins_1, \ldots, ins_n\}$. An event type is denoted by e_j. The set of all event types is denoted as $\mathbb{E} = \{e_1, e_2, \ldots, e_m\}$. We expect m to be much smaller than n ($m << n$). The set of instances of type e_j is represented as $\mathbb{I}_{e_j}$. In other words, the set of all instances is formed by the union of the event instances of event types in $\mathbb{E}$ ($\mathbb{I} = \bigcup_{e_j \in \mathbb{E}} \mathbb{I}_{e_j}$).

A spatiotemporal event sequence (denoted as ES) is an ordered series of event types with possible repetitions.

$$ES_i = (e_{i_1} \rhd e_{i_2} \rhd \ldots \rhd e_{i_k}) \tag{5.1}$$

The *follow relationship* between two event types is denoted by the '$\rhd$' symbol. This is to say, '$e_i \rhd e_j$' indicates e_i is followed-by e_j. Event sequences are derived from instance sequences. An instance sequence (denoted as ISq) is a unique occurrence of a spatiotem-

poral event sequence. Instance sequences are formed by individual instances, which follow each other in spatiotemporal context.

$$ISq_i = (ins_{i_1} \blacktriangleright ins_{i_2} \blacktriangleright \ldots \blacktriangleright ins_{i_k}) \tag{5.2}$$

The number of participating instances in an instance sequence is the length of the instance sequence. To refer to the length-k instance sequences, we will use the term k-sequence. Given an event sequence ES_i, an instance sequence (ISq_i) is *of-type* ES_i, if and only if the event types of the participating instances of ISq are identical and in the same order as the event types in ES_i. This is to say (following the notation in Eq. 5.1 and Eq. 5.2), if ins_i is *of-type* ES_i, then $ins_{i_1}.\mathcal{E} = e_{i_1}, ins_{i_2}.\mathcal{E} = e_{i_2}, \ldots$, and $ins_{i_k}.\mathcal{E} = e_{i_k}$.

5.1.1 Head and Tail Window of an Instance

The instance sequences are formed by two or more instances. Between each two consecutive instances there exists a spatiotemporal *follow* relationship. Essentially, the *follow* relationship occurs between two event instances, and is denoted with the '$\blacktriangleright$' symbol. The relationship is characterized by two predicates that delineate temporal continuity and spatial proximity.

To actualize these predicates, we present two concepts that are the *head* and the *tail window* of instances. The head of an instance refers to the initial segment of the instance's evolving region trajectory. Similarly, the *tail* of an instance refers to the last segment of the trajectory. Tail window is a complex spatiotemporal buffer obtained by spatially buffering and temporally propagating the tail of an instance. Given an instance, ins_i, the head and tail window of ins_i are represented with h_i and tw_i, respectively.

5.1.2 *Generating Head and Tail Window*

An example of head and tail generation from an instance can be seen in Fig 5.2. In our example, we used the interval-based head and tail generation, where the head interval is 2 *days*, and the tail interval is 3 *days*. The initial 2-day segment of the instance, which corresponds to the first two time-geometry pairs of the trajectory, is the head of the instance. Similarly, the final 3-day segment of the instance is the tail of the instance. An example of tail window generation can be seen in the lower-right section of Fig 5.2.

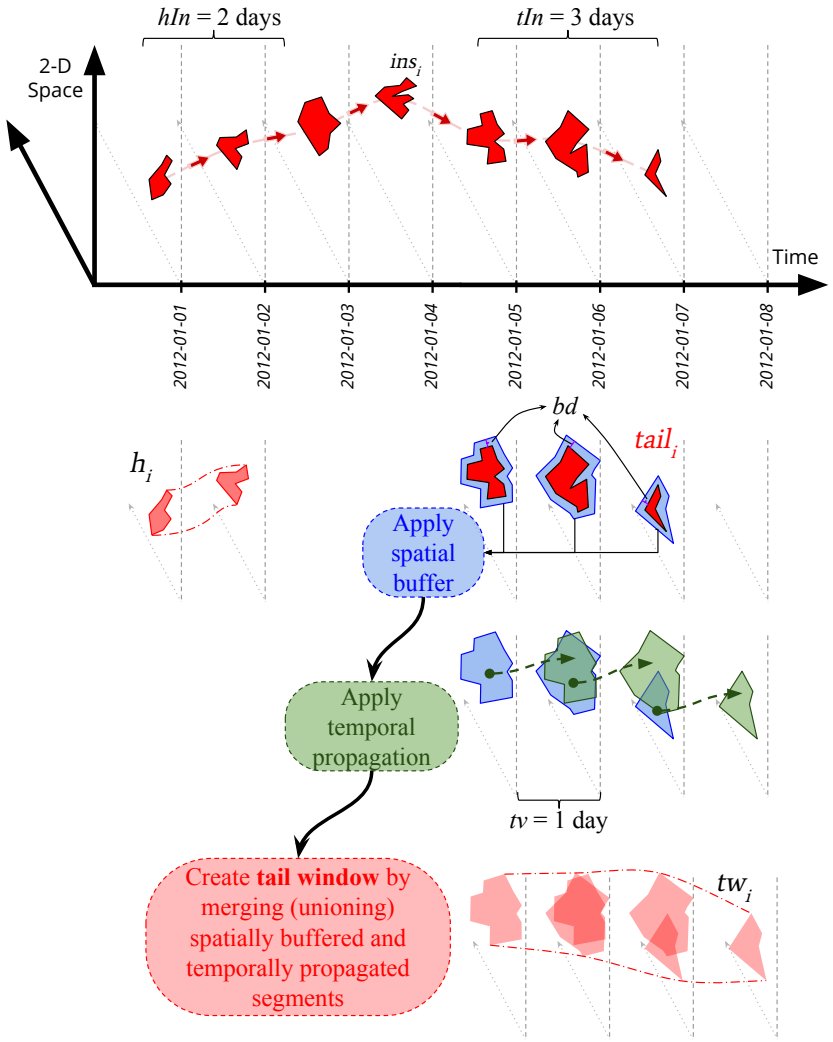


Figure 5.2: Creating the head and tail window of an instance. (Parameters: $\mathtt{hIn} = 2\mathtt{days}$, $\mathtt{tIn} = 3\mathtt{days}$, and $\mathtt{tv} = 1\mathtt{day}$)

Firstly, the tail of the instance in Fig 5.2 is spatially buffered. Then, each geometry in the buffered tail is considered to last its effect for another day; thus, they are propagated in time for one day.

The tail window is a unidirectional temporal projection of buffered tail geometries. It is designated to represent the propagating temporal effect of individual tail geometries. The buffer distance, used when creating the tail buffer, determines the amount of spatial span of the instance at a particular time interval. Tail validity can be seen as the amount of time that the spatial span continues its effectiveness. A fine analogy would be the effect of burglaries at a certain area. If an unexpectedly high number of household burglaries happen at a particular apartment complex (tail), it is expected to lower the rents in that particular complex, as well as the neighboring housing options (buffered tail). We would expect to see the low rent trend caused by the burglaries for a particular amount of time, usually until people are persuaded that the area is secure, or the burglaries are forgotten (tail validity).

When creating the tail window of an instance, we initially get the tail segment of the instance and buffer the geometries in the tail. The buffer operation is a *spatial-only* buffer, where the indivudual geometries are expanded only in two-dimensional space but not in the time dimension. A *spatiotemporal buffer* operation applied to the tail would bidirectionally expand the boundaries of the tail in both spatial and temporal dimensions. Tail window, on the other hand, is the aggregation of the unidirectional temporal projection of the buffered geometries of the tail. It is important to note that buffered geometries in the tail are projected to succeeding timestamps, but the preceding geometries are neither buffered in space nor projected in time.

For clarification, in Fig. 5.2, we illustrated the creation of head and tail window of an instance. In our example, the head interval is $2days$, the tail interval is $3days$, and the tail validity is $1day$. Given the instance, ins_i, in Figure 5.2,

- The head of the instance is the trajectory segment composed of the two initial time-geometry pairs of the instance.
- The tail of the instance is the trajectory segment composed of the final three time-geometry pairs of the instance.
- Buffered tail geometries are only determined by spatially buffering the geometries in the tail.
- The geometries in the tail window are determined by spatially unioning the corresponding buffered tail geometry and $tv = 1day$ previous geometries. The tail window geometry at $t = 2012-01-06$ is found by unioning the buffered geometries from $2012-01-05$ and $2012-01-06$.

5.1.3 Strategies for Head and Tail Window Generation

With the parameterized approach on creating the heads, tails, and tail windows of instances, we aim to create a flexible framework for mining the event sequences. These concepts can be interpreted as the regions of interest for their respective domains. In this part of our discussion, we will present different strategies for generating heads and tails of the instances.

Selection of the Segment: Interval-based vs. Ratio-based Generation

In the interval-based generation strategy, we consider two global parameters to be applied to the instances to generate trajectory segments. These are the head interval (hIn) and the tail interval (tIn) parameters. The head interval refers to the time period for determining the head segment of the instance's trajectory. Similarly, the tail interval is used to determine the tail segment of the trajectory. The length of these intervals are fixed for all the instances in a given dataset. This is to say, all the head segments have

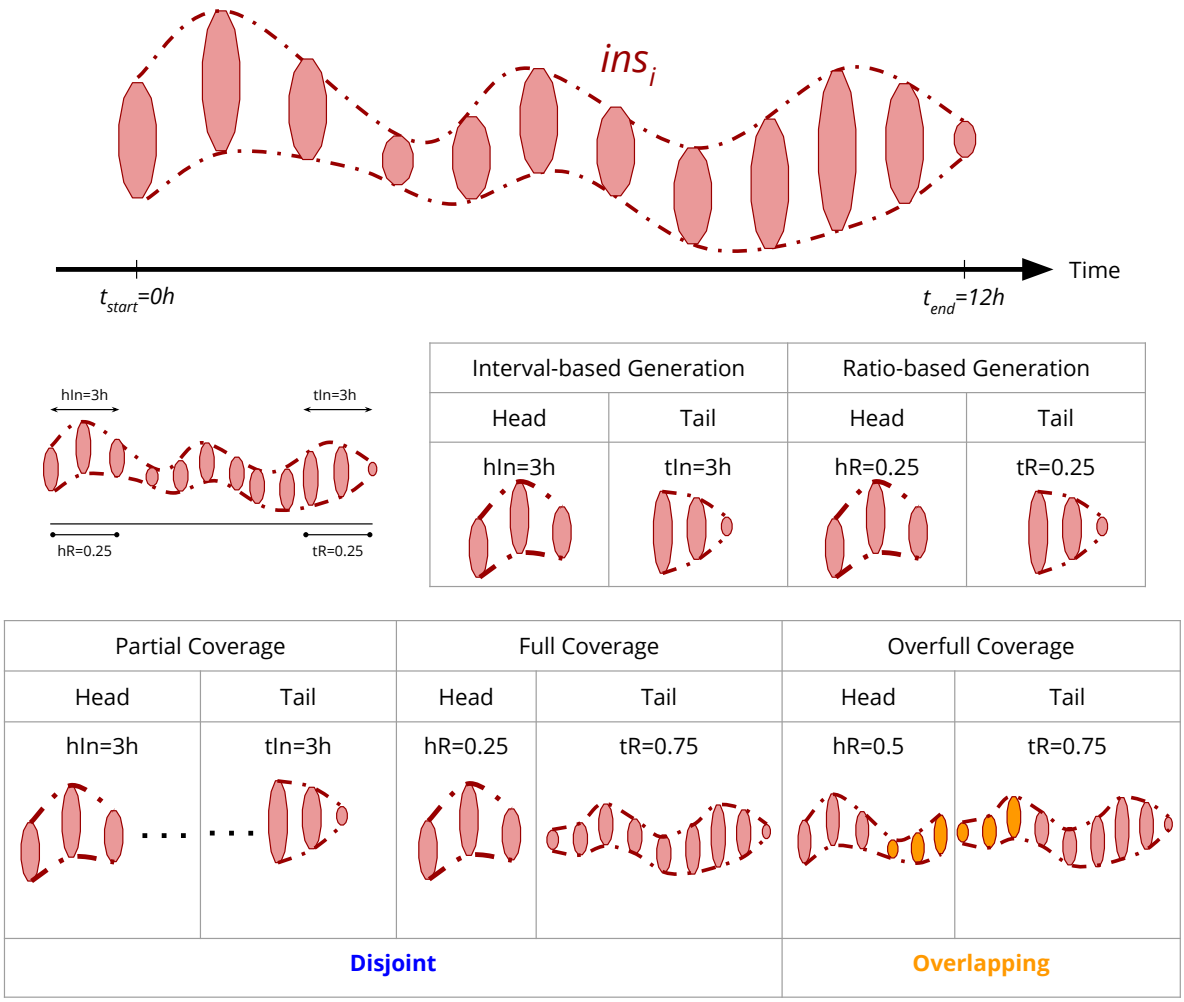


Figure 5.3: Strategies for generating head and tail of an instance

the same interval length (which is hIn), and all the tail segments have the same interval length (which is tIn).

In the ratio-based generation strategy, we are given two ratio-based global parameters that are the head ratio (hR) and the tail ratio (tR). The ratios (hR and tR) imply the proportion of trajectory's lifespan that will be assigned for head and tail segments, respectively. Note that both head and tail ratio is a number between 0 and 1 (0 is excluded, while 1 is not excluded). In this strategy, the lengths of the head and tail segments are variable, and are dependent on the lifespan of the instances.

In the interval-based strategy, the lifespan of the instances do not affect the length of the head and tail segments. Therefore, their lengths are fixed throughout the datasets.

When a given interval (head or tail interval) is greater than the lifespan of the instances, the whole trajectory is considered as either tail or head, and they are not extended. This can be problematic for consistency of the generated heads and tails. In the case of ratio-based strategy, head and tails are determined based on a ratio-based parameter (that is in the range $(0, 1]$), and the problems stemming from fixed intervals do not exist.

Coverage Strategies: Partial, Full and Overfull

An important issue with the head and tail generation is the coverage of instance trajectories. In the full-coverage strategy the entire trajectory is divided into two parts, where the initial segment is considered as the head, and the last segment is considered as the tail. The full-coverage strategy puts a constraint on the instance trajectory by using it entirely to generate the head and tail segments. Part of the trajectory is used as the head segment and the complimentary part is used as the tail segment. To actualize the full-coverage strategy, the ratio-based strategy is needed, where the sum of head and tail ratio must be 1 ($hR + tR = 1$). It can also be speculated that it is possible to use interval-based strategy for full-coverage; however, it requires all the instances in a dataset to have the same lifespan, which is generally unrealistic.

In contrast to full-coverage, with the partial-coverage strategy there can be portions of the instance trajectory not covered by either the head or tail segments. Overfull-coverage occurs when portions of the instance trajectory are covered by both head and tail segments. The partial and overfull coverage strategies are less constrained when compared to full coverage, and can be actualized by both interval and ratio-based strategies. However, to guarantee the coverage schema (for all partial, full, or overfull strategies), the ratio-based schema should be used. For the case of partial coverage $hR + tR$ must be less than 1, while for overfull coverage $hR + tR$ must be greater than 1. Using interval-based schema can create mixed strategies, where some instances might have partial coverage, while the others may have full or overfull coverage.

Overlapping vs. Disjoint Coverage Strategies

Another aspect of the head and tail generation that is worth considering is the characteristics of coverage strategies. The coverage of the instance trajectories is a primary factor in generating the sequence forming behavior, both from the relevance and computational cost perspective. We present two strategies: overlapping strategy and disjoint coverage strategy.

In the disjoint coverage strategy, no segment of the instance trajectory can a part of both head and tail segments. Partial and full coverage strategies create disjoint head and tail segments. In the overlapping coverage strategy, a portion of the instance's trajectory can be included both in the head and tail segments. Overfull-coverage leads to the overlapping strategy. An overlapping strategy guarantees the usage of all the time-geometry pairs of all the instances in the mining process, with some portions of the trajectories are used for both head and tails. In disjoint coverage strategy, portions of the time-geometry pairs may be ignored by the algorithms.

In a particular dataset, overlapping (or disjoint) head and tail segments can be guaranteed by the ratio-based head and tail generation strategy. On the other hand, usage of interval-based generation can lead to a mixed coverage strategy, where head and tail segments can be overlapping or disjoint depending on the lifespan of the instance.

It is also worth noting that using overfull strategy can drastically increase the runtime complexity of the mining algorithms, while using very-low head and tail generation parameters (*i.e.*, $\mathtt{hIn}$ and $\mathtt{tIn}$ or $\mathtt{hR}$ and $\mathtt{tR}$) can decrease the relevancy of the results. Therefore, these two aspects can be traded off to create a mining schema that is more efficient or more relevant.

Temporal Propagation Strategies for Tail Window Generation

Another issue that is worth considering is the determination of the tail validity interval. We propose two alternatives for selecting the interval for temporal propagation. The first

alternative is the fixed interval-based temporal propagation, where the tail is temporally propagated for a fixed time range. Secondly, similar to the ratio-based parameters, the tail validity interval can be determined based on a ratio-based parameter. The ratio-based tail validity interval is dependent on the lifespan of the individual instances.

5.2 Spatiotemporal Follow Relationship and Measuring the Significance

Given two instances ins_i and ins_j, there exists a spatiotemporal *follow* relationship between ins_i and ins_j ($ins_i \blacktriangleright ins_j$) if and only if (1) the start time of ins_i is less than the start time of ins_j, and (2) there exists a spatiotemporal co-occurrence between the tail window of ins_i and the head of ins_j. Under these conditions, ins_i is the *followee* and ins_j is the *follower* in the relationship.

To form a 2-sequence, there must be one spatiotemporal *follow* relationship between two instances. More generally, to form a k-sequence, there must be k-1 spatiotemporal *follow* relationships between each consecutive participating instance. That is, for k instances (ins_1, ins_2, ..., ins_k), the instance sequence $ISq = (ins_1 \blacktriangleright ins_2 \blacktriangleright \ldots \blacktriangleright ins_k)$ exists if and only if there exists a series of *follow* relationships between ins_1 and ins_2 ($ins_1 \blacktriangleright ins_2$), ins_2 and ins_3 ($ins_2 \blacktriangleright ins_3$), ..., and, lastly ins_{k-1} and ins_k ($ins_{k-1} \blacktriangleright ins_k$).

5.2.1 Significance of the Instance Sequences

An important aspect of the spatiotemporal event sequence mining is the determination of significant or spurious instance sequences. The significance assessment is important as the accuracy and reliability of the resulting event sequences are dependent on the discovered instance sequences. For assessing the significance of the *follow* relationship between instances, we present the *chain index* measure.

The chain index, denoted as ci, for 2-sequences is defined as the significance of the spatiotemporal co-occurrence between the tail window of the followee instance and the

head of the follower instance. The significance of spatiotemporal co-occurrences occurring between evolving region trajectories are studied in [37, 142]. The significance can be measured with measures such as OMAX, J, or J^*. For this work, we will use the J^* measure [142]. J^* measure between two trajectory segments is defined as the ratio of intersection to union volume at time intervals where there exists a spatiotemporal overlap.

As previously mentioned, a k-sequence, where $k > 2$, is essentially formed by (k-1) *follow* relationships occurring between each consecutive instance pair. That is to say, there are (k-1) 2-sequences contained in a k-sequence. For sequences of length 3 or more, the chain index is defined as the minimum chain index of all 2-sequences contained.

Formally, for a 2-sequence, $ISq_r = (ins_{r_1} \blacktriangleright ins_{r_2})$, the significance of the *follow* relationship is assesed as follows:

$$ci(ISq_r) = \begin{cases} J^*(tw_{r_1}, h_{r_2}) & \text{if } ins_{r_1}.t_s < ins_{r_2}.t_s, \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where t_s represents the starting time of an instance, and J^* for the tail window and head segments is defined as:

$$J^*(tw, h) = \frac{V_{til^{xco}}(tw \cap h)}{V_{til^{xco}}(tw \cup h)} \text{ - (See [142])} \tag{5.4}$$

For a k-sequence $ISq_i = (ins_{i_1} \blacktriangleright ins_{i_2} \blacktriangleright \ldots \blacktriangleright ins_{i_k})$, where $k > 2$, the significance is assessed as follows:

$$ci(ISq_i) = \min_{1 \leqslant j < k} (ci(ins_{i_j} \blacktriangleright ins_{i_{j+1}})) \tag{5.5}$$

The instance sequences are considered as significant if their chain index value is greater than a user-defined chain index threshold (ci_{th}). The chain index is an antimonotonic

measure. Antimonotonicity (downward closure property) is a crucial property for frequent pattern mining, as it helps pruning the search space efficiently. The property refers to the phenomenon that for any k-sequence, if the k-sequence is significant, any of its subsequences are also significant, and the k-sequence cannot be significant, if at least one of its subsequences is not significant. Next, we will present the proof of antimonotonicity for the chain index.

Lemma: The chain index is antimonotonic.

PROOF: Given $ISq_j = (ins_1 \blacktriangleright ins_2 \blacktriangleright \ldots \blacktriangleright ins_k)$ is an instance sequence. Let pre_j be the length-(k-1) prefix subsequence of ISq_j and suf_j be the length-(k-1) suffix subsequence of ISq_j.

$$\begin{aligned} pre_j &= (ins_1 \blacktriangleright ins_2 \blacktriangleright \ldots \blacktriangleright ins_{k-1}), \\ suf_j &= (ins_2 \blacktriangleright ins_3 \blacktriangleright \ldots \blacktriangleright ins_k). \end{aligned} \tag{5.6}$$

For any chain index threshold ci_{th}, if ISq_j is significant:

$$\begin{aligned} ci_{th} &\leqslant ci(ISq_j), \\ ci_{th} &\leqslant \min(ci(i_1 \blacktriangleright i_2), \ldots, ci(i_{k-1} \blacktriangleright i_k)). \end{aligned} \tag{5.7}$$

The chain indexes of subsequences are defined as:

$$\begin{aligned} ci(pre_j) &= \min(ci(i_1 \blacktriangleright i_2), \ldots, ci(i_{k-2} \blacktriangleright i_{k-1})), \\ ci(suf_j) &= \min(ci(i_2 \blacktriangleright i_3), \ldots, ci(i_{k-1} \blacktriangleright i_k)), \end{aligned} \tag{5.8}$$

Then, $ci_{th} \leqslant ci(ISq_j) \leqslant ci(pre_j)$ and $ci_{th} \leqslant ci(ISq_j) \leqslant ci(suf_j)$, hence, chain index is antimonotonic.

5.2.2 *Prevalence of the Event Sequences*

Event sequences are derived from significant instance sequences. To measure how common a particular event sequence is, we will use the *participation index* measure. The participation index is defined in [110], and signifies the importance of an event sequence. For an event sequence, $ES_j=(e_{j_1} \rhd .. \rhd e_{j_k})$, the participation index of the event sequence is the minimum of participation ratios (pr) of the event types in the sequence.

$$pi(ES_j) = min(pr(e_{i_1}|ES_j), \ldots, pr(e_{i_k}|ES_j)) \tag{5.9}$$

The participation ratio of an event type (e_i) on an event sequence (ES_j) is the ratio of number of unique participators of e_i's instances to the total number of event instances of e_i.

$$pr(e_i|ES_j) = \frac{|\{ins_i|ins_i \in ISq_i \wedge ins_i.\mathcal{E} = e_i \wedge ISq_i \textit{ of-type } ES_j\}|}{|\mathbb{I}_{e_i}|} \tag{5.10}$$

where $|\cdot|$ shows the set size. Event sequences are considered as prevalent, if and only if the participation index of the event sequence is greater than the user-defined participation index threshold (pi_{th}).

5.2.3 *A Discussion on the Ambiguity of Allen's Temporal Algebra and How We Solve It*

In Allen's temporal algebra [7], any two time intervals can have one and only one relationship. While theoretically the algebra is not ambiguous, the same algebraic relation can quantitavely represent remarkably different situations. Additionally, a simple temporal predicate can be represented by more than one algebraic relationships. The lack of robustness in the algebra creates the ambiguity for knowledge discovery. For instance, in our spatiotemporal follow relationship, the *starts after* predicate can be represented

by five different relationships, and multiple follow relationships cannot be robustly captured by using only Allen's algebra.

Moerchen suggests the usage of thresholds and fuzzy extensions to the temporal algebra in order to overcome the ambiguity problems [143]. Following this suggestion, we addressed this problem by adapting two strategies:

1. Instead of using Allen's temporal algebra for starts after predicate, we only check the start times of the potentially sequence forming instances. This check is not based on the intervals, but only the start times of the instances.

2. To capture the sequence forming behavior, we introduced the tail window and head concepts for spatiotemporal event instances. The second predicate of the follow relationship is the spatiotemporal overlap. This predicate is particularly beneficial when checking the strength of the sequence forming behavior at regions of interest by translating the sequence forming behavior to a spatiotemporal co-occurrence relationship.

These two strategies enable us to

- Conveniently and efficiently inspect the starts after temporal predicate,
- Build a robust spatiotemporal follow relationship,
- Most importantly, create a flexible event sequence generation framework with the parameterized tail window and head concepts.

5.3 Apriori-based Algorithms for Mining Spatiotemporal Event Sequences

In this section, we will discuss two Apriori-based spatiotemporal event sequence mining algorithms. The seminal Apriori algorithm, proposed by Agrawal and Srikant [67], is designated for frequent itemset mining from transactional databases. The algorithm proceeds by identifying frequent individual items and extends them to larger and larger

itemsets as long as the discovered itemsets are sufficiently frequent in the transactional database. Apriori uses a bottom up approach, where frequent sub-itemsets are extended one item at a time (with candidate generation), and the candidate itemsets are tested against the database. In other words, Apriori algorithm generates candidate itemsets of size-k from itemsets of length-$(k-1)$. Then, it prunes the candidates which have one or more infrequent sub-itemsets. After that, it scans the transaction database to determine frequent itemsets among these candidates. This procedure is repeated iteratively until no candidate itemset can be generated.

Our Apriori-based algorithms follows a similar approach, where we generate candidate spatiotemporal event sequences, prune the infrequent ones, and repeat the process until no further extension is possible. Our first mining algorithm is called *NaïveApriori* and the second one is called *SequenceConnect*. Both of these algorithms share the same initialization steps, where we create heads, tails, and tail windows from instances.

5.3.1 *Initialization*

In the initialization steps, we create the heads and tail windows of all the instances in the set of all instances ($\mathbb{I}$), and store them for further use in a map structure. The pseudocode for the initialization steps can be seen in Algorithm 3. For each instance, the initialization procedure creates head and tail windows as described in Section 5.1.1, and inserts them to the head ($\mathbb{H}$) and tail window ($\mathbb{TW}$) maps. Both $\mathbb{H}$ and $\mathbb{TW}$ collections are designed as two-level maps that store mappings from event types to instance identifiers, and from instance identifiers to head or tail window trajectory segments. Both the head and tail windows of the instances are stored in the form of an evolving region trajectory.

5.3.2 *Naïve Apriori Algorithm*

The initial iteration of the classical Apriori algorithm discovers frequent size-1 itemsets [67]. In spatiotemporal event sequence mining, our initial iteration identifies the length-2

Algorithm 3: Initialization steps, creation of head and tail windows

Input: The set of all instances ($\mathbb{I}$) and the parameter map ($params$) required for head and tail window generation (If interval-based hIn, tIn, bd, tv, if ratio-based hR, tR, bd, tvR)

Output: Head and tail windows of instances in $\mathbb{I}$ ($\mathbb{H}$, $\mathbb{TW}$)

```
1 Algorithm Initialize(𝕀, params)
2     foreach ins_i in 𝕀 do // for each instance in the dataset
          /* create head segment and add it to the map                    */
3         H_i ← ins_i.CreateHead(params) ;
4         ℍ.Put(ins_i.ℰ, i, H_i) ;
          /* create tail window segment and add it to the map             */
5         TW_i ← ins_i.createTailWindow(params) ;
6         𝕋𝕎.Put(ins_i.ℰ, i, TW_i)
7     return ℍ and 𝕋𝕎
```

spatiotemporal event sequences. After finding length-2 sequences, in the iterative steps, we increase the length of the event sequences one event type at a time, and find the longer length event sequences.

We give the outline of Naïve Apriori-based spatiotemporal event sequence mining algorithm in Algorithm 4. The algorithm starts with head and tail window initialization (Step 1). Through Step 2 to 6, we demonstrate the initial Apriori iteration that finds prevalent length-2 event sequences. In the initial iteration step, we firstly, generate candidate length-2 event sequences (C-ESq), and then generate the (candidate) instance sequences (C-ISq) of length-2 candidate event sequences (Step 3 and 4). Candidate instance sequences are created by performing a spatiotemporal join operation (based on the spatiotemporal overlap of tail window and heads of the instances). For instance, for $\mathbb{E} = \{A, B, C\}$, candidate event sequences $(A \rhd A)$, $(A \rhd B)$, $(A \rhd C)$, $(B \rhd A)$, $(B \rhd B)$, $(B \rhd C)$, $(C \rhd A)$, $(C \rhd B)$, and $(C \rhd C)$ are created and stored in C-ESq list. Then, for each of them, we create the candidate instance sequences by joining the heads and tail windows. For example, in the case of $(A \rhd B)$, we join the tail windows of instances of-type A with the heads of instances of type B. Additionally, we check the start times of instances for starts after predicate of spatiotemporal follow relationship.

Algorithm 4: Naïve Apriori-based STES Mining Algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters, chain index threshold (ci_{th}), participation index threshold (pi_{th})

Output: Set of all prevalent spatiotemporal event sequences based on the given ci and pi thresholds

```
1  Algorithm NaiveAprioriSTESMiner(𝕀,𝔼, params, ci_th, pi_th)
2      ⟨ℍ, 𝕋𝕎⟩ ← Initialize(𝕀, params) ;
       /* Generate candidate event and instance sequences                    */
3      C-ESq ← GenerateCandidates(𝔼) ;
4      C-ISq ← GenerateInstanceSequences(C-ESq,ℍ,𝕋𝕎) ;
       /* Prune insignificant instance sequences                             */
5      S-ISq ← PruneInstanceSequences(C-ISq, ci_th) ;
       /* Prune event sequences based on pi_th (prevalence)                  */
6      P-ESq ← PruneEventSequences(C-ESq,S-ISq,pi_th) ;
7      k ← 2 ;
8      PS[k] ← P-ESq // PS[k] stores k-sequences
9      while PS[k] is not null do
           /* iterative steps:  generate and prune cadidate event sequences  */
10         C-ESq ← GenerateCandidates(PS[k]) ;
           /* Join the head and tail windows of instance sequences           */
11         C-ISq ← GenerateInstanceSequences(C-ESq, S-ISq) ;
12         S-ISq ← PruneInstanceSequences(C-ISq, ci_th) ;
13         P-ESq ← PruneEventSequences(C-ESq, S-ISq, pi_th) ;
14         PS[k+1] ← P-ESq ;
15         k ← k+1 ;
16     return PS ;
```

Later, we prune the length-2 candidate instance sequences based on their significance using ci_{th} and create length-2 significant instance sequences (S-ISq) (Step 5). Then, we prune the candidate event sequences using significant instance sequences based on the pi_{th} value and create prevalent length-2 event sequences (P-ESq) (Step 6).

After the initialization steps, the algorithm proceeds to the iterative steps for candidate sequence generation and testing. In the iterative steps, the length-$(k+1)$ candidate event sequences are discovered by self-joining the prevalent event sequences (P-ESq - length-2) discovered in the previous iteration (Step 10). Then, length-$(k+1)$ candidate instance sequences (*C-ISq*) are generated for each length-$(k+1)$ candidate event sequence found

in Step 10 (Step 11). The candidate event sequences generation is performed by joining head and tail window tables based on spatiotemporal follow predicates (spatiotemporal overlap and starts after). It should be noted that the spatiotemporal join predicate joins the head and tail windows of the instances as in the initialization step. Later, the candidate instance sequences are filtered using the chain index threshold (Step 12). Finally, prevalent sequences are discovered using significant instance sequences (Step 13). This process is continued, until no further prevalent event sequence of length-$(k+1)$ can be generated.

Algorithm 5: Apriori-based SequenceConnect Algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters, chain index threshold (ci_{th}), participation index threshold (pi_{th})

Output: Set of all prevalent spatiotemporal event sequences based on the given ci and pi thresholds

1 **Algorithm** `SequenceConnect(`$\mathbb{I}$,$\mathbb{E}$, $params$, ci_{th}, pi_{th}`)`
2 $\langle \mathbb{H}, \mathbb{TW} \rangle \leftarrow$ `Initialize(`$\mathbb{I}$, $params$`) ;`
`/* Generate length-2 candidate event and instance sequences, and prune */`
3 C-ESq $\leftarrow$ `GenerateCandidates(`$\mathbb{E}$`) ;`
4 C-ISq $\leftarrow$ `GenerateInstanceSequences(`C-ESq,$\mathbb{H}$,$\mathbb{TW}$`) ;`
5 S-ISq $\leftarrow$ `PruneInstanceSequences(`C-ISq, ci_{th}`) ;`
6 P-ESq $\leftarrow$ `PruneEventSequences(`C-ESq,S-ISq,pi_{th}`) ;`
7 $k \leftarrow 2$;
8 $ID_{sgf}[k] \leftarrow$ `GetInstanceIds(`S-ISq`) ;`
9 PS[k] $\leftarrow$ P-ESq `// PS[k] stores k-sequences`
10 **while** PS[k] *is not null* **do**
`/* iterative steps: generate and prune cadidate event sequences */`
11 C-ESq $\leftarrow$ `GenerateCandidates(`PS[k]`) ;`
`/* use identifiers for connection instead of spatiotemporal joins */`
12 $ID_{sgf}[k+1] \leftarrow$ `SequenceConnector(`$ID_{sgf}[k]$`) ;`
13 P-ESq $\leftarrow$ `PruneEventSequences(`C-ESq, $ID_{sgf}[k+1]$, pi_{th}`) ;`
14 PS[k+1] $\leftarrow$ P-ESq ;
15 $k \leftarrow k+1$;
16 **return** PS

5.3.3 *SequenceConnect Algorithm*

In the naïve algorithm, we proposed a solution that uses a brute-force approach when finding the candidate instance sequences. This requires a computationally expensive spatiotemporal join operation on every Apriori iteration. To alleviate the computational burden of expensive candidate instance sequence generation procedures, we propose the SequenceConnect algorithm, which employs the antimonotonic property of the chain index for efficiently discovering significant instance sequences. The following lemma is employed for discovering the instance sequences.

Lemma: If there exists $k-1$ significant 2-sequences such that $(ins_{i_1} \blacktriangleright ins_{i_2})$, $(ins_{i_2} \blacktriangleright ins_{i_3}), \ldots, (ins_{i_{k-1}} \blacktriangleright ins_{i_k})$; then, there is a significant length-k instance sequence, $ISq_i = (ins_{i_1} \blacktriangleright ins_{i_2} \blacktriangleright, \ldots, \blacktriangleright ins_{i_k})$.

Proof: The chain indices of all the $(k-1)$ length-2 instance sequences are greater than or equal to the ci_{th} ($ci(ins_{i_j} \blacktriangleright ins_{i_{j+1}}) \geqslant ci_{th}$), because they are significant. The chain index for ISq_i is the minimum chain index of all the 2-sequences it contains ($ci(ISq_i) = \min_{1 \leqslant j < k}(ci(ins_{i_j} \blacktriangleright ins_{i_{j+1}})$). Since all of the contained 2-sequences are significant, the minimum of their chain indices is greater than or equal to ci_{th}; thus, ISq_i is also significant ($ci(ISq_i) \geqslant ci_{th}$). □

We give the outline of the SequenceConnect algorithm in Algorithm 5. Different from the Naïve Apriori-based algorithm (shown in Algorithm 4) the SequenceConnect algorithm requires the instances participating in the significant chains to be stored (Step 8). Similar to the Naïve Apriori-based algorithm, the initial Apriori steps of candidate event sequence generation and pruning are the same (See Steps 2 to 6). The identifiers of length-2 significant event sequences, which are discovered in Step 8, are used in iterative steps for generating the candidate instance sequences, and determining the prevalent event sequences by pruning the instance sequences.

Similar to the Naïve Apriori algorithm, candidate event sequences are generated from the prevalent event sequences discovered in the previous iteration. However, this time

Algorithm 6: SequenceConnector Procedure

Input: The list of identifiers of length-k significant instance sequences
Output: The list of identifiers of the length-$(k+1)$ significant instance sequences

```
1 Procedure SequenceConnector(ID_sgf)
2     IDs^(k+1) ← [ ] // Longer length connected sequences
3     foreach isq_i, isq_j ∈ ID_sgf where i ≠ j do
4         if Matches(isq_i, isq_j) then
5             isq^(k+1) ← Merge(isq_i, isq_j) ;
6             IDs^(k+1).Add(isq^(k+1)) ;
7     return IDs^(k+1)
1 Procedure Matches(ISq_i, ISq_j)
      /* Let ISq_i be (id_i1 ► ... ► id_ik), and ISq_j be (id_j1 ► ... ► id_jk)   */
2     suffix_i ← (id_i2 ► ... ► id_ik) ;
3     prefix_j ← (id_j1 ► ... ► id_jk-1) ;
4     if suffix_i = prefix_j then
5         return True ;
6     else
7         return False ;
1 Procedure Merge(ins_i, ins_j)
      /* Let ISq_i be (id_i1 ► ... ► id_ik), and ISq_j be (id_j1 ► ... ► id_jk)   */
2     suffix_i ← (id_i2 ► ... ► id_ik) ;
3     last_j ← id_jk-1 ;
4     return Concatenate(suffix_i, last_j) ;
```

we do not use a spatiotemporal join based on overlap predicate in the iterative steps. Instead of the spatiotemporal join, we apply the efficient *SequenceConnector* procedure. The algorithm for the *SequenceConnector* procedure is provided in Algorithm 6. The procedure takes a list of length-k instance sequences (in the form of a list of participating instance identifiers), and returns the identifier list of length-$(k+1)$ instance sequences.

The *SequenceConnector* iterates on a nested loop, where the pairs of length-k instance sequences are merged to create length-$(k+1)$ instance sequences. The criterion for merging is suffix and prefix matching, which is shown in *Matches* procedure in Algorithm 6. Given two length-k instance sequences (ins_i, ins_j) to the *Matches* procedure, the procedure gets the length-$(k-1)$ suffix of the first one (the last $(k-1)$ participating instances

of ins_i) and the length-$(k-1)$ prefix of the first one (the first $(k-1)$ participating instances of ins_j). If the suffix of of ins_i and prefix of of ins_j are the same, the *Matches* procedure returns true. When two instance sequences matches, the *SequenceConnector* procedure merges them, using the *Merge* procedure shown in the final part of Algorithm 6. Lastly, these sequences are added to the identifier list of length-$(k+1)$ instance sequences ($IDs^{(k+1)}$ in Step 6 of Algorithm 6).

We will provide a simple example to clarify the *SequenceConnector* algorithm. Let a length-3 instance sequence, ISq_i, be $(ins_1 \blacktriangleright ins_2 \blacktriangleright ins_3)$. For ISq_i, the join operation essentially finds the instance sequences that starts with ins_2 and ins_3, and merges them with ISq_1. For the sake of example, let ISq_j be $(ins_2 \blacktriangleright ins_3 \blacktriangleright ins_4)$. The result of the merge operation between ISq_i and ISq_j is a length-4 instance sequence $(ins_1 \blacktriangleright ins_2 \blacktriangleright ins_3 \blacktriangleright ins_4)$.

In a nutshell, the SequenceConnect algorithm applies a join on the instance identifiers of the length-k instance sequences to create length-$(k+1)$ instance sequences. Similar to the Naïve Apriori-based algorithm, the last portion of the SequenceConnect algorithm is to test the prevalence of the spatiotemporal event sequences based on the pi_{th} value. The prevalent event sequences are passed to the next iteration of the algorithm, and the iterative process is continued until no further prevalent event sequences are found.

5.4 A Pattern Growth-based Approach for Mining Spatiotemporal Event Sequences

5.4.1 Event Sequences and Graph Representation

One of the difficulties of working with spatiotemporal instances is the computational complexity of spatial operations needed to identify the sequence forming behavior. In SequenceConnect algorithm, we mitigate this problem using *SequenceConnector* procedure, where we do not apply spatiotemporal join, but only a regular join on scalar instance identifiers. Another challenge for the Apriori-based spatiotemporal event se-

quence mining is the computational complexity of the candidate generation procedures. Apriori-based procedures virtually create a lattice and perform self-joins to move from bottom to the top of that lattice. When the lattice is sparse, the number of generated candidates is low. However, with the datasets resulting in a very dense lattice (many patterns being frequent), the candidate generation procedure becomes expensive due to the following reasons: (1) Candidate event sequence generation is a permutational procedure that requires us to find the matching subsequences in every iteration and (2) iteratively finding the matching instance sequences is neither computationally nor storage-wise efficient. Thus, when massive spatiotemporal trajectory datasets are processed, the join operations create a performance bottleneck for mining algorithms. To alleviate this problem, we propose to transform the instances and follow relationships into a graph structure, and mine the spatiotemporal event sequences from this graph.

The graph transformation creates a directed graph from event instances and the *follow* relationships. The instances, which participate in a 2-sequence, are transformed into graph's vertices. The *follow* relationships between instances are represented by the directed edges. Here, the paths in the graph become the instance sequences, and the frequently occurring paths become the event sequences. The task of mining can then be transformed to finding sequences of event types whose instances frequently form paths in the created graph structure. In the following parts of this section, we will initially describe the generation of event sequence graph, and, later, we will introduce our pattern growth-based algorithm for mining spatiotemporal event sequences.

Graph Transformation

The initialization step of the pattern growth-based algorithm includes not only the identification of the follow relationships, but also the creation of the event sequence graph, which is denoted as ESG. Formally, the event sequence graph is a structure that contains a set of vertices (V) and a set of edges (E) as shown in Eq. 5.11. The event sequence

Algorithm 7: Graph transformation of instances and spatiotemporal follow relationships

Input: Set of all instances ($\mathbb{I}$), head and tail window generation parameters ($params$), chain index threshold ci_{th}

Output: The event sequence graph (ESG) created from spatiotemporal follow relationships based on the ci_{th}

```
1  Algorithm GraphTransform(𝕀, params, ci_th)
2      ESG(V, E) = {} ;
3      foreach ins_i ∈ 𝕀 do
4          ESG.AddVertex((i, ins_i.ℰ));
5      ⟨ℍ, 𝕋𝕎⟩ ← Initialize(𝕀, params);
6      foreach TW_i ∈ 𝕋𝕎 do
7          foreach H_j ∈ ℍ do
               /* Check spatiotemporal overlap and starts after predicates    */
8              if STOverlaps(TW_i, H_j) and (ins_i.start < ins_j.start) then
9                  ci ← CalculateCI(TW_i, H_j); // Calculate ci value
10                 if ci > ci_th then
11                     ESG.AddEdge(i, j, ci); // Add an edge from vertex i to j
12     return ESG ;
```

graph is a directed weighted graph, where the vertices represent the event instances, while the weighted edges represent the spatiotemporal follow relationships and their significance as weights of the edges. The vertices, denoted as v_i in vertex set represents an instance (ins_i) and store the identifier of the instance, which is i, and the event type of that instance $ins_i.\mathcal{E}$. Each vertex is uniquely identified by its identifier, which is also the identifier of the instance. The edges are represented as triples comprising the identifier of source vertex, identifier of target vertex, and the weight of the edge. The source vertex identifier represents the identifier of the instance that is being followed (*i.e.*, followee instance), while the target vertex identifier represents the identifier of the instance that

follows (*i.e.*, follower instance). The weight represents the chain index value of the follow relationship from the followee instance to follower instance.

$$
\begin{aligned}
&ESG = (V, E) \\
&V = \{v_1 = [i_1, e_{i_1}], v_2 = [i_2, e_{i_2}], \ldots, v_n = [i_n, e_{i_n}]\} \\
&E = \{[i_{source_1}, i_{target_1}, w_1], \ldots, [i_{source_k}, i_{target_k}, w_k]\}
\end{aligned}
\tag{5.11}
$$

The algorithm for transforming the spatiotemporal follow relationships between instance to the event sequence graph structure is shown in Algorithm 7. The algorithm starts with creating an empty graph and adding the spatiotemporal event instances in the set of all instances as vertices of the graph (Step 3 and 4). Then, we create head and tail windows of the instances using the Initialize procedure shown in Algorithm 3. After creating the head and tail windows of instances, we identify each spatiotemporal follow relationship by checking the two predicates of the relationship. For two instances (ins_i and ins_j) we check the temporal starts after relationship ($ins_i.start < ins_j.start$) and the spatiotemporal co-occurrence relationship between followee instance's tail window and follower instance's head (STOverlaps(TW_i, H_j)) (See Steps 6 through 11). Note that for each follow relationship, we calculate the ci value and test it with the given ci_{th} value. Then, we add the edge with that particular weight from the followee (ins_i) to the follower (ins_j) if it is greater than the given ci_{th}.

By using the ESG, we aim to substantially reduce the storage requirements of our mining algorithm. In the ESG, we only store the unique instance identifiers with the instance's associated event type. The temporal and spatial data (time-geometry pairs), are not stored in the graph. For clarification, in Figure 5.4, we demonstrate the transformed version of our example dataset shown in Fig. 5.1.

Another important aspect of the event sequence graph is the acyclicity. As it can be seen from our example in Figure 5.4, the transformed graph is ordered on time dimension. This comes from the order imposed by the spatiotemporal *follow* relationship that

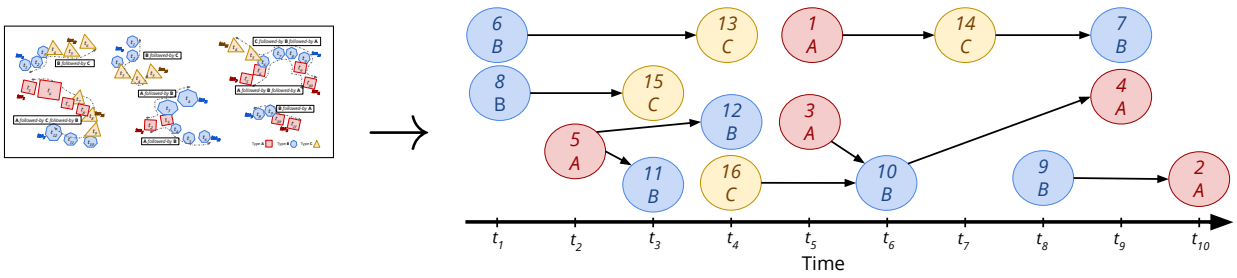


Figure 5.4: The graph representation of the spatiotemporal *follow* relationships and the instances shown in Fig. 5.1. The vertices representing instances are ordered based on their start time.

requires the start time of the followee must be less than the start time of the follower. This condition guarantees the non-existence of a feedback edge set (directed edges creating cycles), and imposes a topological order on the inspected instances based on their start times.

Lemma: The event sequence graph (ESG) is a directed acyclic graph.

Proof: Let $\mathrm{ESG}(V, E)$ be an event sequence graph, and vertices of the ESG be $V = \{v_1, v_2, \ldots v_n\}$. The edges are created when there exists a follow relationship between two instances. Namely, for each edge $v_{i_1} \rightarrow v_{i_2}$, we have $ins_i.start < ins_j.start$. Suppose ESG is not an acyclic graph, which means that there is a cycle, which can be found by a closed walk that starts and ends at the same vertex (*i.e.*, $path = v_{i_1} \rightarrow v_{i_2} \rightarrow \ldots\ldots v_{i_k} \rightarrow v_{i_1}$). Given the starts after predicate of the follow relationship, then the relationships in the path can be expanded as follows:

$$\begin{aligned} v_{i_1} \rightarrow v_{i_2} &\iff ins_{i_1}.start < ins_{i_2}.start \\ v_{i_2} \rightarrow v_{i_3} &\iff ins_{i_2}.start < ins_{i_3}.start \\ &\ldots \\ v_{i_k} \rightarrow v_{i_1} &\iff ins_{i_k}.start < ins_{i_1}.start \end{aligned} \tag{5.12}$$

Then, we can get the following inequality

$$ins_{i_1}.start < ins_{i_2}.start < ins_{i_3}.start < \dots ins_{i_k}.start < ins_{i_1}.start \tag{5.13}$$

However, $ins_{i_1}.start$ cannot be less than itself; thus, it is not possible to have such cyclic behavior in the event sequence graph. Essentially, the starts after predicate of the follow relationship enforces that for any edge in ESG, the start time of the source vertex must be less than the start time of the target vertex. This creates a topological ordering among all the connected vertices. As the vertices in ESG has topological ordering, ESG is a directed acyclic graph. □

5.4.2 EsGrowth Algorithm

In this part, we will explain our pattern growth-based spatiotemporal event sequence mining algorithm, which is called EsGrowth(that stands for ***E**vent **S**equence **Growth***). The EsGrowth algorithm initially discovers the significant *follow* relationships appearing between the instances and transforms them into a directed acyclic graph structure. Using the event sequence graph structure, the algorithm recursively discovers the frequently appearing event sequences using a pattern growth-based approach. The outline of EsGrowth can be seen in Algorithm 8.

Similar to the SequenceConnect algorithm, the EsGrowth algorithm initially discovers the significant follow relationships in graph transformation procedure based on the ci_{th} value (Step 3). After the transformation, the algorithm loops through all the event types in $\mathbb{E}$. This is to find the event sequences starting from a particular event type. Then for each event type e_i, using the *FindInstancesOf* procedure, we discover the non-leaf vertices of type e_i from the event sequence graph (Step 5). The *FindInstancesOf* procedure finds the instance sequences of the event sequences of a given event sequence.

Algorithm 8: Pattern growth-based EsGrowth algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters, chain index threshold (ci_{th}), participation index threshold (pi_{th})

Output: Set of all prevalent spatiotemporal event sequences based on the given ci and pi thresholds

```
1  Algorithm EsGrowth(𝕀,𝔼, params, ci_th, pi_th)
2      ES ← {}; // Global variable
3      ESG ← GraphTransform(𝕀, params, ci_th); // Global variable
4      foreach e_i ∈ 𝔼 do
5          Paths_(e_i) ←FindInstancesOf((e_i), ESG) ;
           /* (e_i) is a temporary 1-sequence to be extended                    */
6          GrowSequence((e_i), Paths_(e_i)) ;
7      return ES
1  Procedure GrowSequence(esq, Paths_esq)
2      SucPaths ← FindSuccessorPaths(Paths_esq) ;
3      foreach e_j ∈ 𝔼 do
4          esq_tmp ← (esq ▷ e_j); // Temporarily append event type to be inspected
5          Paths_esq_tmp ← FindInstancesOf(esq_tmp, SucPaths) ;
6          pi ← CalculatePI(esq_tmp, Paths_esq_tmp) ;
7          if pi > pi_th then
8              ES.Insert(esq_tmp) ;
9              GrowSequence(esq_tmp, Paths_esq_tmp)
```

In the initial iteration, we create a virtual length-1 event sequence that only represents the event type, and find the vertices of the event type in the graph.

After, we find the starting points of the paths in the graph (as $Paths_{(e_i)}$), we call the *GrowSequence* procedure. The *GrowSequence* procedure is shown in the second part of the Algorithm 8. In essence, the procedure extends the paths to find instance sequences of longer length event sequences. Firstly, the procedure finds the successor paths of the given event sequence (See $SucPaths$ in Step 2). After that, we iterate through all the event types, and extend the given event sequence with the event type to create a temporary event sequence (See esp_{tmp} in Step 4). Then, using the *FindInstancesOf* procedure, we extend the instance sequences (in the form of paths) found in the $SucPaths$. With this, we find the instance sequences of the temporary event sequence (esq_{tmp}) (Step 5).

Then, we calculate the participation index of the given event sequence, and test the participation index ($\mathtt{pi}$) with the threshold value $\mathtt{pi}_{\mathtt{th}}$. If the $\mathtt{pi}$ of event sequence is greater than the threshold, we add it to the list of prevalent event sequences ($\mathtt{ES}$) and call the *GrowSequence* with the found paths (representing the instance sequences of esq_{tmp}) of esq_{tmp} (See Steps 6 through 9).

This part of our algorithm extends the pattern growth-based PrefixSpan algorithm [61] to the event sequence graphs. For readers who are familiar with the PrefixSpan algorithm, the set of successer paths, $\mathtt{SucPaths}$, has a similar functionality with the prefix-projected databases [61]. In contrast to the prefix-projected databases, we only pass pointers to the vertices of the graph, which significantly reduces the storage requirements of the algorithm.

The *GrowSequence* procedure is a recursive procedure, where we call it for every prevalent event sequence based on the $\mathtt{pi}_{\mathtt{th}}$ value. For the event sequences who cannot pass the $\mathtt{pi}_{\mathtt{th}}$ test, we do not call this procedure because of the downward closure property. Note that if an event sequence is not prevalent, any of its super-sequences cannot be prevalent.

5.5 Mining the Most Prevalent Spatiotemporal Event Sequences: Top-$(\mathtt{R}\%, \mathtt{K})$ Approach

The Top-K approaches compute the rank for all items and finds the most important K patterns based on an interest measure. Getting the top-K patterns is one of the approaches for solving the problem of not having the prior knowledge, and previously used in many classical [144–147] and spatiotemporal [42, 148] frequent pattern mining approaches.

Previous spatiotemporal event sequence mining algorithms (SequenceConnect and Es-Growth) use significance and prevalence thresholds for discovering the spatiotemporal event sequences. These mining algorithms heavily rely on domain experts knowledge

to choose the optimal threshold parameters, which in some cases is not available. To tackle these issues, we propose an approach for mining the most prevalent K spatiotemporal event sequences from R% most significant follow relationships. In general, we will refer to this class of mining schemata as Top-(R%, K) spatiotemporal event sequence mining. We will propose two algorithms for performing Top-(R%, K) spatiotemporal event sequence mining: (1) Naïve Top-(R%, K)-ES-Miner and (2) Fast Top-(R%, K)-ES-Miner.

In our new class of algorithms, we will use the weights in the event sequence graph more effectively with a version of pattern growth-based EsGrowth algorithm. Instead of mining based on a set threshold, we will get a portion (R%) of the follow relationships from the event sequence graph. Similar to the EsGrowth algorithm algorithm, we will initially perform the graph transformation and later mine the spatiotemporal event sequences by incrementally growing them.

5.5.1 Naïve Approach

The algorithm for Naïve Top-(R%, K)-ES-Miner is outlined in Algorithm 9. This algorithm simply simulates the Top-(R%, K) STES mining using the EsGrowth algorithm. In essence, we find all the spatiotemporal event sequences based on the R% most significant follow relationships, and get the Top-K most prevalent ones.

The naïve algorithm starts by generating the event sequence graph with $ci_{th} = 0.0$, and determines the chain index threshold value that corresponds to the most significant $R\%^{th}$ follow relationship from the edges of the event sequence graph (See Step 2 and 3). In Algorithm 9, the ci value corresponding to the R% is denoted as ciTopR. Later, we utilize the EsGrowth algorithm, and call it with $ci_{th} = ciTopR$ and $pi_{th} = 0.0$. Here, for conciseness of the notation, we show that we call the regular EsGrowth; however, as we have already transformed the instances into the graph structure, we only filter the edges of the event sequence graph based on the given ci_{th}. After getting all the

Algorithm 9: Naïve Top-(R%, K) Spatiotemporal Event Sequence mining algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters, the ratio of significance (R), the number of STESs to be discovered (K)

Output: Set of top-K most prevalent spatiotemporal event sequences based on the given Rpc and K values

1 **Algorithm** `NaiveTopRK-EsMiner(`$\mathbb{I}$,$\mathbb{E}$, $params$, R, K`)`

2 $ESG(V, E) \leftarrow$ `GraphTransform(`$\mathbb{I}$, $params$, $ci_{th} = 0.0$`)` ;
`// find the ci value that is the R% highest ci value in follow edges (E)`

3 $ciTopR \leftarrow$ `findTopR%-Threshold(ESG.E, R)` ;
`// Mine using the EsGrowth, set ci to ciTopR and pi to 0.0`

4 $ES \leftarrow$ `EsGrowth(` $(\langle \mathbb{I}, \mathbb{E}, params \rangle$ *as* $ESG)$, $ci_{th} = ciTopR$, $pi_{th} = 0.0$`)`;

5 $ES^{TopRK} \leftarrow$ `Top(K); // Get top-K most prevalent and return`

6 **return** ES^{TopRK}

prevalent spatiotemporal event sequences, we get the K most prevalent ones based on their prevalence index (pi) values.

5.5.2 *Fast Top-*(R%, K) *Approach*

As mentioned earlier, the naïve approach is simply a simulation of the$Top-(R\%, K)$ spatiotemporal event sequence mining with EsGrowth algorithm. One issue with the naïve approach is that we set the pi_{th} value to 0.0, which can be very problematic, and lead to finding many spatiotemporal event sequences of greater sizes that may have low participation index values. In the Fast Top-(R%, K) approach, we employ a dynamic update mechanism for the pi values.

The fast mining algorithm for Top-(R%, K) spatiotemporal event sequence discovery can be seen in Algorithm 10. The algorithm can be seen as a version of EsGrowth algorithm with dynamic pi_{th} value updates.

The Fast Top-(R%, K)-ES-Miner algorithm starts with creating an empty sorted list, where we store the mappings of pi values and spatiotemporal event sequences (Step 2). The sorted list is denoted as TopES, and its maximum capacity is set to K. When,

Algorithm 10: Fast Top-(R%, K) Spatiotemporal Event Sequence mining algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters, the ratio of significance(R), the number of STESs to be discovered (K)

Output: Set of top-K most prevalent spatiotemporal event sequences based on the given Rpc and K values

```
1  Algorithm FastTopRK-EsMiner(𝕀,𝔼, params, R, K)
       /* Create an empty sorted list (on pi values) of K event sequences   */
       /* TopES = [⟨PI_1, esq_1⟩,... ⟨PI_k, esq_k⟩] and PI_i ⩾ PI_{i+1}      */
2      TopES ← SortedList(max. capacity= K) ; // global variable
3      ESG(V, E) ← GraphTransform(𝕀, params, ci_th = 0.0) ;
       // find the ci value that is the R% highest ci value in follow edges
          (E)
4      ciTopR ← findTopR%-Threshold(ESG.E, R) ;
5      ESG^f ← CIFilter(ESG.E, ciTopR) ;
6      foreach e_i ∈ 𝔼 do
7          Paths_(e_i) ←FindInstancesOf((e_i), ESG^f) ;
           /* (e_i) is a temporary 1-sequence to be extended                */
8          DynamicGrowSequence((e_i), Paths_(e_i)) ;
9      return TopES
1  Procedure DynamicGrowSequence(esq, Paths_esq)
2      SucPaths ← FindSuccessorPaths(V_pre) ;
3      foreach e_j ∈ 𝔼 do
4          esq_tmp ← (esq ▷ e_j); // Temporarily append event type to be inspected
5          Paths_esq_tmp ← FindInstancesOf(esq_tmp, SucPaths) ;
6          pi ← CalculatePI(esq_tmp, Paths_esq_tmp) ;
           // Check with pi of currently K^th event sequence
7          if pi > TopES.Get(K).PI then
8              TopES.Insert(⟨pi, esq_tmp⟩) ;
9              DynamicGrowSequence(esq_tmp, Paths_esq_tmp)
```

a new spatiotemporal event sequence is added to the list, the list stores it based on the event sequence's pi value. The event sequences are sorted in a descending fashion. In other words, the first item in the TopES is the most prevalent spatiotemporal event sequence, while the tenth item corresponds to the tenth most prevalent spatiotemporal event sequence. When the list is full, an *insert* operation on this sorted list simply deletes the K^{th} item and adds the event sequence, if the pi value of the inserted spatiotemporal

event sequence is greater than the K^{th} item's pi value. In other cases, the *insert* operation is rejected.

After initializing the sorted $TopES$ list, we create the event sequence graph (with $ci_{th} = 0.0$) and filter the graph based on the $R\%$ value. The $R\%$ filtering is performed by first finding a ci cutoff point that corresponds to $R\%^{th}$ portion of the edge weights in the ESG (See $ciTopR$ in Step 4) and later removing the edges whose weights are less than the $ciTopR$ (Step 5). The filtered event sequence graph is denoted as ESG^f.

The above mentioned steps (Steps 2 to 5) of the algorithm can be seen as the initialization for Top-$(R\%, K)$ mining schema. Then, similar to the EsGrowth algorithm, we iterate through the event types (e_i) and find spatiotemporal event sequences that starts with a particular event type (Steps 6 to 8). In these iterative steps, we find the paths starting from a non-leaf instance vertex whose event type is e_i, and call the *DynamicGrowSequence* procedure. This procedure is similar to the *GrowSequence* procedure in Algorithm 8, but it dynamically updates the pi_{th} value by checking the pi value of the K^{th} most prevalent spatiotemporal event sequence (See the condition $pi > TopES.Get(K).PI$ in Step 7). At any particular time, the pi value of the K^{th} element in the sorted $TopES$ list corresponds to the pi_{th} value, and the sorted nature of the list guarantees the correctness of our results.

5.6 Bootstrap Approach: Mining Spatiotemporal Event Sequences without Thresholds

Bootstrap is a resampling technique for estimating the distribution of a statistic [149], and it is especially useful when there is no analytical form of help in estimating the distribution of the statistics of interest. Here, we treat the participation index (pi) values of STESs as a complex statistic to be obtained from the event sequence graph (ESG) structure, and have the opportunity to explain the prevalence of spatiotemporal event sequences as a distribution rather than a single value.

In our novel bootstrap approach, we resample the edges in the event sequence graph, and similar to the Top-$(R\%, K)$ approach, we discover the event sequences from a subgraph of event sequence graph and collect its result. Yet, we perform this operation many times based on a parameter, the number of bootstrap trials (denoted as η).

Algorithm 11: Bootstrap-based Spatiotemporal Event Sequence mining algorithm

Input: Set of all instances ($\mathbb{I}$), set of all event types $\mathbb{E}$, head and tail window generation parameters ($params$), resampling ratio (R), the number of bootstrap trials (ν)

Output: A complex map of spatiotemporal event sequences and their pi values found after ν bootstrap trials

Algorithm `Btsp-EsMiner`($\mathbb{I}$,$\mathbb{E}$, $params$, rR, ν)
- `/* Create an empty STES map storing pi values for each trial */`
- `/*` $BtspES = [\langle esq_1 \rightarrow [pi_1, \ldots, pi_\nu]\rangle, \ldots \langle esq_n \rightarrow [pi_1, \ldots, pi_\nu]\rangle]$ `*/`
- $BtspES \leftarrow [\,]$; `// global variable`
- $ESG(V, E) \leftarrow$ `GraphTransform(`$\mathbb{I}$, $params$, $ci_{th} = 0.0$`)` ;
- **foreach** j *from 1 to* ν **do**
  - $rESG \leftarrow$`EdgeResample(ESG, rR)` ;
  - $iES \leftarrow \{\}$;
  - **foreach** $e_i \in \mathbb{E}$ **do**
    - $Paths_{(e_i)} \leftarrow$`findInstancesOf(`(e_i), $rESG$`)` ;
    - `GrowBtspSequence(`(e_i), $Paths_{(e_i)}$, iES, $rESG$`)` ;
  - `BtspES.Append(iES)`
- **return** $BtspES$

Procedure `GrowBtspSequence(`esq, $Paths_{esq}$, iES, $rESG$`)`
- $SucPaths \leftarrow$ `FindSuccessorPaths(`$Paths_{esq}$`)` ;
- **foreach** $e_j \in \mathbb{E}$ **do**
  - $esq_{tmp} \leftarrow (esq \rhd e_j)$;
  - $Paths_{esq_{tmp}} \leftarrow$ `findInstancesOf(`esq_{tmp}, $SucPaths$`)` ;
  - $pi \leftarrow$ `CalculatePI(`esq_{tmp}, $Paths_{esq_{tmp}}$`)` ;
  - **if** $pi > 0.0$ **then**
    - `iES.Insert(`$\langle esq_{tmp} \rightarrow [pi]\rangle$`)` ;
    - `GrowSequence(`esq_{tmp}, $Paths_{esq_{tmp}}$, iES`)`

In Algorithm 11, we give the overview of the Btsp-EsMiner mining algorithm. The algorithm takes the set of all instances ($\mathbb{I}$), set of all events ($\mathbb{E}$), head and tail window generation parameters ($params$), resampling ratio (rR) that is the ratio between the

number of edges to be resampled and total number of edges in ESG, and the number of bootstrap trials to be performed (ν) as input parameters.

In a nutshell, the Btsp-EsMiner algorithm performs resampling of the edges in the ESG structure for ν times to estimate the pi value for the spatiotemporal event sequences. Similar to the earlier pattern growth-based algorithms, using *GraphTransform* procedure, our Btsp-EsMiner algorithm transforms the spatiotemporal follow relationships into the event sequence graph (ESG) structure. After the initialization, the algorithm performs ν bootstrap trials (See Steps 4 to 10). Each trial can be considered as a new call to EsGrowth algorithm on a randomly resampled subgraph of the full event sequence graph.

The iterative calls for each bootstrap trial can be summarized as follows. Firstly, the algorithm performs edge resampling based on resampling ratio (rR) parameter (Step 5). The resampling ratio parameter determines the ratio of the edges to be included in the new random subgraph. The randomly created subgraph is generated by selecting k edges where k is $\lceil |\text{ESG.E} \times \text{rR}| \rceil$. We denote the randomly created subgraph after edge resampling as rESG. Note that our weighted graph structure is not a multi-graph; therefore, we opt for resampling without replacement. After resampling, we perform a version of the EsGrowth algorithm and obtain the event sequences and their pi values (Steps 6 to 10). Lastly, we append them to the map structure (BtspES) for every iteration, and return the BtspES.

The discovery of spatiotemporal event sequences from the resampled subgraphs of ESG can be summarized as follows. The resampled subgraphs (rESG) For each resampled subgraph, we perform a recursive procedure similar to the EsGrowth algorithm. For each event type e_i, we find the non-leaf vertices of e_i. These vertices corresponds to the starting vertices in the paths (which represents the instance sequences). Next, we grow the sequences using the resampled graph and add the results to the intermediate event sequence list (iES) (See GrowBtspSequences procedure in the second part of Al-

gorithm 11). This can be considered as running the EsGrowth algorithm with $\texttt{pi}_{\texttt{th}} = 0.0$. Lastly, we append them to the map structure (BtspES) for every iteration, and return the BtspES, which contains the discovered STESs and a size-(ν) list of pi values for each discovered STES.

6 EXPERIMENTAL EVALUATION

In this chapter, we will present a comparative analysis of our algorithms using real-life solar event datasets. We will firstly provide details on our experimental settings, introduce our data sources, and explain the pre-processing steps (tracking and interpolation) that we applied to the raw solar event data. Later, we will discuss the efficiency of our algorithms in the context of runtime complexity and storage requirements. In the efficiency analysis, we will compare the different steps of the algorithm such as head and tail window generation, the identification of spatiotemporal follow relationships, and the discovery of spatiotemporal event sequences using instance sequences. In the relevancy analysis section, we will primarily inspect the characteristics of spatiotemporal event sequences found from our datasets. We will discuss the advantages and disadvantages of using different types of algorithms.

6.1 Experimental Settings and Solar Event Datasets

6.1.1 Lifecycle of Solar Event Data

Solar physics researchers entered the big data era with the launch of NASA's Solar Dynamics Observatory (SDO) mission, which captures approximately 60,000 high resolution images every day, and generates 0.55 petabytes of raster data each year [85]. The big data trend in solar data is anticipated to be sustained by the ground-based DKIST telescope, which is expected to generate three to five petabytes of data each year [9].

We illustrate the lifecycle of solar event data from images to evolving region trajectories in Figure 6.1 To process and analyze the data, NASA selected a consortium (Feature Finding Team, FFT) to produce a comprehensive automated solar event recognition

system for solar images captured by the SDO. The automated system contains many individual modules detecting the spatial locations of different types of solar events from the SDO data [123]. The detected solar event instances are object data with spatiotemporal characteristics [8]. Recently, the curated large-scale solar image datasets with labeled event regions was published in [150]. Next, we will briefly point out how the solar events are tracked and interpolated.

Tracking the Solar Events

The tracking algorithm for events are introduced by Kempton et al. in [23, 133]. The algorithm utilizes the locations and image parameters for linking the polygon based instances. Therefore, it creates spatiotemporal trajectory objects with extended geometric representations.

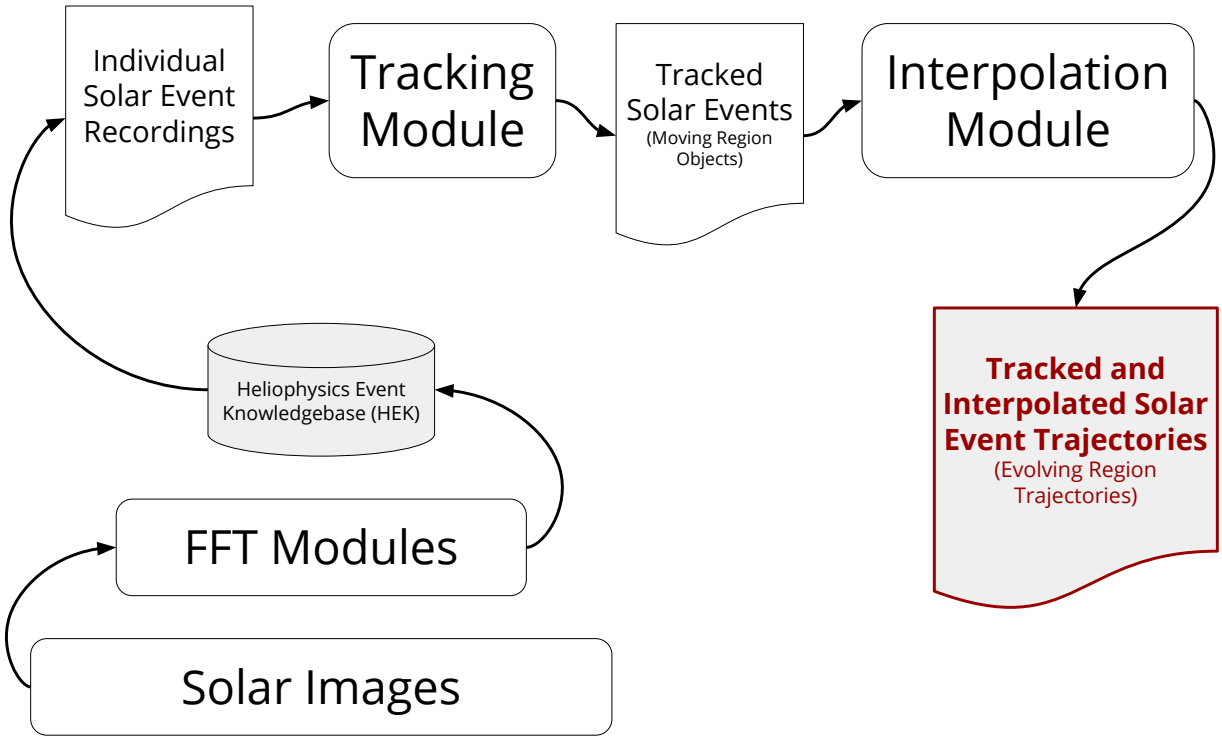


Figure 6.1: Lifecycle of solar event data

The goal of the tracking module is to link the solar events, which represent the same phenomenon, reported by the FFT modules into a chronologically ordered sequence representing the trajectories of the solar events. The algorithm, firstly links the individual event instances by projecting a detected object forward using the known differential rotation of the solar surface and searching for the potential detections that overlap with the search area at the next time step. If there is one and only one possible detection to be linked to, the algorithm links them together.

Then, the algorithm repeats the search for possible detections to link to. In these later steps, it considers detections that had multiple paths in their search region. To determine which path a tracked object takes, several aspects of visual and motion similarity are compared to produce a probable path for the object. The resultant paths are again fed into another iteration of the algorithm with larger and larger gaps between detections allowed to account for missed detections in the original metadata.

Interpolating the Tracked Solar Event Trajectories

Though the tracking algorithm generates moving region objects that can last over days, there are gaps in the individual solar event recordings. To increase the accuracy of our mining results, we fill these gaps using our specifically designed spatiotemporal interpolation techniques as appeared in our work [24].

As an example, for the case of filament events, they are alternately reported every 12 hours from the Kanzelhoehe and Big Bear Solar Observatories. On the other hand, active region events and coronal hole events are reported more frequently (approximately every 4 hours). This is essentially where the spatiotemporal interpolation methods are utilized, which allows the expansion of the tracked solar event data to the sites (locations on the Sun) where no events have been reported.

We proposed different interpolation strategies depending on the solar event type to be interpolated. The simplest interpolation method is the MBR-Interpolation, which is de-

signed for event types that are reported using their minimum bounding rectangles. For the event types that are reported using their complex polygon boundaries, we use the (Complex-Polygon Interpolation) CP-Interpolation algorithm, which uses the centroid-based shape signature along with the dynamic time warping alignment to match and regenerate the complex geometries. The Filament Interpolation (FI-Interpolation) is another interpolation method that includes the unique physical characteristics of the filament event type to make the interpolation more specialized. In addition to interpolating trajectory data, we also use extrapolation to estimate the shape of the geometries that do not belong to any track (single event as a moving object).

In summary, using the tracking algorithm, we are able to access and make use of solar events in the form of moving region objects, whose locations and shape and areal characteristics change continuously over time. We interpolate the solar event data to create more accurate spatiotemporal trajectories of solar events, which is in the form of spatiotemporal trajectories of continuously evolving polygons.

Table 6.1: Characteristics of the Solar Event Datasets

Dataset	Tag	#of Instances	#of Region Polygons
January 2012	Jan	2,072	159,773
February 2012	Feb	1,253	111,615
March 2012	Mar	2,027	157,374
April 2012	Apr	1,778	124,611
May 2012	May	2,258	199,390
June 2012	Jun	2,240	206,442
July 2012	Jul	2,387	182,601
August 2012	Aug	2,052	193,028
September 2012	Sep	2,123	186,906
October 2012	Oct	1,949	178,642
November 2012	Nov	2,058	161,930
December 2012	Dec	1,682	156,333

6.1.2 *Our Datasets*

To analyze the performance levels of our three different classes of algorithms we used twelve real-life solar event datasets. These datasets include the spatiotemporal instances of seven different solar event types that are: Active Regions (ar), Coronal Holes (ch), Emerging Flux (ef), Filaments (fi), Flares (fl), Sigmoids(sg), and Sunspots (ss). Each instance consists of region polygons, obtained from FFT module's reportings stored in the Heliophysics Event Knowledgebase (HEK) [151], and the regions are tracked and interpolated using the algorithms presented in [24, 133]. The characteristics of our real-life datasets can be seen in Table 6.1. Additionally, for each dataset, we show the number of event instances in our datasets for each different event type in Figure 6.2. For many datasets, there are usually a high number of Flare instances and low number of Sunspot instances. Additionally, the number of instances in May, Jun, and Jul datasets are higher.

6.1.3 *Implementation Details and Experimental Settings*

For the experimental evaluation, we implemented complex mining modules in Java programming language. Our modules can handle our base data types that we have presented in Chapter 1. To reiterate these base data types, we have moving region objects that consistently change their shape and location as our base data type for spatiotemporal event instances. These are represented using the raw trajectory data model, where each particular segment of the trajectory is represented discretely as time-geometry pairs. Each time-geometry pair object has a time interval object and a spatial geometry object that is of polygon type.

Then, on top of the base data types, we have implemented spatial, temporal, and spatiotemporal operations such as temporal starts after, spatial buffer, intersection, and union, and spatiotemporal intersection and union operators. These operations help per-

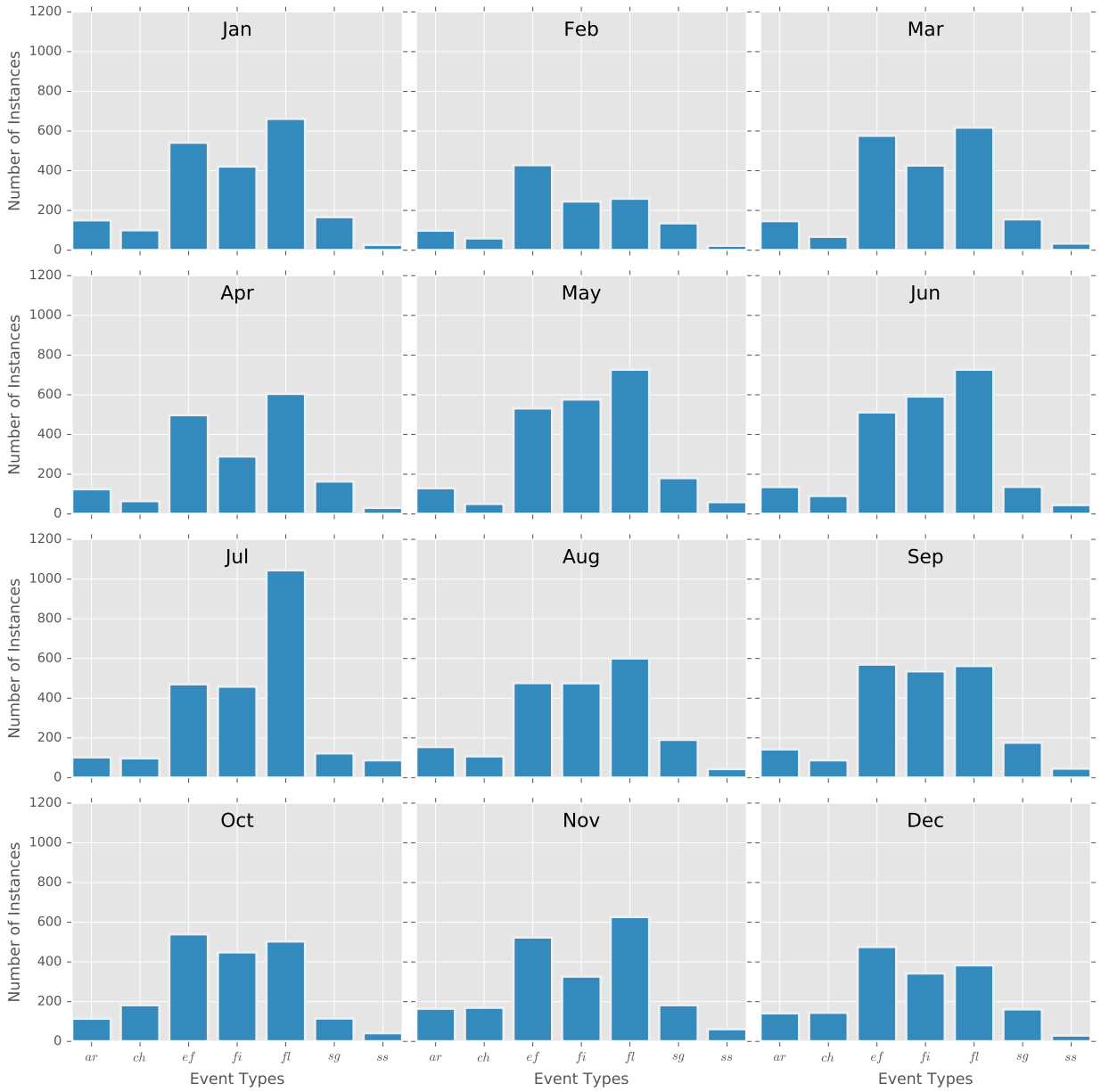


Figure 6.2: The number of instances per event type in our datasets

form the low-level operations when identifying the spatiotemporal follow relationship, as well as finding the significance of these follow relationships.

Next, we have a miner module, where we implement the mining algorithms presented in Chapter 5. The miner module makes use of the base data types and spatiotemporal operations. Additionally, we implemented an event sequence graph structure, which is a directed acyclic graph implementation. All of our implementations, excluding the Se-

quenceConnect algorithm, makes use of the event sequence graph. Our implementations do not use database connections to create a fair comparison environment. Note that the *SequenceConnect* may use a database when performing joins, while the event sequence graph structure can easily be stored in the main memory.

All of our modules are implemented in Java programming language (JDK version 8). For spatial operations, we used the JTS Topology Suite library [152]. For graph data types and operations, we used the JGraphT library [153]. We store our datasets in text files, and read them to memory for a fair comparison. Similarly, the graph structures are also stored in memory. All the experiments are performed on an Ubuntu virtual machine (in a dedicated server) with 1TB RAM and Intel Xeon processor (E7-8860, 2.20GHz).

We run our experiments using the following predefined head and tail window generation parameters. We set the head and tail ratio parameters to 0.1 and 0.2, respectively. We set the buffer distance parameter to 10 arcsec, and tail validity interval parameter to 2 hours. To show the characteristics of the algorithms in different parametric settings, we ran each algorithm 16 times. For all of our algorithms, we have two parameters of interest, and we set each of these parameters to four different values in each set of experiments and combine different results. For threshold-based approaches (*SequenceConnect* and EsGrowth algorithms) we set the chain index threshold (ci_{th}) to 0.1, 0.15, 0.2, and 0.25; and the participation index threshold (pi_{th}) to 0.04, 0.08, 0.12, and 0.16. For the Top-$(R\%, K)$ mining algorithms, we set the $R\%$ value to 0.2, 0.4, 0.6, and 0.8; and K value to 5, 25, 125, and 625. For the bootstrap-based algorithm, we set the resampling ratio (rR) to 0.1, 0.2, 0.3, and 0.4; and the number of bootstrap trials parameter (ν) to 50, 100, 150, 200. In total, we run the algorithms for 16 times for each dataset. Thus, in total we will be presenting our results based on a total of 960 experiments coming from 12 datasets, 16 runs, and 5 algorithms.

6.1.4 *Agenda of Our Experiments*

In the remainder of this chapter, we will analyze the efficiency of our mining algorithms primarily from the running time performance aspect. We ran 16 experiments with all *SequenceConnect*, EsGrowth, *Naïve Top-*(R%, K)*-EsMiner*, *Fast Top-*(R%, K)*-EsMiner*, and *Btsp-EsMiner* event sequence miner algorithms on 12 datasets. All of our algorithms share the initialization steps, where we generate heads and tail windows of the event instances and identify the spatiotemporal follow relationships. Excluding the *SequenceConnect* algorithm, we also transform the follow relationships into the event sequence graph structure. We will firstly inspect the running time performance of the initialization steps.

After that, we will compare the running time performance algorithms from different perspectives. Note that, we will not include the Naïve Apriori-based algorithm in our discussion, as we have shown that it is inefficient when compared to the *SequenceConnect* algorithm in our earlier work [154].

6.2 Initialization Times

In Figure 6.3, we illustrate the running times we recorded for the initialization steps of head and tail window generation (H-TW Generation) and spatiotemporal follow relationship discovery (Follow Discovery). We also show the number of vertices and edges in the event sequence graph for each dataset. Note that the number of edges show the number of spatiotemporal follow relationships we discovered from each dataset. The running times are aligned to major Y-axis and shown with green and blue lines, while the graph properties are aligned to minor Y-axis and shown with the grey and red bars.

From the results shown in Figure 6.3, we can see that the head and tail window generation time varies for each dataset. We can observe that part of it stems from the number of instances (vertices) in the dataset, and another factor is the number of individual region

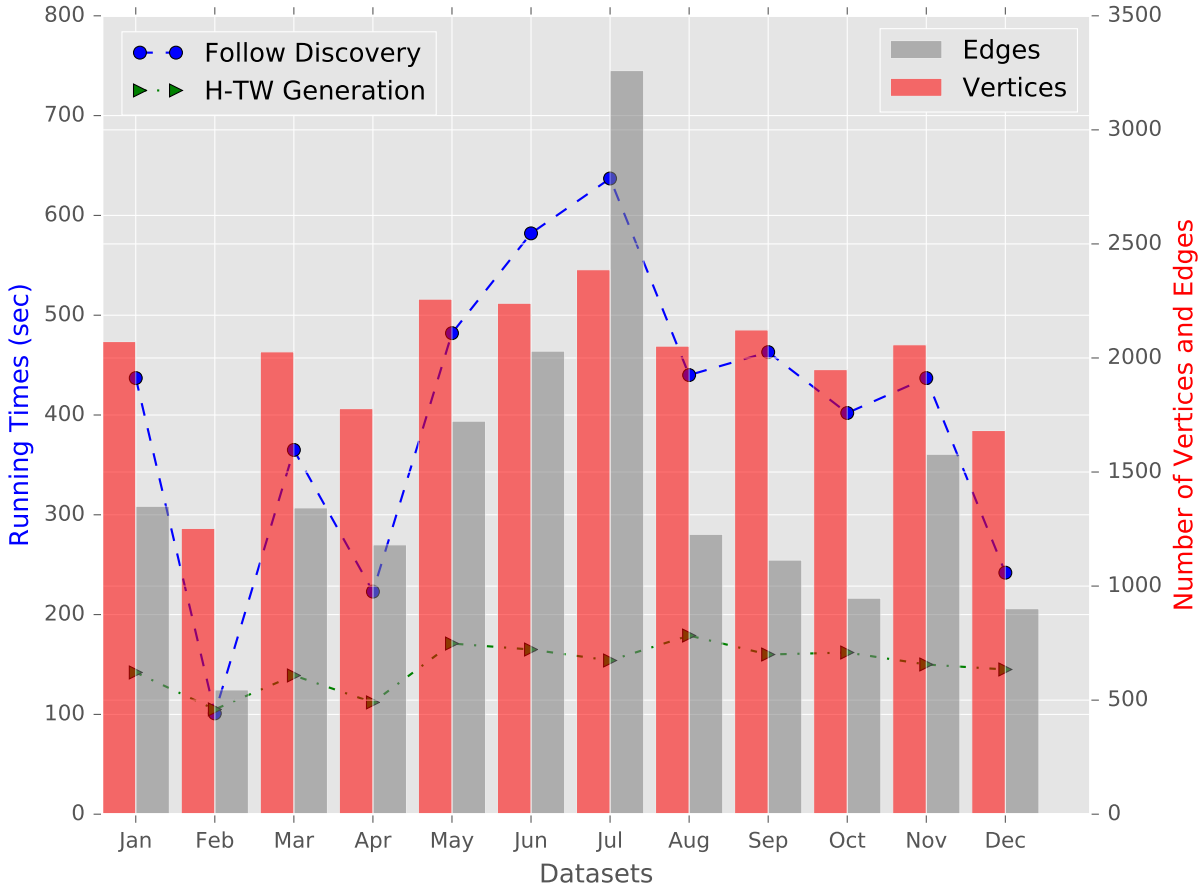


Figure 6.3: Average initialization times (follow discovery and head and tail window generation) for all of the algorithms, aligned on the major Y-axis. The number of edges and the number of vertices for each dataset are shown with bars aligned on the minor Y-axis.

polygons in the datasets, which can be seen in Table 6.1. We observe the highest head and tail window generation times are recorded for May, Jun, and Jul datasets, where we have the highest number of region polygons in the datasets. Similarly, the lowest head and tail window generation times are recorded for February and April datasets where we have the lowest number of region polygons.

The follow time is also inconsistent in our datasets. The follow time depends on the number of spatiotemporal follow relationships among the instances in the dataset. While they are not completely correlated, the number of edges (follow relationships) in the generated graph is a good indicator for the follow discovery time. Another factor that impacts the follow discovery time is the number of instances and region polygons,

because we get 10% and 20% of the instance trajectories as heads and tails (as $hR = 0.1$ & $tR = 0.2$).

For the case of the head and tail-window generation, our algorithm iterates through all the instances in the database and compute the time propagated and spatially buffered time-geometry pairs (representing the region trajectories). This process is done in linear time which explains the relation between the running time and the number of instances and region polygons. On the other hand, the event sequence graph generation algorithm iterates through the tail windows and performs a spatiotemporal join on overlap predicate with the head of instances. This makes the complexity of the follow relationship identification quadratic; however, since we apply a two-step filter based on the time-overlap and the spatial-overlap predicates the complexity becomes subquadratic (and very close to linear) with respect to the number of region polygons in follow time. It should be noted that, in the situation where there is a time requirement constraint, the user can shrink the size of the tail window (using tR, $t\nu$ and bd parameters) to decrease the amount of overlap; thus, reducing the number of follow relationships.

6.3 Overview of Running Times

In Figure 6.4, we illustrate the total running times of the algorithms (*SequenceConnect*, EsGrowth, *Fast Top*(R%, K)*-EsMiner*, and *Btsp-EsMiner*) on our datasets. We show the average running times of sixteen experiments for each algorithm for all our datasets. The running times of threshold-based algorithms and the *Top*(R%, K)*-EsMiner* are very similar on average. However, the bootstrap-based algorithm (*Btsp-EsMiner*) takes 25%to 60%more time than the threshold-based algorithms. This difference is much expected, as we perform 50 to 200 (based on the ν - number of bootstrap trials parameter) resampling and graph mining procedures. This means that we essentially apply the mining procedure on the resampled graphs, and the overhead generated by these operations are reflected on the average running times of the *Btsp-EsMiner* algorithm.

Another observation we can make from both Figure 6.3 and Figure 6.4 is that the total running times of the algorithms are dominated by the initialization times, where we apply the spatiotemporal operations. It should be noted that tail window generation as well as the follow relationship discovery (with ci calculations) are computationally expensive procedures.

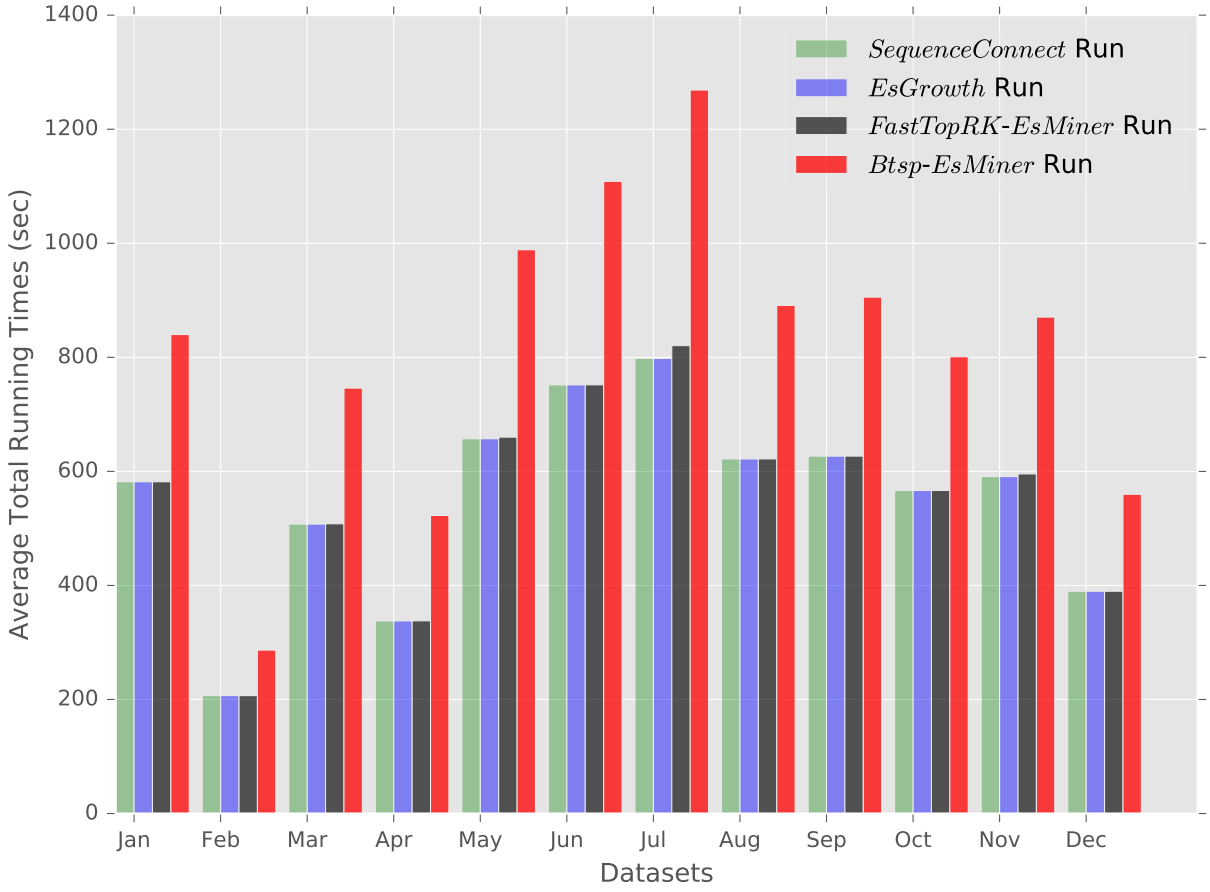


Figure 6.4: Total running times of the algorithms, averaged over sixteen individual runs of each algorithm for all the datasets

6.4 Analysis of Threshold-based Approaches

In the previous sections, we inspected the initialization and total running times of all the algorithms. In this part of our discussion, we will analyze the results from two of our threshold-based approaches that are: the *SequenceConnect* and the EsGrowth algorithms.

To remind the readers: the *SequenceConnect* algorithm is an Apriori-based algorithm, where we generate candidates and prune them by testing the sequences. The EsGrowth algorithm is pattern growth-based and it iteratively grows the sequences using the event sequence graph.

For all the different values of $\mathtt{pi}_{\mathtt{th}}$ and $\mathtt{ci}_{\mathtt{th}}$ parameters, we show the total running time (including initialization phase) in Figure 6.5 and the total number of spatiotemporal event sequences in Figure 6.6.

In a nutshell, we can observe that EsGrowth algorithm generally performs similarly with *SequenceConnect* algorithm. However, when there are more follow relationships (as in the case for the May, Jun and Jul datasets), we start observing the drastic differences between these two algorithms. This is much expected, as the number of the generated candidates increases the efficiency of the *SequenceConnect* algorithm significantly drops. In most cases, the bottleneck for *SequenceConnect* algorithm is the spatiotemporal follow relationships discovery (*i.e.*, the length-2 candidate instance sequence generation). However, when there are a lot of spatiotemporal follow relationships between instances (as in Jul dataset - See Figure 6.3) the candidate sequence generation and joins become the bottleneck of the algorithm.

In our earlier work [155], we also compared the *SequenceConnect* and EsGrowth algorithms, and illustrated the effect of varying the head and tail window generation parameters as well as the size of different datasets. Our findings here align with our earlier results, where more instances and more follow relationships impact the running time performance of the *SequenceConnect* algorithm immensely. Overall, for threshold-based approaches, we can conclude that EsGrowth algorithm is more efficient than *SequenceConnect* algorithm. This is because the EsGrowth avoids the expensive candidate generation and filtering steps. It is also worth noting that, *SequenceConnect* writes the significant instance sequences back to use it in the next iterations, which is an extra overhead. EsGrowth, on the other hand, exploits the efficient searching capabilities of the

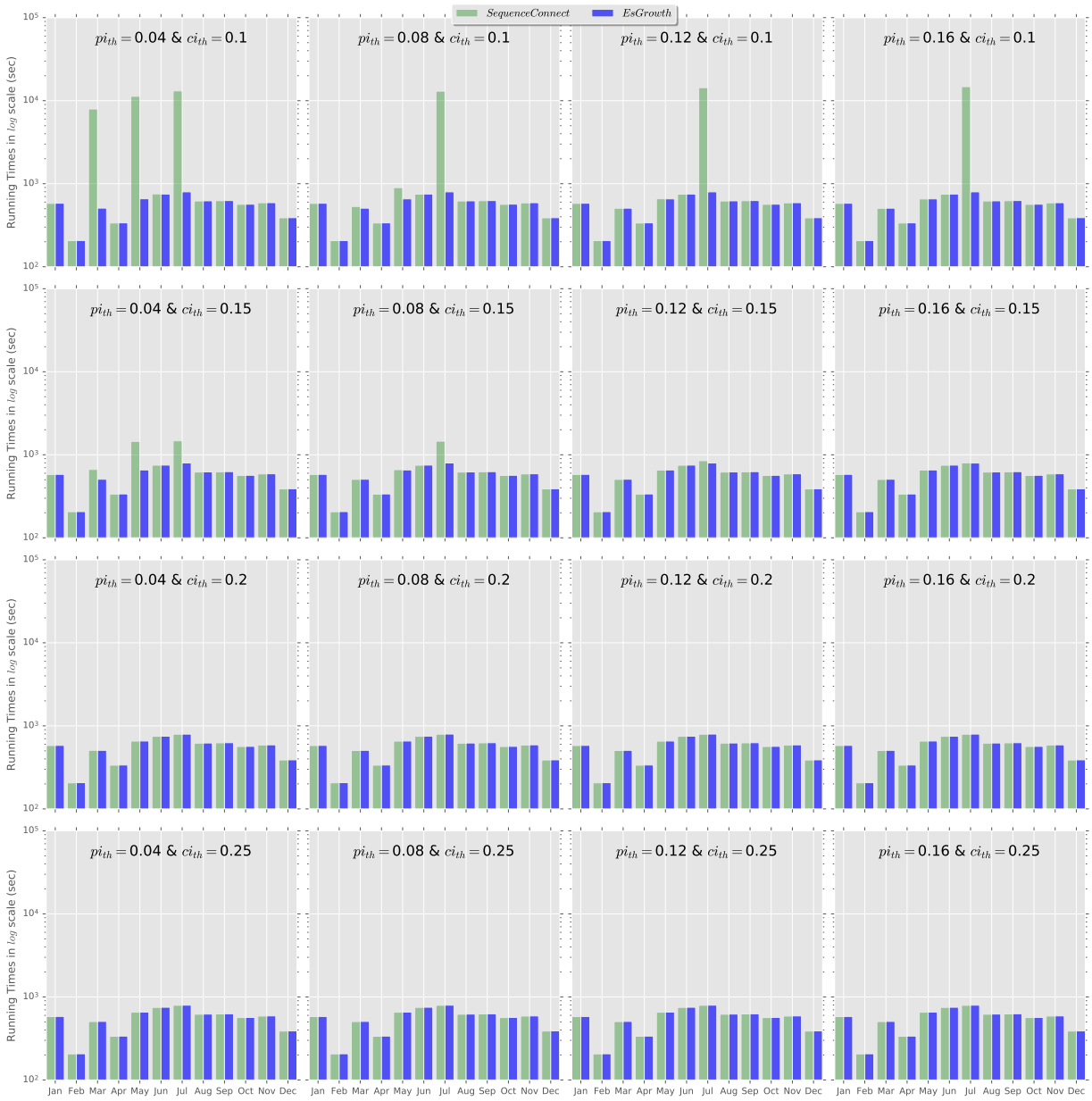


Figure 6.5: The running times of *SequenceConnect* and EsGrowth algorithms for all datasets under various threshold parameters

event sequence graph structure. For the EsGrowth algorithm, the event sequence graph generation is an overhead, and it can be seen on Figure 6.5, in the last row ($ci_{th} = 0.25$), where *SequenceConnect* takes slightly less running time. On the other hand, the recursive *GrowSequence* procedure is not much affected from the higher number of follow relationships as the graph structure allows efficient in-memory search for the discovery

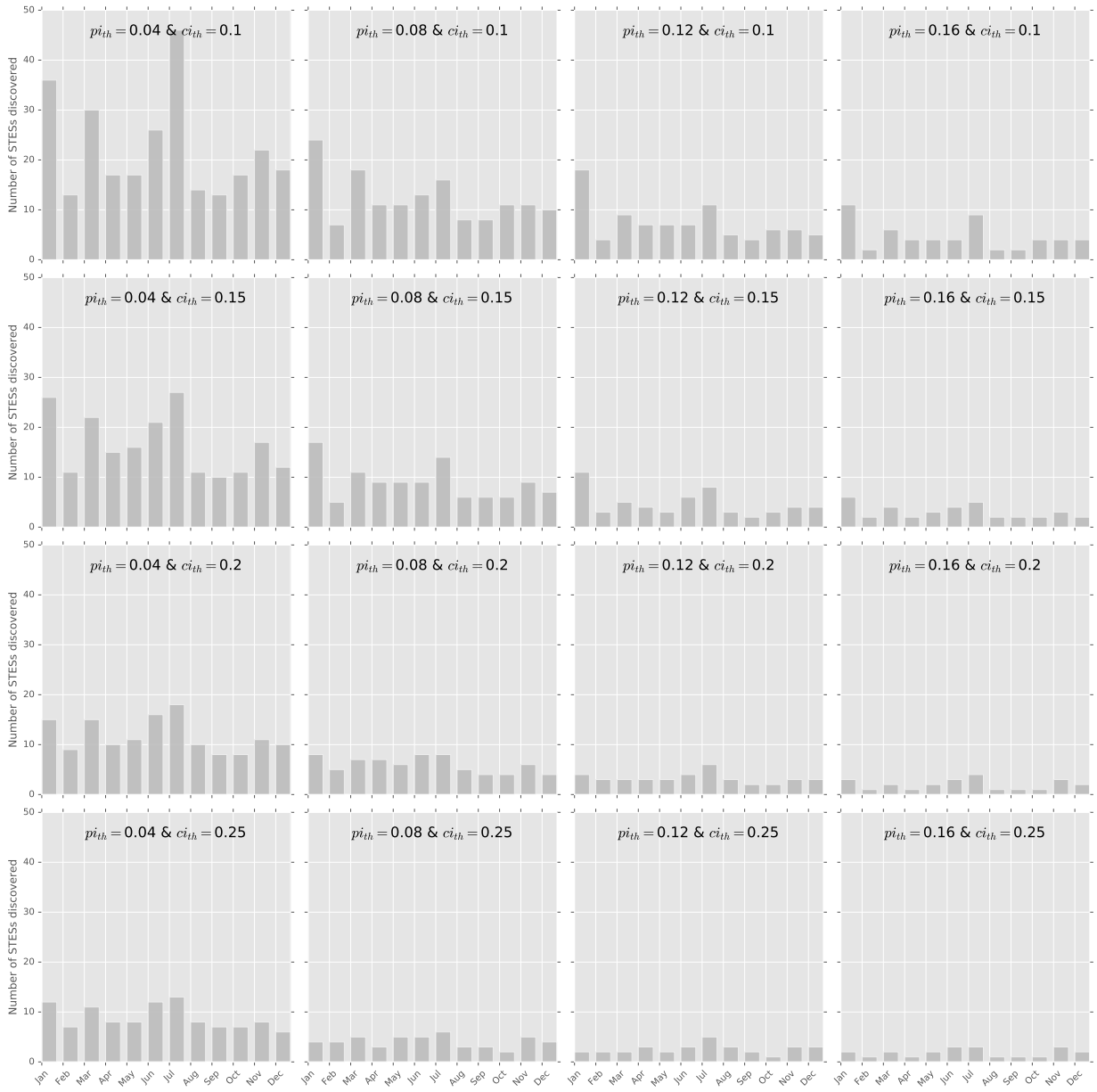


Figure 6.6: The number of spatiotemporal event sequences discovered in the threshold-based approaches with different $\mathtt{ci}_{\mathtt{th}}$ and $\mathtt{pi}_{\mathtt{th}}$ values. As they are the same, we showed only one bar for each different threshold.

of potential instance sequences. It is also worth mentioning that, as the graphs become denser and denser, the running time complexity as well as the storage complexity of both of our algorithms becomes exponential. Thus, for data analysis with large-scale datasets, it can be more feasible to use non-threshold-based approaches.

6.5 Analysis of Top-(R%, K) Approach

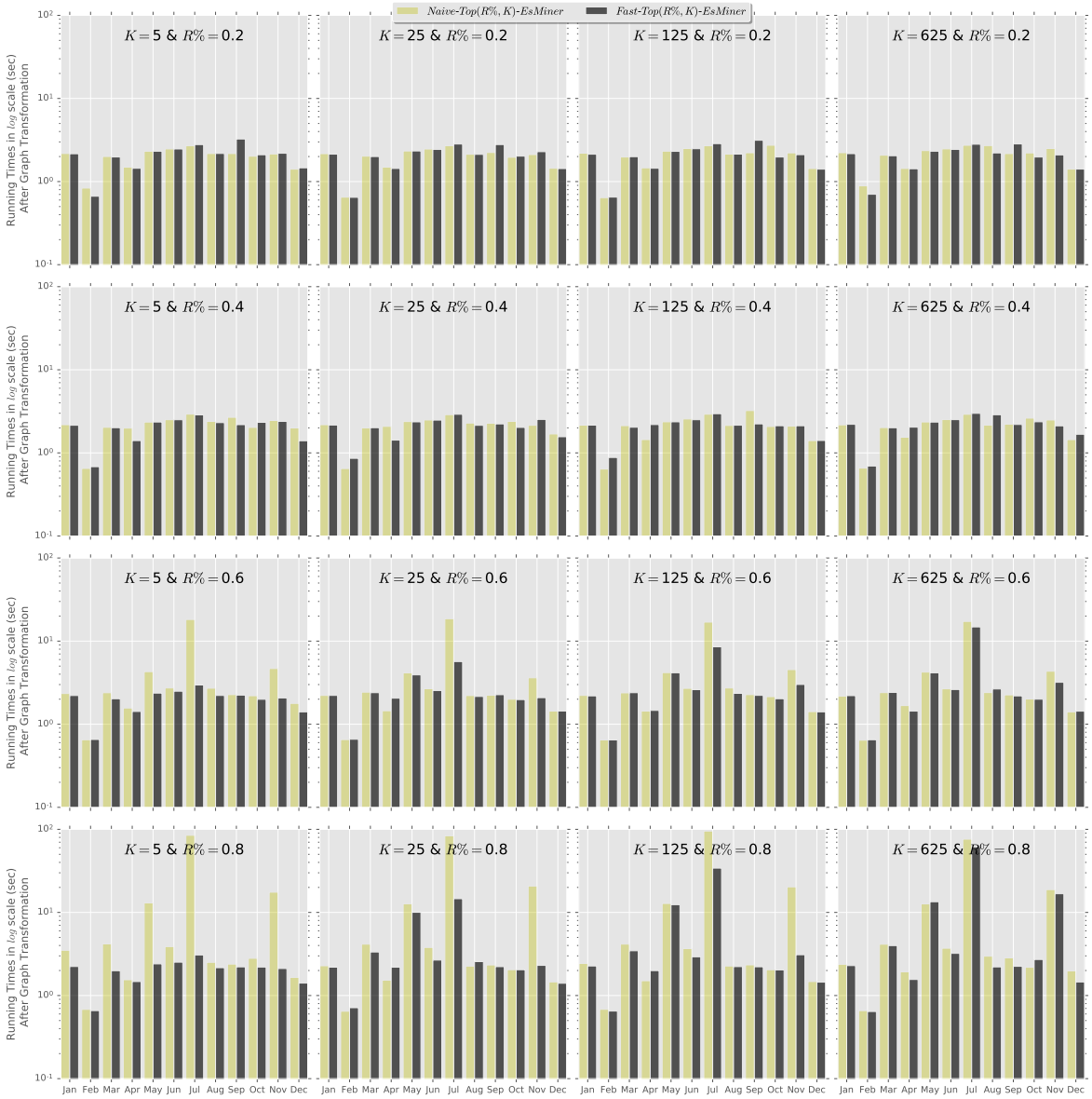


Figure 6.7: Running times of Naïve and Fast Top-(R%, K)-EsMiner

6.5.1 *Running Time Analysis of Top-(R%, K) Algorithms*

In this part of our discussion, we will compare the running times of our Top-*(R%, K)* STES mining algorithms. Later, we will compare the running times of the Top-*(R%, K)* algorithms with a run of EsGrowth algorithm with comparable `ci` and `pi` thresholds.

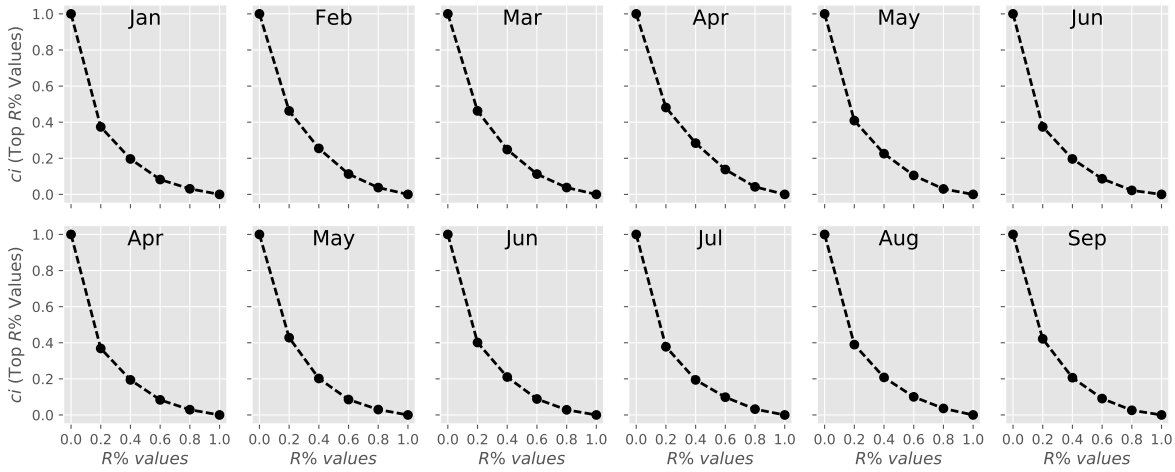


Figure 6.8: The ci values corresponding to the R% most significant follow relations

In Figure 6.7, we demonstrate the total running times of our algorithms after the initialization and graph transformation steps for each dataset. The K and R% values are displayed on each chart. The yellow bars show the running times for Naïve-Top-*(R%, K)*-EsMiner, while the black ones shows the running times for Fast-Top-*(R%, K)*-EsMiner. The running times are displayed in log scale. Additionally, we demonstrate the ci values corresponding the the R% most significant follow relationhips for each dataset.

For low R% values (*i.e.*, R% $= 0.2$ and R% $= 0.4$, displayed on the first two rows in Figure 6.7) we observe very similar running times. This is primarily caused by the number of discovered STESs being very few (often less than K), where the effect of dynamic $\mathtt{pi}_{\mathtt{th}}$ updates do not benefit the Fast-Top-*(R%, K)*-EsMiner algorithm. In other words, the $\mathtt{pi}_{\mathtt{th}}$ are never increased in the *DynamicGrowSequence* procedure since the sorted TopES list never reaches size K. It can also be observed from Figure 6.8, where R% $= 0.2$ and R% $= 0.4$ that we consistently get ci values of ~0.4 and ~0.2. This makes the filtered $\mathtt{ESG}^{\mathtt{f}}$ sparse, and reduces the number of discovered STESs in the end.

On the other hand, we start to see the running time differences when the R% value is increased (*i.e.*, R% $= 0.6$ and R% $= 0.8$, displayed on the last two rows in Figure 6.7). We see the largest differences in the datasets, where we have more dense ESGs such as in May, Jun, Jul, Nov datasets. The number of edges and vertices in these ESGs can

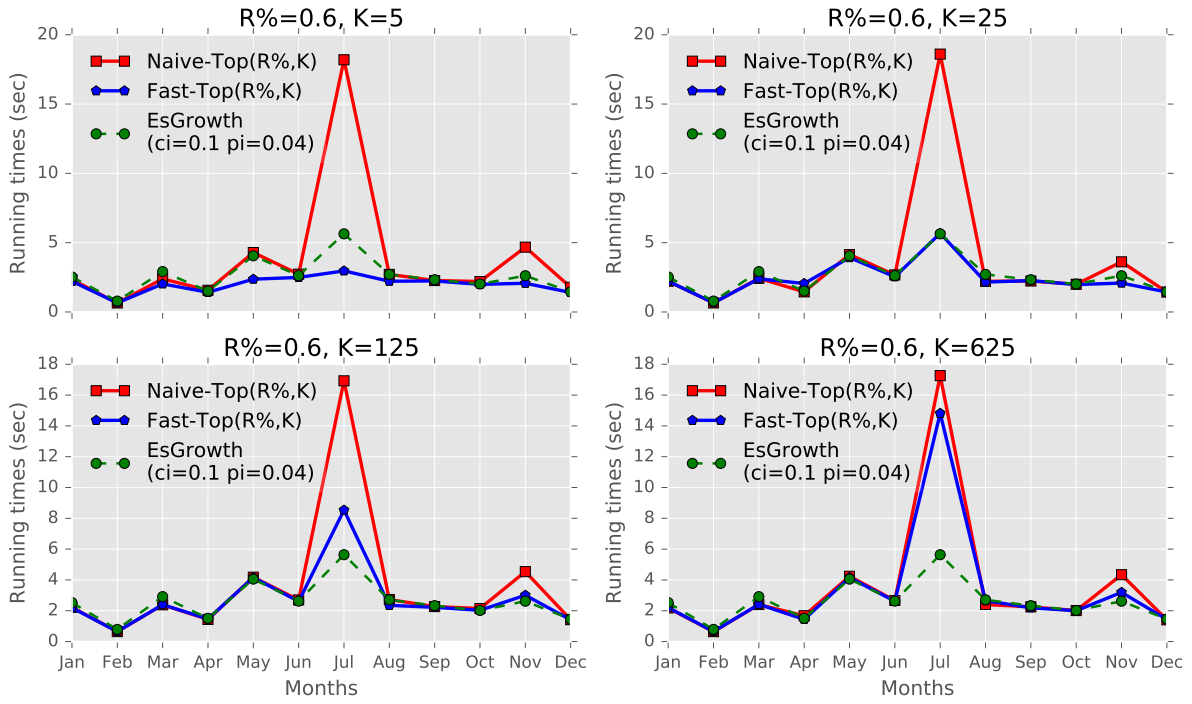
be seen in Figure 6.3. Additionally, when we increase the K values, the gap between the running times of Naïve and Fast Top-*(R%, K)*-EsMiner algorithms are closed, and we can see the largest differences when the K value is set to 5. Thus, we can conclude that for sparse ESG and high K values, Naïve and Fast Top-*(R%, K)*-EsMiner algorithms perform similarly; however, when the K value is reduced and as the ESGs gets denser, Fast-Top-*(R%, K)*-EsMiner algorithm becomes more and more efficient.

6.5.2 *Comparison of EsGrowth and Top-(R%, K) Approach*

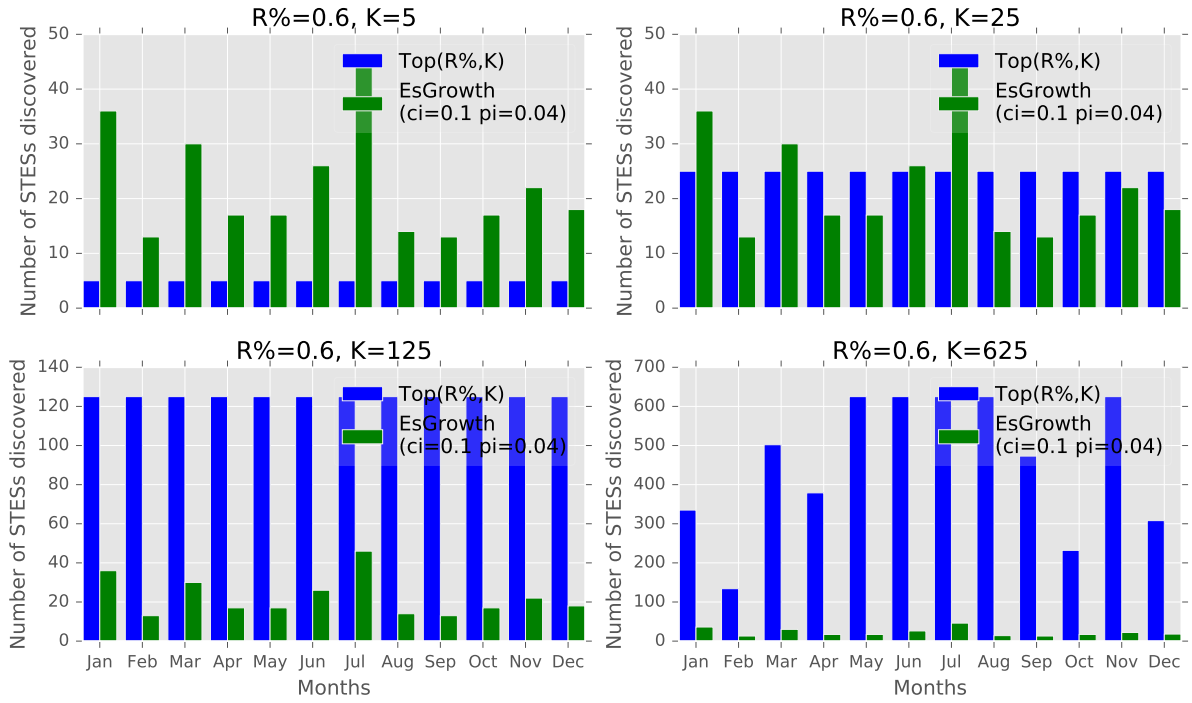
In this part, we will compare the efficiency of our Top-*(R%, K)* STES mining algorithms with threshold-based EsGrowth algorithm. While the algorithms of Fast-Top-*(R%, K)*-EsMiner and EsGrowth seems similar, their behavior is different due to the different parametric settings. In Fast-Top-*(R%, K)*-EsMiner, the running time is primarily dependent on the K and R% values. The R% value makes the ESG more sparse or dense, while the K value eventually bounds the number of STES to be discovered, where the algorithm can efficiently prune the search space using pi updates. In the EsGrowth, ci_{th} serves a purpose similar to R%, where we filter the edges in the ESG. However, the pi_{th} is constant, and the running time for EsGrowth is dependent on the data.

To show the above mentioned behavior, for our EsGrowth run, we picked a ci_{th} value ($= 0.1$), which would create an ESG that is comparable to the Top-*(R%, K)* experiments with R% value set to 0.6 ($K \in \{5, 25, 125, 625\}$). It can be seen from Figure 6.8 that the ci values for $R\% = 0.6$ is ~0.1 for all our datasets. In Figure 6.9.a, we show the running times of our EsGrowth (the same run displayed) and Naïve and Fast Top-*(R%, K)*-EsMiner algorithms. Figure 6.9.b shows the number of STESs discovered.

In Figure 6.9.a, we can see that the running times for Naïve-Top-*(R%, K)*-EsMiner algorithm does not change as we are not updating the pi values. For all the K values, Fast-Top-*(R%, K)*-EsMiner performs better than the Naïve one. When compared to EsGrowth, we see that running times for Fast-Top-*(R%, K)*-EsMiner algorithm drastically



(a)



(b)

Figure 6.9: (a) Running times of Naïve and Fast Top-(R%, K)-EsMiner compared to EsGrowth (b) The number of STESs from Top-(R%, K)-EsMiner and EsGrowth

change as we increase the K value. When $K = 5$, Fast-Top-*(R%, K)*-EsMiner performs up to 2x better than the comparable EsGrowth (in Jul dataset). When $K = 625$, Fast-Top-*(R%, K)*-EsMiner performs ~3x worse than EsGrowth. The reason for this is the number of discovered STESs in each dataset depends on different parametric settings.

In Figure 6.9.b, the green bars show the number of discovered STESs using the EsGrowth algorithm (the values are the same for all four sub-charts). We can observe that, as K value gets smaller, the running time of the Fast-Top-*(R%, K)*-EsMiner becomes less data dependent (See Mar, Jun, Jul, Nov datasets when $K = 5$ and $R\% = 0.6$). This is not the case for EsGrowth algorithm, where the pi_{th} efficiently eliminates the STESs in sparse ESGs such as Feb and Dec datasets, while it becomes inefficient for dense ESGs as in May and Jul datasets. We can conclude that the running time requirements of the Fast-Top-*(R%, K)*-EsMiner is less dependent on the sparsity of the underlying ESGs, and it can efficiently find the Top-K most prevalent STESs. However, it can be inefficient to use high K values with Fast-Top-*(R%, K)*-EsMiner as it cannot enable the dynamic pi updates in the algorithm.

6.6 Analysis of Bootsrap Approach

In this part of our experiments, the running time requirements of our *Btsp-ESMiner* algorithm will be compared to the EsGrowth algorithm. Earlier in Figure 6.4, we demonstrated the total running times of all of our algorithms for each dataset. In Figure 6.10, we demonstrate the average time spent on mining STESs from ESGs for EsGrowth (running time after initialization steps) and average time spent on a bootstrap trial (mining) for *Btsp-EsMiner* with different resampling ratio ($rR \in \{0.1, 0.2, 0.3, 0.4\}$) and number of bootstrap trial parameters $\nu = \{50, 100, 150, 200\}$.

In Figure 6.4, the blue bars show the average running time of EsGrowth algorithm with 16 different threshold parameter settings values. The red bars show the total running time of *Btsp-ESMiner* algorithm, which consists of 50, 100, 150, and 200 bootstrap

runs on the ESGs randomly bootstrapped with four resampling ratios. In Figure 6.10, we demonstrate the running times required for mining STESs from ESG. The initialization steps (H-TW generation and follow times shown in Figure 6.3) are omitted, and we report the average running times of threshold-based runs (with 16 different threshold parameters), and the average running time of 2000 ($4 \times (50 + 100 + 150 + 200)$) bootstrap trials for each dataset.

From the results shown in Figure 6.4, we can notice that the total running times required for discovering the STESs follow a very similar pattern to the initialization steps, and it can be observed that for threshold-based approach, the total running time is dominated by the initialization (See Figure 6.3). In EsGrowth experiments, as we use the

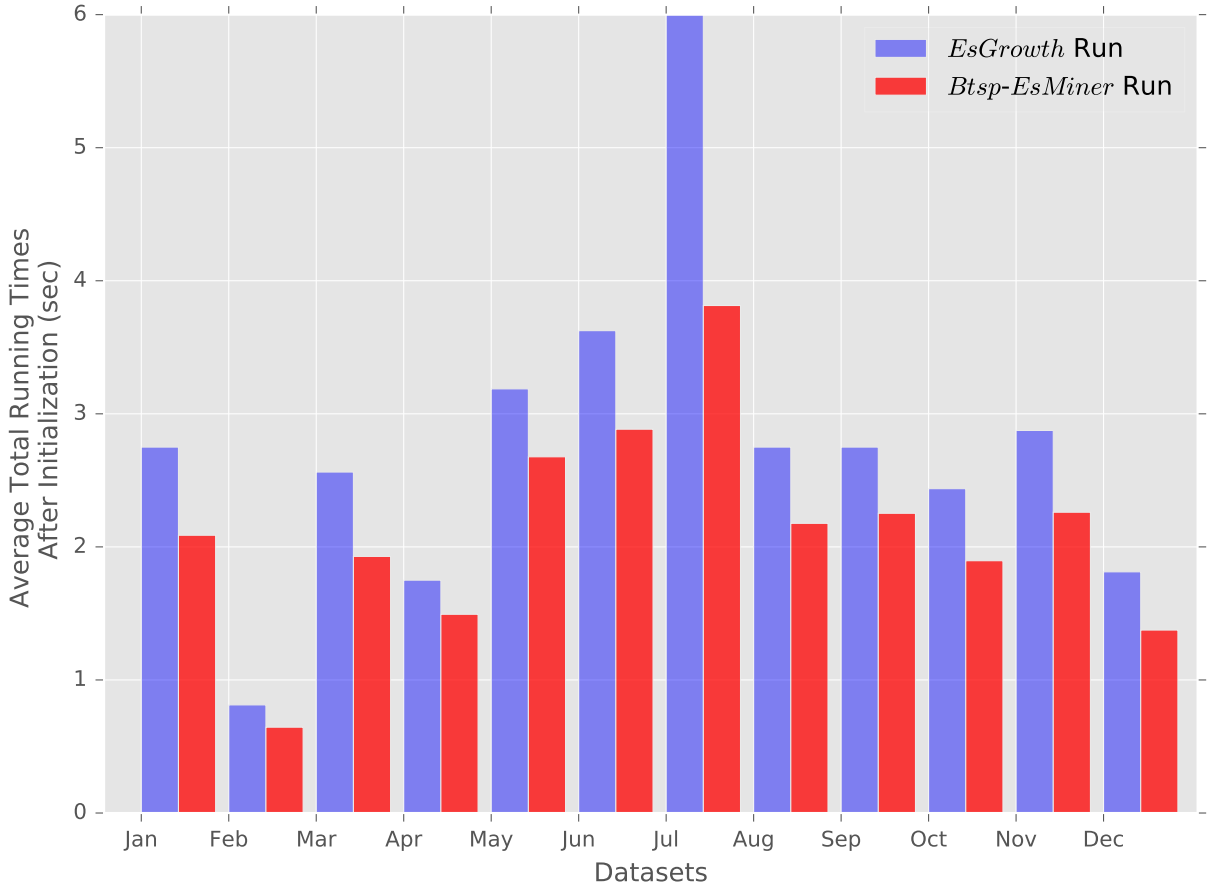


Figure 6.10: Average running time of EsGrowth is compared with the average running times of bootstrap trials

higher ci_{th} values for filtering, the insignificant follow relationships (or edges) are extensively pruned from the ESG, leading to very low graph mining times. Nevertheless, it is difficult to make conclusions about the trustworthiness of the STESs with high ci_{th} values. When we analyze the performance of the *Btsp-EsMiner* algorithm, we see that for sparse ESGs (such as Feb, Apr, and Dec datasets) the total running time of the *Btsp-EsMiner* is likely to be similar to the EsGrowth. On the other hand, for the denser ESGs (such as May, Jun, Jul, and Nov datasets), we observe slightly greater differences. This can be well explained with the algorithmic setup of bootstrap-based approach and the observations from Figure 6.10. The average ESG mining (*i.e.*, bootstrap trials) time of *Btsp-EsMiner* in May, Jun, Jul, and Nov datasets are relatively higher than the ones for other datasets. In our experiments, the ESG is resampled 50, 100, 150, and 250 times, and total running time of *Btsp-ESMiner* includes all the bootstrap trials. Whereas, for the threshold-based EsGrowth algorithm, the ESG is mined only once.

In summary, the total running times for *Btsp-ESMiner* (averaged over different bootstrap parameters) are approximately 40% more than EsGrowth. The running time required for the *Btsp-ESMiner* is primarily dependent on the resampling ratio and the number of trials. To increase the trustworthiness of the results, one can increase the number of trials and resampling ratio. In addition to that, the trustworthiness of the results can be traded off with the running time. Choosing a lower resampling ratio or number of trials would decrease the running time, as well as the trustworthiness of the results.

In Figure 6.11, we demonstrate the total number of STESs discovered from each dataset. An important observation we can make here is that our *Btsp-EsMiner* algorithm can find many more patterns when compared to the threshold-based approaches. In both *SequenceConnect* and EsGrowth, we find considerably small set of resulting STESs with our thresholds. However, setting the *right* thresholds for these algorithms are not viable

in most cases. Similarly, setting very low thresholds to comprehensively find all the STESs often times requires exponential running times.

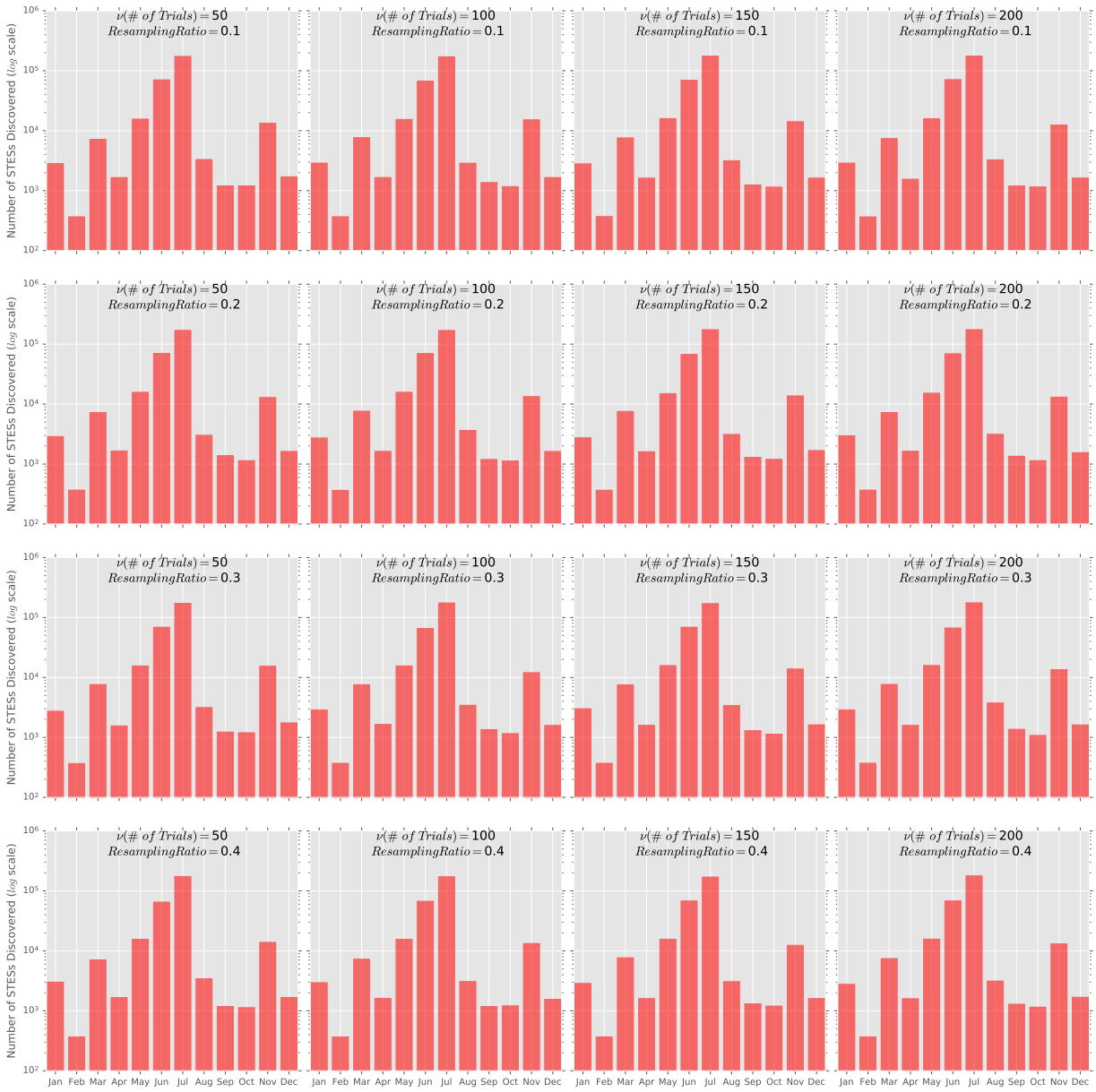


Figure 6.11: The total number of STESs discovered from our datasets with different bootstrap parameters in *Btsp-EsMiner* experiments

In Figure 6.12, we show the number of STESs discovered for different resampling ratios and bootstrap trials, where the STES counts are categorized based on the length of the STESs. For almost all datasets, the number of STESs discovered with different param-

eters is not affected by the changes in the number of bootstrap trials or resampling ratio parameters. Observing Figure 6.11 and Figure 6.12, we can conclude that the number of discovered STESs are not much affected by increasing or decreasing the resampling ratio or the number of bootstrap trials. While the number of trials or resampling ratio does not significantly impact the number, we can suggest that it affects the quality of the discovered results.

Next, we will discuss the relevance and quality of the mining results from *Btsp-ESMiner* algorithm. To make it concise, we will illustrate the distributions of resulting pi values of two set of *Btsp-EsMiner* trials for all the datasets: (1) when $rR = 0.1$ and $\nu = 50$ – in Figure 6.13 and (1) when $rR = 0.4$ and $\nu = 200$ – in Figure 6.14. We will report the top-15 most prevalent length-2 STESs in each dataset to keep it simple. The event sequences are sorted based on their average pi value from all the bootstrap trials. In addition to the results from bootstrap-based approach, we also demonstrate the mining results from EsGrowth algorithm. We show the resulting pi values of the STESs for different thresholds in EsGrowth experiments. The comprehensive results for longer length STESs can be found in our website [156].

From Figure 6.13 and Figure 6.14, we can see that the discovered top-15 STESs are consistent throughout all of our datasets. When we compare the STESs discovered from four datasets, we see that while their median values can differ, their confidence intervals generally overlap with one another. This shows that our results are consistent among the datasets. Another important observation we can make is the range of the confidence intervals. We often observe larger confidence intervals when there is an imbalance between the instance counts of event types (See STESs involving ar and fl together). The smaller confidence interval for an STES primarily suggests that the discovered pi value estimation is more robust within that dataset.

Another important observation we can make is the effect of edge resampling and bootstrap trials. When the resampling ratio is increased from $rR = 0.1$ to $rR = 0.4$, and

the number of bootstrap trials (ν) are also increased from 50 to 200, we observe similar STESs discovered from the same datasets. One observable difference here is the confidence intervals and the number of outliers in the distributions of the pi values for STESs. When we have less trials, we observe a larger confidence interval, and more outliers

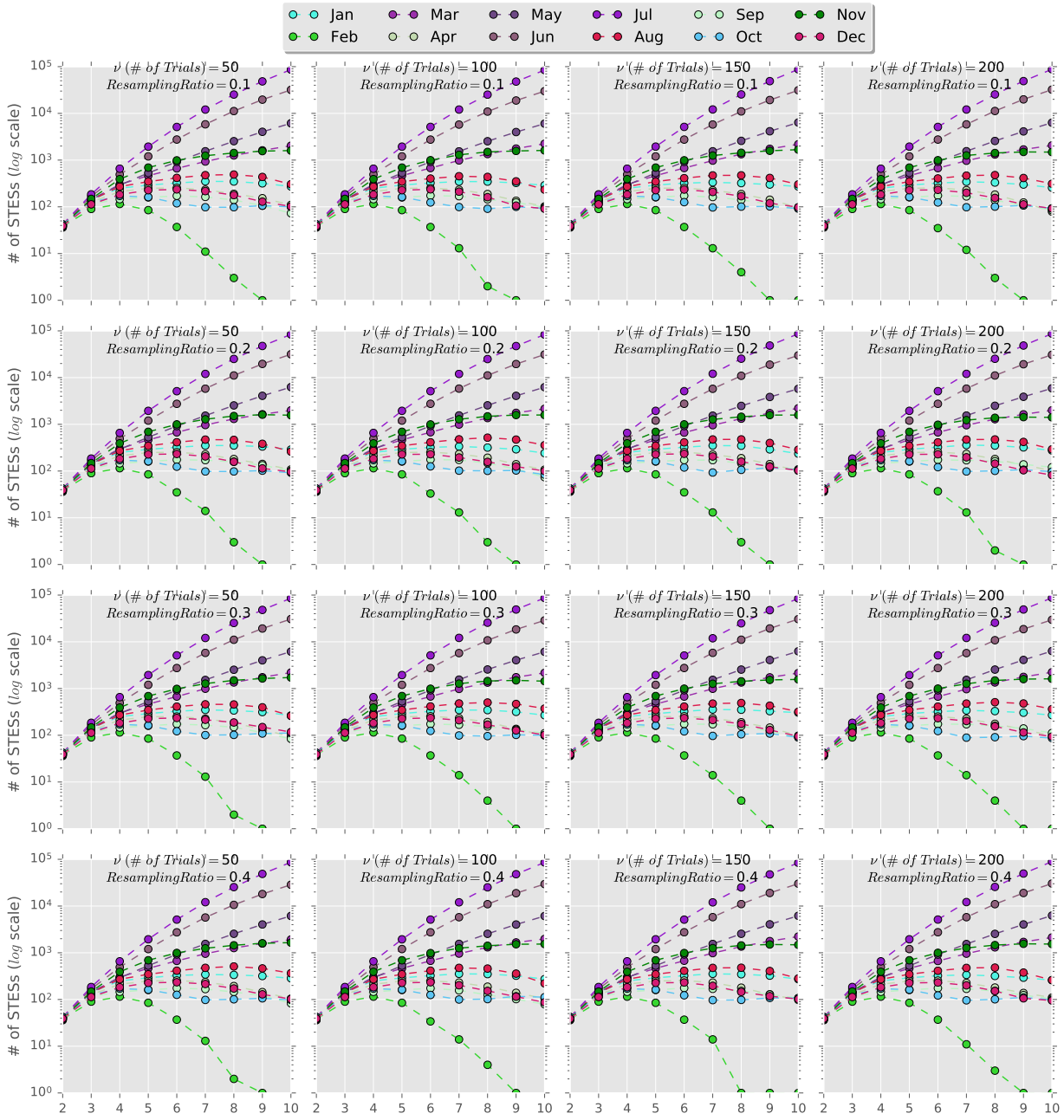


Figure 6.12: The number of STESs discovered with different bootstrap parameters in *Btsp-EsMiner* experiments

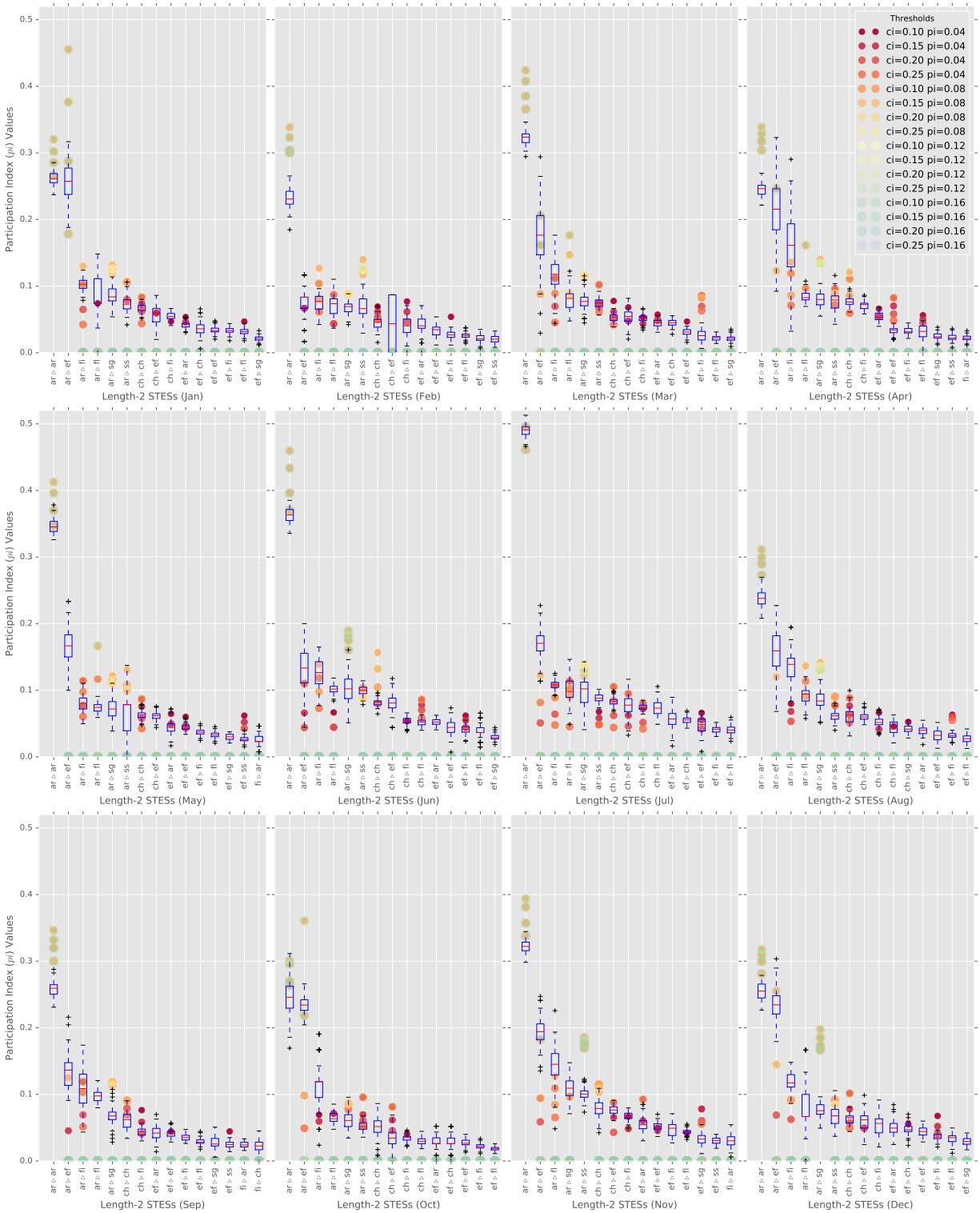


Figure 6.13: Length-2 STESs discovered from *Btsp-EsMiner* (with parameters $\mathrm{rR} = 0.1$ and $\nu = 50$) to the results from EsGrowth runs with different ci and pi thresholds

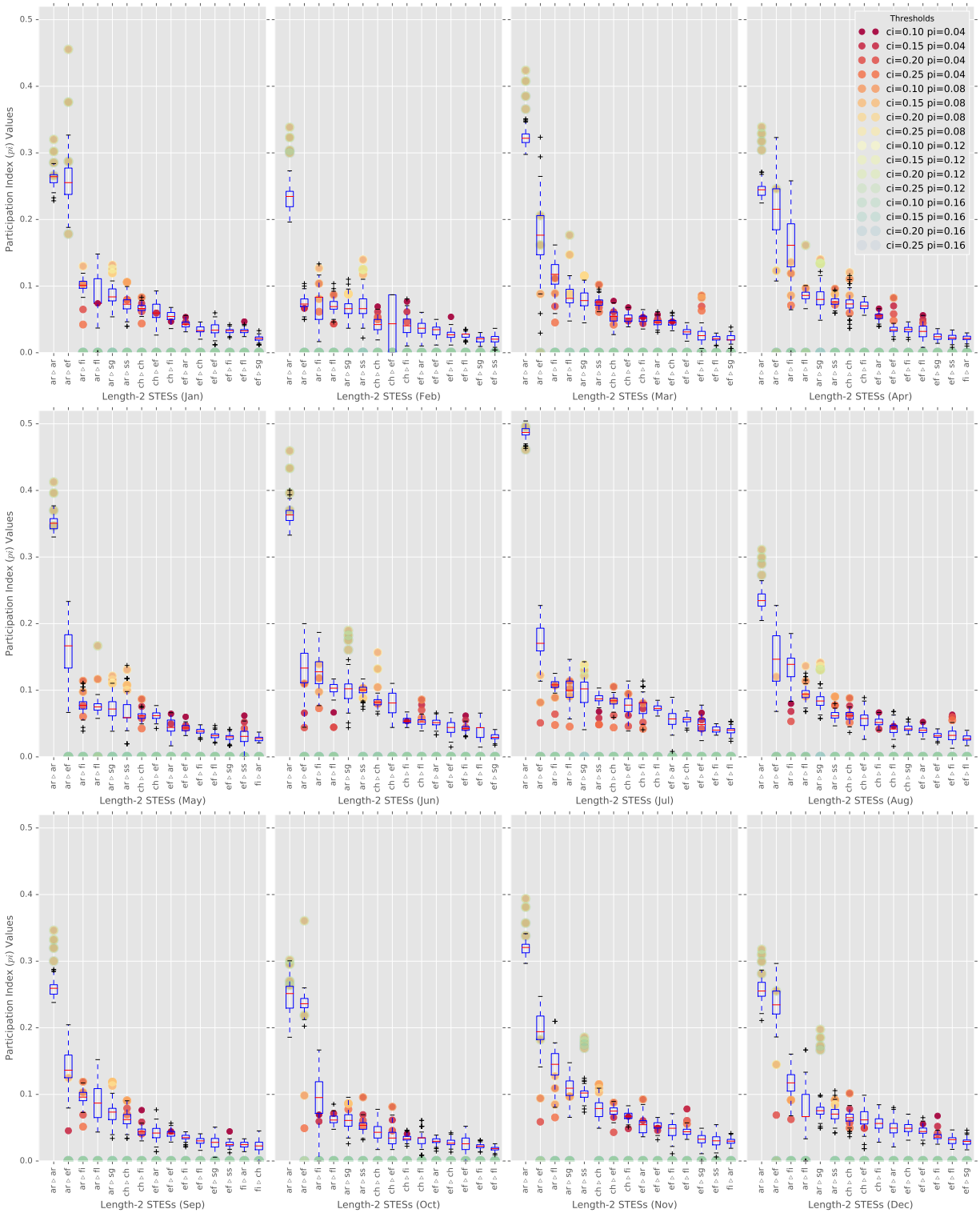


Figure 6.14: Length-2 STESs discovered from *Btsp-EsMiner* (with parameters $\mathtt{rR} = 0.4$ and $\nu = 200$) to the results from EsGrowth runs with different ci and pi thresholds

even though we have less runs and larger inter-quartile ranges. However, increasing the resampling ratio and the number of bootstrap trials, eliminates most of the outliers, and still decreases the confidence intervals. This situation is much more clear in $\mathtt{ar} \triangleright \mathtt{ef}$ and $\mathtt{ar} \triangleright \mathtt{fi}$ sequences.

Another aspect of our evaluation is the relevance comparison with threshold-based approach. One observation we can make is the variation of the $\mathtt{pi}$ values when using different $\mathtt{ci}_{\mathtt{th}}$ values in the threshold-based approach. The variation is two-fold: (1) The variation of the $\mathtt{pi}$ values for a particular STES and (2) the variation of the $\mathtt{pi}$ values across different STESs. The latter is much expected as the natural phenomena may or may not be spatiotemporally following each other. However, the former variation poses a challenge that is difficult to solve with trial and error. For example, for $(\mathtt{ar} \triangleright \mathtt{ar})$ sequences $\mathtt{ci}_{\mathtt{th}} = 0.25$ can be an accurate cut-off point (given the distributions from *Btsp-EsMiner*); however, if we set the $\mathtt{ci}_{\mathtt{th}}$ to 0.25 for the entire dataset, we miss all $(\mathtt{ar} \triangleright \mathtt{fl})$ sequences, as well as the sequences including the $(\mathtt{ar} \triangleright \mathtt{fl})$ subsequence. It is well-known to solar physics experts that flares can occur anywhere on the Sun's surface, from active regions ($\mathtt{ar}$) to the the boundaries of the magnetic network of the quiet Sun [135]. However, large area flares ($\mathtt{fl}$) have preferred locations. They originate from the large active regions showing a complex geometry of the 3D magnetic field [136]. Even for the simplistic cases of $(\mathtt{ar} \triangleright \mathtt{ar})$ and $(\mathtt{ar} \triangleright \mathtt{fl})$, creating user-defined thresholds is difficult, primarily because of the unbalanced nature of the natural phenomena. Therefore, we can suggest that mining a distribution of $\mathtt{pi}$ values using edge resampling from the sample $\mathtt{ESG}$ is a better approach for explorative analysis, because outputting a mere $\mathtt{pi}$ value based on set thresholds cannot properly represent the characteristics of the population.

7 CONCLUSION

7.1 Future Work

We plan to extend our research on spatiotemporal event sequence mining algorithms in several directions. In the following subsections, we listed these ideas into three categories.

7.1.1 Mixed Mining of Spatial, Temporal, and Image Data:

Initially, we want to include spatiotemporal attributes such as motion parameters (velocity, acceleration), shape parameters (shape evolution, areal evolution) for understanding the dynamics of the sequences better. We also intend to incorporate the available image-based raster data to aid in the spatiotemporal event sequence discovery, especially for understanding the unique characteristics of particular sub-classes (e.g., the change in the image parameters of X-class vs. M-class flares and their associations with discovered patterns). Another future direction is examining the other available data sources such as physical parameters of the solar events, which can help us understand the distinct characteristics of each event type and help us create better models in spatiotemporal frequent pattern mining.

7.1.2 Creating a Solar Event Search Engine

We are currently developing two web-based services for the solar physics community - a search tool for solar events and image parameters, ISD (Integrated Solar Database) [157] and a video-based visualization tool, SOLEV (Solar Event Video Generation Framework) [24]. We developed the back-end of the ISD. In the future, we would like to integrate

these two systems with the solar graph index (SOLGRIND) [158], and eventually create a solar event search engine that is capable of handling spatial, temporal, spatiotemporal, and textual queries. For textual queries, we intend to create semantic annotations using SOLGRIND, and utilize them for queries such as 'long filaments on the east limb' or 'flaring active regions close to two sigmoids'.

7.1.3 From Knowledge to Wisdom - Utilizing Patterns for Prediction

One particularly entertaining potential application area for spatiotemporal event sequence discovery is the prediction. We are interested in the prediction of solar events such as solar flares and CMEs that can create geomagnetic storms and drastically impact our world. For the task of prediction, we plan to utilize the spatiotemporal event sequences and spatiotemporal co-occurrence patterns.

7.2 Concluding Remarks

From various data sources including sensors, satellite imagery, GPS signals, business transactions, social networks, healthcare applications, and many others, we have seen an explosive growth in the volume of the data being generated. According to an IBM report in 2012, we are expected to create 2.5 quintillion bytes of data every day [159]. To answer the needs of the society and domain experts in their respective scientific fields, it is necessary to create automated knowledge discovery tools to generate interesting, useful, and actionable patterns. These patterns can be used for verification of currently known relationships, prediction of specific events, and even potential discovery of unknown relationships from the data.

Mining frequent patterns from spatiotemporal datasets has emerged in recent decades with a primary focus on understanding the implicit spatial and temporal relationships among instances and discovering useful and interesting underlying spatiotemporal pat-

terns. In this thesis, we have introduced the spatiotemporal event sequence mining from evolving region trajectories. The spatiotemporal event sequences is one type of spatiotemporal frequent patterns that has its roots in temporal event sequence mining and spatial co-location mining. Our goal in introducing spatiotemporal event sequences and creating a framework for discovery can be summarized with one sentence: *Understand the follow relationships between different groups of spatiotemporal event types and their instances.* By doing this, we aimed to help the researchers in their fields obtain an automated knowledge discovery framework, which is capable of showing which trajectories come after another and are located close-by.

Our research has achieved these goals in several ways. Firstly, we introduced a data model for spatiotemporal trajectories. Our models are conceptually designed and also implemented in many different computing environments including an object oriented design in Java [155], a relational database extension (in PostgreSQL with PostGIS) [160], and a columnar non-relational database extension [100]. Secondly, we introduced the groundwork for our spatiotemporal event sequence mining framework from the perspective of spatiotemporal co-occurrences, and introduced novel significance measures that is more relevant and more efficient for identifying the spatiotemporal follow relationships [134, 142]. Then, we designed novel models for the spatiotemporal follow relationships in the context of evolving region trajectories with the co-occurrence of head and tail window concepts [154], and extended our models with additional flexibility on head and tail window generation in [155]. Lastly, we have introduced algorithms for discovering spatiotemporal event sequences [154, 155, 161]. As a by-product, we also created an indexing technique, SOLGRIND, for indexing the spatiotemporal relationships among the solar event instances [158].

Comprehensively, our work, presented in this thesis, enables domain experts to conduct an in-depth study of solar event types, whose spatial and temporal extensions can be represented as evolving region trajectories. To our knowledge, none of the ex-

isting methods deal with the problem of mining spatiotemporal event sequences from datasets with evolving regions. We explored and evaluated trajectory data models, significance and prevalence measures, Apriori and pattern growth-based algorithms, as well as Top-*(R%, K)* and Bootstrap-based extensions (for mining without thresholds) of our spatiotemporal event sequence discovery algorithms.

BIBLIOGRAPHY

[1] LMSAL. Heliophysics Event Registry, March 2014. URL: http://lmsal.com/isolsearch, [Online; accessed 26-June-2014].

[2] Connie C Wong, Kevin E Loewke, Nancy L Bossert, Barry Behr, Christopher J De Jonge, Thomas M Baer, and Renee A Reijo Pera. Non-invasive imaging of human embryos before embryonic genome activation predicts development to the blastocyst stage. *Nature Biotechnology*, 28(10):1115–1121, 2010.

[3] Joe Conaghan, Alice A Chen, Susan P Willman, Kristen Ivani, Philip E Chenette, Robert Boostanfar, Valerie L Baker, G David Adamson, Mary E Abusief, Marina Gvakharia, et al. Improving embryo selection using a computer-automated time-lapse image analysis test plus day 3 morphology: results from a prospective multicenter trial. *Fertility and sterility*, 100(2):412–419, 2013.

[4] Henri E. Z. Tonnang, Richard Y. M. Kangalawe, and Pius Z. Yanda. Predicting and mapping malaria under climate change scenarios: the potential redistribution of malaria vectors in africa. *Malaria Journal*, 9:111 – 120, 2010.

[5] Krijn P. Paaijmans, Simon Blanford, Andrew S. Bell, Justine I. Blanford, Andrew F. Read, and Matthew B. Thomas. Influence of climate on malaria transmission depends on daily temperature variation. *Proceedings of the National Academy of Sciences*, 107(34):15135–15139, 2010.

[6] Katrin Kuhn, Diarmid Campbell-Lendrum, Andy Haines, and Jonathan Cox. Using climate to predict infectious disease epidemics. *World Health Organization, Geneva*, 2005.

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

[8] N Hurlburt, M Cheung, C Schrijver, L Chang, S Freeland, S Green, C Heck, A Jaffey, A Kobashi, D Schiff, et al. Heliophysics event knowledgebase for the solar dynamics observatory (sdo) and beyond. In *The Solar Dynamics Observatory*, pages 67–78. Springer, 2012.

[9] PCH Martens, GDR Attrill, AR Davey, A Engell, S Farid, PC Grigis, J Kasper, K Korreck, SH Saar, A Savcheva, et al. Computer vision for the solar dynamics observatory (SDO). In *The Solar Dynamics Observatory*, pages 79–113. Springer, 2012.

[10] Thorsten Wiegand, Kirk Moloney, and Suzanne Milton. Population dynamics, disturbance, and pattern evolution: Identifying the fundamental scales of organization in a model ecosystem. *The American Naturalist*, 152(3):321–337, 1998.

[11] Martin Hoerling and Arun Kumar. Atmospheric response patterns associated with tropical forcing. *Journal of Climate*, 15(16):2184–2203, 2002.

[12] Barbara Romanowicz. Spatiotemporal patterns in the energy release of great earthquakes. *Science*, 260(5116):1923–1926, 1993.

[13] Fahui Wang and William Minor. Where the jobs are: Employment access and crime patterns in cleveland. *Annals of the Association of American Geographers*, 92(3):435–450, 2002.

[14] Mohammed Javeed Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

[15] Guozhu Dong and Jian Pei. *Sequence Data Mining*, volume 33 of *Advances in Database Systems*. Kluwer, 2007.

[16] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. Mining access patterns efficiently from web logs. In *Knowledge Discovery and Data Mining, Current Issues and New Applications, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, April 18-20, 2000, Proceedings*, pages 396–407, 2000.

[17] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology - EDBT'96, 5th Interna-*

tional Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings, pages 3–17, 1996.

[18] Yanchang Zhao, Huaifeng Zhang, Longbing Cao, Hans Bohlscheid, Yuming Ou, and Chengqi Zhang. Data mining applications in social security. In *Data Mining for Business Applications*, pages 81–96. Springer, 2009.

[19] Martin Erwig. Toward spatio-temporal patterns. In Rita de Caluwe, Guy de Tré, and Gloria Bordogna, editors, *Spatio-Temporal Databases*, pages 29–53. Springer Berlin Heidelberg, 2004.

[20] Russell L Elsberry. Predicting hurricane landfall precipitation: Optimistic and pessimistic views from the symposium on precipitation extremes. *Bulletin of the American Meteorological Society*, 83(9):1333–1339, 2002.

[21] Sidney A. Gauthreaux and Carrol G. Belser and. Bird movements on Doppler weather surveillance radar. *Birding*, 35(6):616–628, 2003.

[22] Michael A. Schuh, Rafal A. Angryk, Karthik Ganesan Pillai, Juan M. Banda, and Petrus C. Martens. A large-scale solar image dataset with labeled event regions. In *IEEE International Conference on Image Processing, ICIP 2013, Melbourne, Australia, September 15-18, 2013*, pages 4349–4353, 2013.

[23] Dustin Kempton, Karthik Ganesan Pillai, and Rafal A. Angryk. Iterative refinement of multiple targets tracking of solar events. In *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*, pages 36–44, 2014.

[24] Soukaina Filali Boubrahimi, Berkay Aydin, Dustin Kempton, and Rafal A. Angryk. Spatio-temporal interpolation methods for solar events metadata. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 3149–3157, 2016.

[25] Stephanie R Langhoff and Tore Straume. Highlights of the "Space Weather Risks and Society" workshop. *Space Weather*, 10(6), 2012.

[26] Dianna Payne, Sean P Flaherty, Michael F Barry, and Colin D Matthews. Preliminary observations on polar body extrusion and pronuclear formation in human oocytes using time-lapse video cinematography. *Human Reproduction*, 12(3):532–541, 1997.

[27] Jacques de Mouzon, V Goossens, Siladitya Bhattacharya, JA Castilla, AP Ferraretti, V Korsak, Markus Kupka, Karl-Gosta Nygren, A Nyboe Andersen, et al. Assisted reproductive technology in europe, 2006: results generated from european registers by eshre. *Human Reproduction*, page deq124, 2010.

[28] Mark C. Walker, Kellie E. Murphy, Saiyi Pan, Qiuying Yang, and Shi Wu Wen. Adverse maternal outcomes in multifetal pregnancies. *BJOG: An International Journal of Obstetrics Gynaecology*, 111(11):1294–1296, 2004.

[29] Javier Herrero and Marcos Meseguer. Selection of high potential embryos using time-lapse imaging: the era of morphokinetics. *Fertility and sterility*, 99(4):1030–1034, 2013.

[30] Marcos Meseguer, Javier Herrero, Alberto Tejera, Karen Marie Hilligsøe, Niels Birger Ramsing, and Jose Remohí. The use of morphokinetics as a predictor of embryo implantation. *Human reproduction*, 26(10):2658–2671, 2011.

[31] Rachel Lowe, Trevor C Bailey, David B Stephenson, Richard J Graham, Caio AS Coelho, Marilia Sa Carvalho, and Christovam Barcellos. Spatio-temporal modelling of climate-sensitive disease risk: Towards an early warning system for dengue in brazil. *Computers & Geosciences*, 37(3):371–381, 2011.

[32] PS Mellor and CJ Leake. Climatic and geographic influences on arboviral infections and vectors. *Revue scientifique et technique (International Office of Epizootics)*, 19(1):41–54, 2000.

[33] David J Rogers, Sarah E Randolph, Robert W Snow, and Simon I Hay. Satellite imagery in the study and forecast of malaria. *Nature*, 415(6872):710–715, 2002.

[34] Shashi Shekhar, Zhe Jiang, Reem Y Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: A computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[35] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

[36] Karthik Ganesan Pillai, Rafal A. Angryk, Juan M. Banda, Michael A. Schuh, and Tim Wylie. Spatio-temporal co-occurrence pattern mining in data sets with evolving regions. In *12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012*, pages 805–812, 2012.

[37] Karthik Ganesan Pillai, Rafal A. Angryk, and Berkay Aydin. A filter-and-refine approach to mine spatiotemporal co-occurrences. In *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, pages 104–113, 2013.

[38] Berkay Aydin, Dustin Kempton, Vijay Akkineni, Shaktidhar Reddy Gopavaram, Karthik Ganesan Pillai, and Rafal A. Angryk. Spatiotemporal indexing techniques for efficiently mining spatiotemporal co-occurrence patterns. In *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*, pages 1–10, 2014.

[39] Berkay Aydin, Dustin Kempton, Vijay Akkineni, Rafal Angryk, and Karthik Ganesan Pillai. Mining spatiotemporal co-occurrence patterns in solar datasets. *Astronomy and Computing*, 13:136 – 144, 2015.

[40] Berkay Aydin, Vijay Akkineni, and Rafal A. Angryk. Mining spatiotemporal co-occurrence patterns in non-relational databases. *GeoInformatica*, 20(4):801–828, 2016.

[41] Shah Muhammad Hamdi, Berkay Aydin, and Rafal Angryk. A pattern growth-based approach for mining spatiotemporal cooccurrence patterns. In *IEEE International Conference on Data Mining Workshop, ICDMW 2016, Barcelona, Spain, December 12-15, 2016*, 2016.

[42] Karthik Ganesan Pillai, Rafal A. Angryk, Juan M. Banda, Dustin Kempton, Berkay Aydin, and Petrus C. Martens. Mining at most top-k% spatiotemporal co-occurrence patterns in datasets with extended spatial representations. *ACM Trans. Spatial Algorithms and Systems*, 2(3):10:1–10:27, 2016.

[43] Tamas Abraham and John F. Roddick. Opportunities for knowledge discovery in spatio-temporal information systems. *Australasian J. of Inf. Systems*, 5(2), 1998.

[44] John F. Roddick and Brian G. Lees. Spatiotemporal data mining paradigms and methodologies. In Harvey Miller and Jiawei Han, editors, *Geographic Data Mining and Knowledge*. CRC Press, 2001.

[45] Daniel Walgraef. *Spatio-Temporal Pattern Formation: With Examples from Physics, Chemistry, and Materials Science*. Springer Verlag, 1997.

[46] Shashi Shekhar and Hui Xiong, editors. *Encyclopedia of GIS*. Springer, 2008.

[47] Yong Ge, Hui Xiong, Chuanren Liu, and Zhi-Hua Zhou. A taxi driving fraud detection system. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 181–190, 2011.

[48] James M. Kang, Shashi Shekhar, Michael Henjum, Paige J. Novak, and William A. Arnold. Discovering teleconnected flow anomalies: A relationship analysis of dynamic neighborhoods (RAD) approach. In *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings*, pages 44–61, 2009.

[49] Florian Verhein and Sanjay Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In *Database Systems for Advanced Applications, 11th International Conference, DASFAA 2006, Singapore, April 12-15, 2006, Proceedings*, pages 187–201, 2006.

[50] Yan Huang, Liqin Zhang, and Pusheng Zhang. A framework for mining sequential patterns from spatio-temporal event data sets. *IEEE Trans. Knowl. Data Eng.*, 20(4):433–448, 2008.

[51] Yijun Lu. *Concept hierarchy in data mining: Specification, generation and implementation*. PhD thesis, Simon Fraser University, 1997.

[52] Michael R. Evans, Dev Oliver, Shashi Shekhar, and Francis Harvey. Summarizing trajectories into *k*-primary corridors: a summary of results. In *SIGSPATIAL 2012 International*

Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012, pages 454–457, 2012.

[53] Noel A. C. Cressie. *Statistics for spatial data*. Wiley series in probability and mathematical statistics. J. Wiley & Sons, 1993.

[54] Slava Kisilevich, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo. Spatio-temporal clustering. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 855–874. Springer, 2010.

[55] Derya Birant and Alp Kut. ST-DBSCAN: an algorithm for clustering spatial-temporal data. *Data Knowl. Eng.*, 60(1):208–221, 2007.

[56] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.

[57] Toshiro Tango, Kunihiko Takahashi, and Kazuaki Kohriyama. A space–time scan statistic for detecting emerging outbreaks. *Biometrics*, 67(1):106–115, 2011.

[58] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008.

[59] Jungho Im, John R. Jensen, and Jason A. Tullis. Development of a remote sensing change detection system based on neighborhood correlation image analysis and intelligent knowledge-based systems. In *IEEE International Geoscience & Remote Sensing Symposium, IGARSS 2005, July 25-29, 2005, Seoul, Korea, Proceedings*, pages 2129–2132, 2005.

[60] Martin Kulldorff. Prospective time periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 164(1):61–72, 2001.

[61] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004.

[62] Panagiotis Papapetrou, George Kollios, Stan Sclaroff, and Dimitrios Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 354–361, 2005.

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

[64] Robert Moskovitch and Yuval Shahar. Fast time intervals mining using the transitivity of temporal relations. *Knowl. Inf. Syst.*, 42(1):21–48, 2015.

[65] Shashi Shekhar and Sanjay Chawla. *Spatial databases - A tour*. Prentice Hall, 2003.

[66] Shashi Shekhar and Yan Huang. Discovering spatial co-location patterns: A summary of results. In *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings*, pages 236–256, 2001.

[67] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[68] Jin Soung Yoo and Shashi Shekhar. A partial join approach for mining co-location patterns. In *12th ACM International Workshop on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA, Proceedings*, pages 241–249, 2004.

[69] Jin Soung Yoo and Shashi Shekhar. A joinless approach for mining spatial colocation patterns. *IEEE Trans. Knowl. Data Eng.*, 18(10):1323–1337, 2006.

[70] Sajib Barua and Jörg Sander. Mining statistically significant co-location and segregation patterns. *IEEE Trans. Knowl. Data Eng.*, 26(5):1185–1199, 2014.

[71] Mete Celik, Shashi Shekhar, James P. Rogers, and James A. Shine. Mixed-drove spatiotemporal co-occurrence pattern mining. *IEEE Trans. Knowl. Data Eng.*, 20(10):1322–1335, 2008.

[72] Mete Celik, Shashi Shekhar, James P. Rogers, and James A. Shine. Sustained emerging spatio-temporal co-occurrence pattern mining: A summary of results. In *18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006), 13-15 November 2006, Washington, DC, USA*, pages 106–115, 2006.

[73] Mete Celik. Discovering partial spatio-temporal co-occurrence patterns. In *IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services, ICSDM 2011, Fuzhou, China, June 29 - July 1, 2011*, pages 116–120, 2011.

[74] Mete Celik, Nuh Azginoglu, and Ramazan Terzi. Mining periodic spatio-temporal co-occurrence patterns: A summary of results. In *Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on*, pages 1–5, July 2012.

[75] Feng Qian, Qinming He, and Jiangfeng He. Mining spread patterns of spatio-temporal co-occurrences over zones. In *Computational Science and Its Applications - ICCSA 2009, International Conference, Seoul, Korea, June 29-July 2, 2009, Proceedings, Part II*, pages 677–692, 2009.

[76] Zhongnan Zhang and Weili Wu. Composite spatio-temporal co-occurrence pattern mining. In *Wireless Algorithms, Systems, and Applications, Third International Conference, WASA 2008, Dallas, TX, USA, October 26-28, 2008. Proceedings*, pages 454–465, 2008.

[77] Huiping Cao, Nikos Mamoulis, and David W. Cheung. Mining frequent spatio-temporal sequential patterns. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 82–89, 2005.

[78] Florian Verhein. Mining complex spatio-temporal sequence patterns. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 605–616, 2009.

[79] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 330–339, 2007.

[80] Pradeep Mohan, Shashi Shekhar, James A. Shine, and James P. Rogers. Cascading spatio-temporal pattern discovery. *IEEE Trans. Knowl. Data Eng.*, 24(11):1977–1992, 2012.

[81] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas F. La Porta. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *PVLDB*, 7(9):769–780, 2014.

[82] Hugo Alatrista Salas, Sandra Bringay, Frédéric Flouvat, Nazha Selmaoui-Folcher, and Maguelonne Teisseire. The pattern next door: Towards spatio-sequential pattern discovery. In *Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conf., PAKDD 2012, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Proc., Part II*, pages 157–168, 2012.

[83] A Savtchenko, D Ouzounov, S Ahmad, J Acker, G Leptoukh, J Koziana, and D Nickless. Terra and aqua modis products available from nasa ges daac. *Advances in Space Research*, 34(4):710–714, 2004.

[84] NOAA. NOAA GOES Geostationery Satellite Server, January 2016. URL: http://www.goes.noaa.gov/, [Online; accessed 29-March-2017].

[85] W Dean Pesnell, BJ Thompson, and PC Chamberlin. The solar dynamics observatory (sdo). In *The Solar Dynamics Observatory*, pages 3–15. Springer, 2011.

[86] Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo, and Jon Crowcroft. Recommending social events from mobile phone location data. In *ICDM*, pages 971–976, 2010.

[87] Maike Buchin, Somayeh Dodge, and Bettina Speckmann. Context-aware similarity of trajectories. In *International Conference on Geographic Information Science*, pages 43–56. Springer, 2012.

[88] Junhong Wang, Kate Young, Terry Hock, Dean Lauritsen, Dalton Behringer, Michael Black, Peter G Black, James Franklin, Jeff Halverson, John Molinari, et al. A long-term, high-

quality, high-vertical-resolution gps dropsonde dataset for hurricane and other studies. *Bulletin of the American Meteorological Society*, 96(6):961–973, 2015.

[89] Ling Chen, Mingqi Lv, and Gencai Chen. A system for destination and future route prediction based on trajectory mining. *Pervasive and Mobile Computing*, 6(6):657–676, 2010.

[90] Heng Ma, Tsueng-Fang Tsai, and Chia-Cheng Liu. Real-time monitoring of water quality using temporal trajectory of live fish. *Expert Syst. Appl.*, 37(7):5158–5171, 2010.

[91] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, José Antônio Fernandes de Macêdo, Fábio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data Knowl. Eng.*, 65(1):126–146, 2008.

[92] Eleftherios Tiakas, Apostolos Papadopoulos, Alexandros Nanopoulos, Yannis Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *Journal of Systems and Software*, 82(5):772–788, 2009.

[93] Irene Ntoutsi, Nikos Mitsou, and Gerasimos Marketos. Traffic mining in a road-network: How does the traffic flow? *IJBIDM*, 3(1):82–98, 2008.

[94] José Antonio Cotelo Lema, Luca Forlizzi, Ralf Hartmut Güting, Enrico Nardelli, and Markus Schneider. Algorithms for moving objects databases. *Comput. J.*, 46(6):680–712, 2003.

[95] Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.

[96] Cédric du Mouza and Philippe Rigaux. Mobility patterns. *GeoInformatica*, 9(4):297–319, 2005.

[97] Gerasimos Marketos and Yannis Theodoridis. Mobility data warehousing and mining. In *Proceedings of the VLDB 2009 PhD Workshop. Co-located with the 35th International Conference on Very Large Data Bases (VLDB 2009). Lyon, France, August 24, 2009*, 2009.

[98] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady L. Andrienko, Natalia V. Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, José Antônio Fernandes de Macêdo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32, 2013.

[99] Ralf Hartmut Güting, Fabio Valdés, and Maria Luisa Damiani. Symbolic trajectories. *ACM Trans. Spatial Algorithms and Systems*, 1(2):7:1–7:51, 2015.

[100] Berkay Aydin, Vijay Akkineni, and Rafal Angryk. Modeling and indexing spatiotemporal trajectory data in non-relational databases. In *Managing Big Data in Cloud Computing Environments*, pages 133–162. IGI Global, 2016.

[101] Michael Steinbach and Vipin Kumar. Generalizing the notion of confidence. *Knowl. Inf. Syst.*, 12(3):279–299, 2007.

[102] Roberto J. Bayardo Jr. and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*, pages 145–154, 1999.

[103] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), 2006.

[104] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 32–41, 2002.

[105] Kenneth McGarry. A survey of interestingness measures for knowledge discovery. *Knowledge Eng. Review*, 20(1):39–61, 2005.

[106] Zeng-Jie Yang and Robert J. Wechsler-Reya. Hit 'em where they live: Targeting the cancer stem cell niche. *Cancer Cell*, 11(1):3 – 5, 2007.

[107] Barry D Keim, Robert A Muller, and Gregory W Stone. Spatiotemporal patterns and return periods of tropical storm and hurricane strikes from texas to maine. *Journal of Climate*, 20(14):3498–3509, 2007.

[108] Krzysztof Koperski and Jiawei Han. Discovery of spatial association rules in geographic information databases. In *Advances in Spatial Databases, 4th International Symposium, SSD'95, Portland, Maine, USA, August 6-9, 1995, Proceedings*, pages 47–66, 1995.

[109] Yasuhiko Morimoto. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 353–358, 2001.

[110] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering colocation patterns from spatial data sets: A general approach. *IEEE Trans. Knowl. Data Eng.*, 16(12):1472–1485, 2004.

[111] Hui Xiong, Shashi Shekhar, Yan Huang, Vipin Kumar, Xiaobin Ma, and Jin Soung Yoo. A framework for discovering co-location patterns in data sets with extended spatial objects. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*, pages 78–89, 2004.

[112] Yan Huang, Jian Pei, and Hui Xiong. Mining co-location patterns with rare events from spatial data sets. *GeoInformatica*, 10(3):239–260, 2006.

[113] Jin Soung Yoo, Shashi Shekhar, Sangho Kim, and Mete Celik. Discovery of co-evolving spatial event sets. In *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 306–315, 2006.

[114] Huiping Cao, Nikos Mamoulis, and David W. Cheung. Discovery of collocation episodes in spatiotemporal data. In *Proc. of the 6th IEEE Int. Conf. on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 823–827, 2006.

[115] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups of trackable moving point objects. In *Geographic Information Science, Second International Conference, GI-Science 2002, Boulder, CO, USA, September 25-28, 2002, Proceedings*, pages 132–144, 2002.

[116] Joachim Gudmundsson, Marc J. van Kreveld, and Bettina Speckmann. Efficient detection of motion patterns in spatio-temporal data sets. In *12th ACM International Workshop on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA, Proceedings*, pages 250–257, 2004.

[117] Patrick Laube, Marc van Kreveld, and Stephan Imfeld. *Finding REMO — Detecting Relative Motion Patterns in Geospatial Lifelines*, pages 201–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[118] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *Advances in Spatial and Temporal Databases, 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005, Proceedings*, pages 364–381, 2005.

[119] Mete Celik, Shashi Shekhar, James P. Rogers, James A. Shine, and Jin Soung Yoo. Mixed-drove spatio-temporal co-occurence pattern mining: A summary of results. In *Proc. of the 6th IEEE Int. Conf. on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 119–128, 2006.

[120] Mete Celik. Partial spatio-temporal co-occurrence pattern mining. *Knowledge and Information Systems*, pages 1–23, 2014.

[121] Karthik Ganesan Pillai, Rafal A Angryk, Juan M Banda, Tim Wylie, and Michael A Schuh. Spatiotemporal co-occurrence rules. In *New Trends in Databases and Information Systems*, pages 27–35. Springer, 2014.

[122] Berkay Aydin and Rafal A. Angryk. Spatiotemporal frequent pattern mining on solar data: Current algorithms and future directions. In *IEEE International Conference on Data Mining Workshop, ICDMW 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 575–581, 2015.

[123] 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, and R. Timmons. Heliophysics event knowledgebase for the solar dynamics observatory (sdo) and beyond. *Solar Physics*, 275(1):67–78, 2010.

[124] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 207–216, 1993.

[125] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[126] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[127] M. Shahriar Hossain and Rafal A. Angryk. Gdclust: A graph-based document clustering technique. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 417–422, 2007.

[128] Peter John Taylor. *Quantitative methods in geography : an introduction to spatial analysis / Peter J. Taylor.* Boston : Houghton Mifflin, c1977., 1977.

[129] Elena Deza and Michel Deza. *Dictionary of distances*. North-Holland, 2006.

[130] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proc. of 20th Int. Conf. on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[131] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 1–12, 2000.

[132] Berkay Aydin, Rafal A. Angryk, and Karthik Ganesan Pillai. ERMO-DG: Evolving region moving object dataset generator. In *Proc. of the 27th FLAIRS Int. Conf.*, pages 321–326. AAAI Press, 2014.

[133] DJ Kempton and RA Angryk. Tracking solar events through iterative refinement. *Astronomy and Computing*, 13:124–135, 2015.

[134] Berkay Aydin, Ahmet Kucuk, and Rafal A. Angryk. Measuring the significance of spatiotemporal co-occurrences. *ACM Trans. Spatial Algorithms and Systems*, page Under Review.

[135] S. Krucker, A. O. Benz, T. S. Bastian, and L. W. Acton. X-Ray Network Flares of the Quiet Sun. *The Astrophysical Journal*, 488:499–505, October 1997.

[136] Stéphane Régnier and Richard C Canfield. Evolution of magnetic fields and energetics of flares in active region 8210. *Astronomy & Astrophysics*, 451(1):319–330, 2006.

[137] MJ Murray, L van Driel-Gesztelyi, and D Baker. Simulations of emerging flux in a coronal hole: oscillatory reconnection. *Astronomy & Astrophysics*, 494(1):329–337, 2009.

[138] E. B. Mayfield and J. K. Lawrence. The correlation of solar flare production with magnetic energy in active regions. *Solar Physics*, 96:293–305, April 1985.

[139] J Heyvaerts, ER Priest, and DM Rust. An emerging flux model for the solar flare phenomenon. *The Astrophysical Journal*, 216:123–137, 1977.

[140] J Feynman and SF Martin. The initiation of coronal mass ejections by newly emerging magnetic flux. *Journal of Geophysical Research: Space Physics*, 100(A3):3355–3367, 1995.

[141] Nariaki V Nitta and Hugh S Hudson. Recurrent flare/cme events from an emerging flux region. *Geophysical research letters*, 28(19):3801–3804, 2001.

[142] Berkay Aydin, Vijay Akkineni, and Rafal Angryk. Time-efficient significance measure for discovering spatiotemporal co-occurrences from data with unbalanced characteristics. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '15, pages 80:1–80:4, New York, NY, USA, 2015. ACM.

[143] Fabian Mörchen. Algorithms for time series knowledge mining. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 668–673, 2006.

[144] Petre Tzvetkov, Xifeng Yan, and Jiawei Han. TSP: mining top-*k* closed sequential patterns. *Knowl. Inf. Syst.*, 7(4):438–457, 2005.

[145] Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. Mining *top-k* frequent patterns in the presence of the memory constraint. *VLDB J.*, 17(5):1321–1344, 2008.

[146] Geoffrey I. Webb. Filtered-top-*k* association discovery. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 1(3):183–192, 2011.

[147] Vincent S. Tseng, Cheng-Wei Wu, Philippe Fournier-Viger, and Philip S. Yu. Efficient algorithms for mining top-k high utility itemsets. *IEEE Trans. Knowl. Data Eng.*, 28(1):54–67, 2016.

[148] Mete Celik, Shashi Shekhar, James P. Rogers, James A. Shine, and James M. Kang. Mining at most top-k% mixed-drove spatio-temporal co-occurrence patterns: A summary of results. In *Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007, 15-20 April 2007, Istanbul, Turkey*, pages 565–574, 2007.

[149] Bradley Efron. Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods. *Biometrika*, 68(3):589–599, 1981.

[150] MA Schuh, RA Angryk, and PC Martens. Solar image parameter data from the sdo: Long-term curation and data mining. *Astronomy and Computing*, 13:86–98, 2015.

[151] Heliophysics event knowledgebase. http://www.lmsal.com/hek/. Last Accessed: 21 March 2015.

[152] Shashi Shekhar and Hui Xiong. Java topology suite (jts). In *Encyclopedia of GIS*, pages 601–601. Springer, 2008.

[153] B Naveh. Jgrapht, 2017. URL: http://jgrapht.org/, [Online].

[154] Berkay Aydin and Rafal Angryk. Spatiotemporal event sequence mining from evolving regions. In *23rd International Conference on Pattern Recognition (ICPR), Cancún, México, December 4-8, 2016*, pages 4167–4172, 2016.

[155] Berkay Aydin and Rafal Angryk. Discovering spatiotemporal event sequences. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 46–55. ACM, 2016.

[156] Berkay Aydin. Dissertation Thesis: Discovery of Spatiotemporal Event Sequences, April 2017. URL: http://cs.gsu.edu/~baydin2/thesis/.

[157] Dmlab. Integrated Solar Event Database, December 2016. URL: http://isd.dmlab.cs.gsu.edu/, [Online; accessed 26-March-2017].

[158] Berkay Aydin, Ahmet Kucuk, and Rafal A. Angryk. Indexing spatiotemporal relations in solar event datasets. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 3140–3148, 2016.

[159] IBM. What is big data?, 2012. URL: https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html, [Online; accessed 29-Mar-2017].

[160] Ahmet Kucuk, Shah Muhammad Hamdi, Berkay Aydin, Michael A. Schuh, and Rafal A. Angryk. Pg-Trajectory: A PostgreSQL/PostGIS based data model for spatiotemporal trajectories. In *2016 IEEE Sixth International Conference on Big Data and Cloud Computing, BDCloud 2016, Atlanta, GA, October 8-10, 2016*, pages 81–88, 2016.

[161] Berkay Aydin and Rafal A. Angryk. A graph-based approach to spatiotemporal event sequence mining. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.*, pages 1090–1097, 2016.