ABSTRACT

Spatiotemporal co-occurrence patterns (STCOPs) represent the subsets of event types that occur together in both space and time. However, the discovery of STCOPs in data sets with extended spatial representations that evolve over time is computationally expensive because of the necessity to calculate interest measures to assess the co-occurrence strength, and the number of candidates for STCOPs growing exponentially with the number of spatiotemporal event types. In this paper, we introduce a novel and effective filter-and-refine algorithm to efficiently find prevalent STCOPs in massive spatiotemporal data repositories with polygon shapes that move and evolve over time. We provide theoretical analysis of our approach, and follow this investigation with a practical evaluation of our algorithm effectiveness on three real-life data sets and one artificial data set.

Categories and Subject Descriptors
H.4 [Spatio-Temporal Data Analysis]: Spatial Data Mining and Knowledge Discovery; D.2.8 [Geographic Information Retrieval]: Spatial Data Structures and Algorithms

General Terms
Spatiotemporal Data Mining

Keywords
evolving spatiotemporal events; extended spatial representations; spatiotemporal co-occurrence patterns;

1 Introduction

The growth of data volumes in nearly all scientific disciplines, business sectors and federal agencies is reaching historic proportions [3]. The data acquisition speed is becoming progressively faster and leading to uncontrollably growing data. In astronomy in particular, rapid advances in three technology areas (telescopes, detectors and computation) have caused the generation of massive data [2]. With the launch of NASA’s Solar Dynamics Observatory (SDO) mission, solar physics researchers started dealing with big data. SDO’s space telescope registers approximately 70,000 high resolution (4096 x 4096 pixels) images daily, obtaining one image every ten seconds [16]. It sends out 0.55 petabytes of raster data to Earth each year [16]. This trend in solar data is anticipated to be sustained by ground-based Advanced Technology Solar Telescope, which is expected to capture one million images per day and generate three to five PB of data per year [13]. To facilitate the important needs of Space Weather monitoring (which can have vital impacts on space and air travel, power grids, GPS and communication devices), many software modules working continuously on massive SDO raster data and generating object data with spatiotemporal characteristics. One application area for our research is solar events, since spatiotemporal co-occurrence patterns (STCOPs) frequently occur among various solar events. Given a spatiotemporal (abbreviated as ST in the rest of the paper) database in which data objects are represented as polygons which continuously change their movement, shape, and size, our goal is to identify STCOPs representing the subset of different event types that occur together in space and time.

1.1 A Real-life Example

STCOPs frequently occur among various solar events. Fig. 1 shows four types of solar events, Active Regions (AR), Filaments (FL), Sigmoids (SG), and Sunspots (SS) in spatial and temporal contexts with their corresponding shapes and bounding boxes. As seen in Fig. 1, the shapes of the solar events are represented as extended spatial representations (polygons). Moreover, the shape, size, and location of
the solar events continuously evolve over time. All of these factors influence relationships between various solar events, which lead to complex spatial and temporal interactions. Identifying STCOPs on the Sun could help us better understand the relationships between solar events and lead to better modeling and forecasting of important events such as coronal mass ejections and solar flares, which impact radiation in space, and can reduce the safety of space and air travel [11], disrupt intercontinental communication [11] and GPS [11], and even damage power grids [11].

1.2 Contributions

This paper presents a novel approach to our recent work initiated in [17], where we introduced the STCOPs mining problem, developed a general framework to discover STCOPs, proposed a naïve Apriori-based [1] STCOPs mining algorithm, and evaluated it using a real-life data set. This work makes the following new contributions: (1) We introduce a new and computationally efficient STCOPs mining algorithm (FastSTCOPs-Miner), using a filter-and-refine strategy to prune irrelevant STCOPs based on the usage of all-confidence (OMAX) measure as filtering mechanism for Jaccard-based analysis which is the standard measure in data mining [17, 21, 20]. (2) We provide a theoretical analysis to show the correctness and completeness of our FastSTCOPs-Miner algorithm. (3) We experimentally verify the correctness of proposed algorithm with our naïve STCOPs algorithm [17] on three real-life data sets and one artificial data set, and provide extended experimental results demonstrating the computational and memory efficiency of the FastSTCOPs-Miner algorithm.

The following issues are beyond the scope of this paper: (1) Determining threshold values to capture the strength of ST neighborhood relation between the instances of different event types; (2) ST data indexing strategies related to mining these events; and (3) Comparison with non-Apriori approaches. The reason we chose to start our work with Apriori-based technique is because we believe that our filter-and-refine strategy is easier to explain in these simpler settings. We hope to be able to incorporate the filter step in our future works on non-Apriori approaches; however, currently there is no non-Apriori method that is capable of taking advantage of our novel OMAX-based approach to filter patterns.

The rest of the paper is organized as follows: Sec. 2 gives background information on related work. We review important concepts of modeling STCOPs for evolving extended spatial representations and provide theoretical analysis of our approach in Sec. 3. In Sec. 4, we present our proposed FastSTCOPs-Miner algorithm. Finally, we present a variety of experiments demonstrating the effectiveness of our approach with real-life and artificial data sets, concluding with a summary of results and future work.

2 Related Work

Since ST data mining is an important area, many algorithms have been proposed in literature for co-location mining in ST databases: topological pattern mining [22], co-location episodes [4], mixed drove co-occurrence mining [5], spatial co-location pattern mining from extended spatial representations [23], ST pattern mining in scientific data [24], and interval orientation patterns [15]. However, none of these approaches focus on mining ST co-occurrences from data represented as polygons evolving in time.

Mining topological patterns, also called co-location patterns, from ST databases was introduced by Wang et al. in [22]. The authors introduced a summary-structure to record the number of instances of a feature in a region for a given time window. The authors used the summary-structure to approximate the instance counts of a co-location pattern. They also introduced the TopologyMiner algorithm to discover frequent co-location patterns in a depth-first manner.

Cao et al. introduced the problem of mining co-location episodes in ST data [4]. The authors define a co-location episode as a sequence of co-location patterns with some common feature type across consecutive time slots. The authors also introduced a two-step framework for mining co-location episodes. In the first step of the framework, the authors transform the original trajectories of moving objects to a sequence of close features to the corresponding object. Eventually, the object pairs of different feature types \((f_i, f_j)\) with close concurrent subsequences are identified. In the next step of the framework, an Apriori-based [1] technique is used to discover the frequent episodes, using the transformed sequence of feature sets.

Celik et al. introduced the problem of mining mixed-drove ST co-occurrence patterns (MDCOPs) in ST data [5]. An MDCOP is defined as a subset of ST mixed feature types whose instances are neighbors in space and time. They introduced the MDCOP-Miner algorithm, which extends the standard spatial co-location mining algorithm [9] to include time information. The algorithm first discovers all \(size-(k)\) spatially prevalent MDCOPs, and applies a time-prevalence based filtering to discover MDCOPs. Finally, the MDCOP-Miner algorithm generates \(size-(k+1)\) candidate MDCOPs using \(size-(k)\) MDCOPs.

Xiong et al. introduced the problem of mining spatial co-location patterns from extended spatial representations in [23]. The authors introduced a buffer based model to find co-location patterns. In the buffer based model, the neighborhood of an extended spatial representation is defined by the spatial buffer operation. The Euclidean neighborhood \(N(f)\) of some feature \(f\) is defined as the union of neighborhoods for every instance \(i\) of the feature \(f\). The Euclidean neighborhood \(N(f_1, \ldots, f_k)\) for a feature set \(E = \{f_1, \ldots, f_k\}\) is defined as \(\bigcap_{i=1}^{k} N(f_i)\). The authors introduced the Apriori-based EXCOM algorithm to find spatial co-location patterns in data sets with extended spatial representations.

Mining ST patterns in scientific data was first introduced by Yang et al. in [24]. They introduced a general framework to discover spatial associations and ST episodes for scientific data sets. The authors modeled features as geometric objects rather than points. They also extend their approach to accomodate emporal information and propose an algorithm to derive ST episodes.

The problem of mining interval orientation patterns in ST data was introduced by Patel in [15]. The event features are modeled by taking the duration of feature into account. Thus, the author was able to capture the temporal influence of a feature on other features within a spatial neighborhood. An Interval Orientation (IO) pattern is a frequent sequence of features with the annotations of temporal and directional relationships between every pair of features. The author, introduced an algorithm called IOMiner to mine frequent IO patterns.

In 2012, Ganesan Pillai et al. were the first who intro-
we do not show the sequence of evolution of our data. In our example, E has two instances (instances of different event types from our example data set in Fig. 2).

Figure 2: A ST dataset with 2D spatial objects evolving in time.

Reduced the problem of discovering STCOPs mining from data with polygon-like representations that evolve over time [17], [18].

3 Basic Concepts

3.1 Modeling STCOPs

Given a set of ST event types denoted \( E = \{e_1, \ldots, e_M\} \), and a set of continuously evolving instances \( I = \{i_1, \ldots, i_N\} \) of these event types, such that \( M \ll N \). An STCOP is a subset of ST event types that co-occur in both space and time.

In Fig. 2, we show an example data set that we will use to explain the definitions in detail. In Table 1, we show the Instance ID, Start Time, and End Time of instances of different event types from our example data set in Fig. 2. This data set contains four event types. The event type \( e_1 \) has a total of five ST instances \( \{i_1 \cap i_5\}, e_2 \) has three instances \( \{i_6 \cap i_8\}, e_3 \) has four instances \( \{i_9 \cap i_{12}\} \), and \( e_4 \) has two instances \( \{i_{13} \cap i_{14}\} \). For simplicity, in this example we do not show the sequence of 2D shapes that reflect the ST evolution of our data. In our example, \( E = \{e_1, e_2, e_3, e_4\} \), \( M = 4 \), and \( N = 14 \) (all instance IDs are listed in the first column of Table 1). From our real-life data gathered by SDO mission shown in Fig. 1, we can derive the example data set shown in Fig. 2 by following the trajectories of instances of different solar event types. In Fig. 1 time stamps are printed on top of each image showing the evolution of instances over time.

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

We can have multiple size-k STCOPs derived from the set \( E \), so to separate them we will use substrings in future definitions, (e.g., \( SE \)) with an arbitrarily chosen substracts.

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Event Type</th>
<th>Start Time (HH:MM)</th>
<th>End Time (HH:MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i_1 )</td>
<td>( e_1 )</td>
<td>10:00</td>
<td>10:30</td>
</tr>
<tr>
<td>( i_2 )</td>
<td>( e_1 )</td>
<td>10:10</td>
<td>10:40</td>
</tr>
<tr>
<td>( i_3 )</td>
<td>( e_1 )</td>
<td>11:00</td>
<td>11:20</td>
</tr>
<tr>
<td>( i_4 )</td>
<td>( e_1 )</td>
<td>13:00</td>
<td>13:30</td>
</tr>
<tr>
<td>( i_5 )</td>
<td>( e_1 )</td>
<td>11:20</td>
<td>11:50</td>
</tr>
<tr>
<td>( i_6 )</td>
<td>( e_2 )</td>
<td>10:20</td>
<td>10:50</td>
</tr>
<tr>
<td>( i_7 )</td>
<td>( e_2 )</td>
<td>10:20</td>
<td>10:40</td>
</tr>
<tr>
<td>( i_8 )</td>
<td>( e_2 )</td>
<td>11:20</td>
<td>11:40</td>
</tr>
<tr>
<td>( i_9 )</td>
<td>( e_3 )</td>
<td>10:20</td>
<td>10:50</td>
</tr>
<tr>
<td>( i_{10} )</td>
<td>( e_3 )</td>
<td>10:30</td>
<td>10:40</td>
</tr>
<tr>
<td>( i_{11} )</td>
<td>( e_3 )</td>
<td>11:20</td>
<td>11:40</td>
</tr>
<tr>
<td>( i_{12} )</td>
<td>( e_3 )</td>
<td>11:10</td>
<td>11:30</td>
</tr>
<tr>
<td>( i_{13} )</td>
<td>( e_4 )</td>
<td>11:10</td>
<td>11:30</td>
</tr>
<tr>
<td>( i_{14} )</td>
<td>( e_4 )</td>
<td>11:30</td>
<td>12:00</td>
</tr>
</tbody>
</table>

Table 1: Temporal information about event instances of data shown in Fig. 2 to denote uniqueness, i.e. \( SE_i \neq SE_j \), and \( SE_i \) and \( SE_j \) contain different event types. Note that indices \( i \) or \( j \) do not indicate number of event types involved in the co-occurrence (we only use \( k \) for that).

**Definition 2.** A pattern instance of an STCOP \( SE \), if and only if \( pat_{instance} \) contains unique instances of all event types from \( SE \). No proper subset of \( pat_{instance} \) is considered to be a pattern instance of \( SE \).

For example, \( \{i_3, i_4, i_6\} \) is a size-3 \( k = 3 \) pattern instance of co-occurrence \( SE = \{e_1, e_2, e_3\} \) in the example ST data set.

**Definition 3.** A collection of pattern instances of \( SE \) is a table instance of \( SE \), and is denoted as \( tab_{ins}(SE) \).

For example, \( \{\{i_1, i_6, i_9\}, \{i_2, i_7, i_{10}\}\} \) is a size-3 \( k = 3 \) \( tab_{ins}(SE = \{e_1, e_2, e_3\}) \) in the example data set presented in Fig. 2 and Table 1.

**Definition 4.** A prevalent STCOP is of the form \( SE_{(ccc, p)} \), where \( SE \) is an ST co-occurrence pattern, and parameters \( ccc \), \( p \) characterize the prevalent pattern in the following manner:

1. \( ccc \) stands for co-occurrence coefficient [17]. It is an indicator of the strength of a ST relation’s occurrence that is investigated. The naive STCOPs algorithm uses the ST relation Overlap for ccc. To distinguish the ST relation from the purely spatial one, we will use capital letter in the name of the former. Some examples of ST Overlap are \( \{i_1, i_6\}, \{i_2, i_7\} \), and \( \{i_7, i_{10}\} \) in Fig. 2. We will discuss this in more detail in Sec. 3.3.
2. \( p \) is the prevalence measure. The prevalence measure emphasizes how interesting the ST co-occurrences are based on prevalence. In our investigation, we used the participation index (\( pi \)), proposed in [9], as the prevalence measure. The participation index is monotonically non-increasing when the size of the STCOP increases, which is exploited for computational efficiency.

3.2 Measures

Our naive STCOPs algorithm [17] uses the ST co-occurrence coefficient to calculate ccc. The ST co-occurrence coefficient is closely related to the coefficient of areal correspondence (CAC) proposed in [21]. CAC is usually computed for any two (or more, for longer patterns) overlapping polygons as the area of intersection, divided by the area of union (i.e. spatial version of Jaccard measure). In our work [17], we extend CAC to three dimensions (two dimensions correspond to space and the third dimension corresponds to time) and calculate the ST co-occurrence coefficient using ST volumes:

(1) the volume of Intersection of two or more ST objects’ trajectories, and (2) the volume of the Union of their trajectories.

**Definition 5.** The Intersection volume of a size-k pattern instance, denoted \( V(i_1 \cap i_2 \ldots i_{k-1} \cap i_k) \), is the volume of the three dimensional object representing the Intersection of the trajectories of all instances involved in a given pattern instance.

**Definition 6.** The Union volume of a size-k pattern instance, denoted as \( V(i_1 \cup i_2 \ldots i_{k-1} \cup i_k) \), is the volume of the three dimensional object representing the Union of the trajectories of all instances involved in a given co-occurrence.
3.3 Co-occurrence coefficient $c_{se}$

We use the ST co-occurrence coefficient ($c_{se}$) to assess the strength of the ST relation Overlap. $c_{se}$ is typically calculated for a size-$k$ pattern instance as the ratio $J = V(i_1 \cap i_2 \ldots \cap i_k) / \sum_{i} V(i)$. The symbol $J$ stands for the Jaccard measure, which is commonly accepted by data mining practitioners to measure the co-occurrence of items in shopping baskets [12, 20], among spatial objects [21], and in ST data [17]. Computing the $c_{se}$ for extended ST representations such as evolving polygons is not a trivial task for massive data sets. In Fig. 3, we show the movement of a pair of instances of two different event types that change sizes, shapes and locations and movements and directions through time instances. We also show the regions of intersection and the regions of spatial unions at different time instances. Moreover, the volumes resulting from the Intersection and Union of objects’ trajectories are shown in Fig. 3. If we assume that instances $\{i_1, i_2\}$, in our example data set (Fig. 2 and Table 1), have ST Intersection volume $V(i_1 \cap i_2) = 241$ and a ST Union volume $V(i_1 \cup i_2) = 1005$, then the ST co-occurrence coefficient is equal to $c_{se} = \frac{241}{241 + 1005} = 0.23$ (see the notes under Table 2 for detailed calculation of $c_{se}$). In Table 2, the third column shows time instances (with $\Delta t=10$ min. used as our sampling interval), the fourth column $Area(i_1 \cap i_2)$ shows intersection areas, and the fifth column $Area(i_1 \cup i_2)$ shows union areas at each time instant.

![Figure 3: Example of size-2 co-occurrence of ST objects (assume $i_1$ is an instance of ST event type $e_1$, and $i_2$ is an instance of ST event type $e_2$). Color red reflects ST Intersection volume, while color green represents ST Union volume.](image)

<table>
<thead>
<tr>
<th>Instance of $e_1$</th>
<th>Instance of $e_2$</th>
<th>Time Instant ($\Delta t=10$ minutes)</th>
<th>$Area(i_1 \cap i_2)$</th>
<th>$Area(i_1 \cup i_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_1$</td>
<td>$i_2$</td>
<td>10:10</td>
<td>25</td>
<td>120</td>
</tr>
<tr>
<td>$i_1$</td>
<td>$i_2$</td>
<td>10:20</td>
<td>95</td>
<td>115</td>
</tr>
<tr>
<td>$i_1$</td>
<td>$i_2$</td>
<td>10:30</td>
<td>15</td>
<td>140</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$i_1$</td>
<td>10:40</td>
<td>0</td>
<td>150</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$i_1$</td>
<td>10:50</td>
<td>0</td>
<td>140</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$i_1$</td>
<td>11:00</td>
<td>16</td>
<td>130</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$i_1$</td>
<td>11:10</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$i_1$</td>
<td>11:20</td>
<td>0</td>
<td>60</td>
</tr>
</tbody>
</table>

$\text{c_{se}} = \frac{V(i_1 \cap i_2)}{V(i_1) + V(i_2) - V(i_1 \cap i_2)} = \frac{241}{241 + 1005 - 241} = \frac{241}{1005} = 0.23$

Table 2: Example of a pattern instance in $\text{tab}_{\text{se}}$s of $SE_i = \{e_1, e_2\}$ with calculation of $c_{se}$ from data shown in Fig. 3.

We would like to point out here that computing $J$ for ST pattern instances is quite expensive (due to the necessary calculations of intersection and union geometries for each time stamp, and storage space required to save these geometries). In this work, we introduce an alternate measure $OMAX$, defined as $\frac{\max_{i} V(i_1 \cap i_2 \ldots \cap i_k)}{\sum_{i} V(i)}$, that can be effectively used to assess the ST co-occurrence strength of a pattern instance, and it will provide significant speed-up for discovery of STCOPs based on the commonly used Jaccard measure. $OMAX$ is the foundation of our filter-and-refine approach. We filter out the pattern instances that do not satisfy the user-specified threshold, $c_{se \text{th}}$, with $OMAX$ (in Sec. 3.6 we will prove that such patterns will not satisfy $J$ with the same $c_{se \text{th}}$ as well), and then calculate $J$ for the reduced set of pattern instances. $OMAX$ represents the all-confidence measure [14] in classical association rules mining literature and it is time and storage-wise significantly cheaper to calculate than Jaccard on ST data. We will show the proofs for the completeness of STCOPs generated with our filter-and-refine approach as well as experimental results confirming our theoretical investigations and space and time scalability of our approach through the rest of this paper.

3.4 Prevalence of STCOPs

Definition 7. The participation index $pi(SE_i)$ of a ST co-occurrence $SE_i$ is defined as:

$$pi(SE_i) = \min_{k} \frac{k}{\text{pr}(SE_i, e_j)}$$

(1)

where $k$ is the size of the pattern (i.e. cardinality of $SE_i$), and the participation ratio, $pr(SE_i, e_j)$, for a ST event type $e_j$ is the fraction of the total number of instances of $e_j$ forming ST co-occurring instances in $SE_i$. For example, from Fig. 2 and Table 1 we can see that the pattern instances of $SE_i = \{e_1, e_2, e_3\}$ are $\{i_1, i_2, i_3\}, \{i_2, i_7, i_8\}$). Only two ($i_1, i_2$) out of five instances of the event type $e_1$ participate in $SE_i = \{e_1, e_2, e_3\}$, $\text{pr}(i_1, e_1, e_3) = \frac{1}{5} = 0.20$. Similarly $\text{pr}(i_1, e_2, e_3) = \frac{1}{5} = 0.20$, and $\text{pr}(i_1, e_2, e_3) = \frac{1}{5} = 0.20$. The participation index of ST co-occurrence $SE_i = \{e_1, e_2, e_3\}$ is $\text{pi}(\{e_1, e_2, e_3\}) = \min(0.40, 0.67, 0.50) = 0.40$.

Definition 8. The ST co-occurrence $SE_i$ is a prevalent pattern if it satisfies a user-specified minimum participation index threshold denoted as $pi_{th}$.

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

3.5 Problem Statement

**Input:**

1. A set of ST event types $E = \{e_1, e_2, \ldots, e_M\}$ over a common ST framework.
2. A set of $N$ ST event instances $I = \{i_1, i_2, \ldots, i_N\}$, where each $i_j \in I$ is a tuple <instance-id, ST event type, sequence of <2D shape, matching time instant> pairs>, where the sequence of 2D shape and matching time instant pairs reflects the evolution of the given ST event.
3. A user-specified ST co-occurrence coefficient threshold ($c_{se \text{th}}$).
4. A user-specified participation index threshold ($pi_{th}$).
5. A time interval of data sampling ($\Delta t$). All events are sampled with the same interval making the shapes of individual events exactly aligned in time.

**Objective/Output:** Find the complete and correct result set of spatiotemporal co-occurrence patterns (STCOPs) with $c_{se} > c_{se \text{th}}$ and $pi > pi_{th}$.

3.6 Analysis of relations between $J$ and $OMAX$ measures

In this section, we will show that we can use $OMAX$ as filter $c_{se}$ measure to prune candidates for $J$ measure, and
reduce memory cost and running time of our J-based Apri-
or algorithm [17]. For this filter step with OMAX for the
J-based Apriori algorithm to be correct the following prop-
erties between J and OMAX are necessary: (1) We show
that the cce values computed using the J and OMAX are
monotonically non-increasing as the size of the pattern in-
stance increases for a fixed cce value (see Lemmas 3.1 and
3.2 below). (2) We show the ordering relation on the select-
vivty of J and OMAX (shown in Lemma 3.3 below). (3)
We show that the STCOPs found using J is a subset of the
STCOPs found with OMAX for a fixed cce and pth values
(see Lemma 3.4 below). All three properties are useful and
necessary to reduce the number of candidate STCOPs in
an accurate filter-and-refine strategy. This will greatly
improve the performance of our naive STCOPs algorithm
[17].

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

Proof: The measure J for a size-k pattern instance is
defined as:
\[ V(i_1 \cap i_2, \ldots, i_k) \]
\[ \frac{V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1})}{V(i_1 \cup \ldots, i_k \cup i_{k+1})} \] (2)

For any size-(k + 1) pattern instance denoted as
\text{pat}_{\text{instance}}^\prime is equal to \text{pat}_{\text{instance}} \cup (i_{k+1}), where
\text{pat}_{\text{instance}} is a size-k pattern instance and \( i_{k+1} \notin \text{pat}_{\text{instance}} \).
Proof: We claim the measure J follows the relation:
\[ V(i_1 \cap i_2, \ldots, i_k) \leq V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \]
\[ V(i_1 \cup i_2, \ldots, i_k) \geq V(i_1 \cup i_2, \ldots, i_k \cup i_{k+1}) \] (3)

Therefore, we need to prove:
\[ V(i_1 \cap i_2, \ldots, i_k) \geq V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \], (4)

and
\[ V(i_1 \cup i_2, \ldots, i_k) \leq V(i_1 \cup i_2, \ldots, i_k \cup i_{k+1}) \] (5)

Since, adding one more instance of a different event type to a
pattern instance can either reduce or not affect the volume
of Intersection of instance trajectories, we obtain the relation
\[ V(i_1 \cap i_2, \ldots, i_k) \geq V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \]
from Eq. 4. Similarly, adding one more instance of different
event type to a pattern instance can either increase or not affect
the volume of Union of instance trajectories, we obtain the
relation \[ V(i_1 \cup i_2, \ldots, i_k) \leq V(i_1 \cup i_2, \ldots, i_k \cup i_{k+1}) \]
from Eq. 5. Thus, our relation in Eq. 3 holds for all
positive real numbers that represent volumes of ST objects
with evolving polygons [17].

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

Proof: The measure OMAX for a size-k pattern instance is
defined as:
\[ \frac{V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1})}{\max(V(i_1), \ldots, V(i_k))} \] (6)

For any size-(k + 1) pattern instance denoted \text{pat}_{\text{instance}}^\prime is equal to \text{pat}_{\text{instance}} \cup (i_{k+1}), where
\text{pat}_{\text{instance}} is a size-k pattern instance and \( i_{k+1} \notin \text{pat}_{\text{instance}} \). We claim
the measure OMAX follows the relation:
\[ \frac{V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1})}{\max(V(i_1), \ldots, V(i_k))} \geq \frac{V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1})}{\max(V(i_1), \ldots, V(i_k))} \] (7)

Therefore, we need to prove:
\[ V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \geq V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \] (8)
and
\[ \max(V(i_1), \ldots, V(i_k)) \leq \max(V(i_1), \ldots, V(i_{k+1})) \] (9)

Since, once again, adding one more instance of different
event type to a pattern instance can either reduce or not af-
fect the volume of the Intersection of the instance trajec-
tries; we obtain the relation shown in Eq. 8. Similarly, adding
another instance of a different event type to a pattern in-
stance can not reduce the maximum volume of instance tra-
jectories, so we obtain the relation \[ \max(V(i_1), \ldots, V(i_k)) \leq \max(V(i_1), \ldots, V(i_{k+1})) \].
Thus, Eq. 7 holds for all positive real numbers that represent volumes of ST objects
with evolving polygons [17].

Lemma 3.3 : The selectivity of the measures J and
OMAX for a size-k pattern instance follows the order
\[ V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \leq V(i_1 \cap i_2, \ldots, i_k \cap i_{k+1}) \]
\[ V(i_1 \cup i_2, \ldots, i_k \cup i_{k+1}) \] (9) (10)

Proof: Since numerators are same, for the ordering rela-
tion \[ J \leq OMAX \], we can derive relations between both
denominators:
\[ \max(V(i_1), \ldots, V(i_k)) \leq V(i_1 \cup i_2, \ldots, i_k \cup i_{k+1}) \] (10)

Maximum volume of all trajectories is always less than or
equal to the volume of union of all of them, thus the relation
J \leq OMAX always holds for positive real numbers [17].

Lemma 3.4 : For a given user-specified participation index
denominator \( p_{th} \) and ST co-occurrence strength threshold
\( cce_{th} \), the set of STCOPs generated using J, is a subset
of STCOPs generated using OMAX measure, for the same
cce_{th} and \( p_{th} \).

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

For given user-specified thresholds cce_{th} and \( p_{th} \), we rep-
resent the set of all STCOPs generated for J as STCOPJ,
and the set of all STCOPs generated for OMAX as
STCOPOMAX. Furthermore, we denote participation in-
dex \( p(SE_i) \) of a ST co-occurrence SE_i as (Def. 7), derived
by using J as \( p_j(SE_i) \), and utilizing measure OMAX as
\( p_{OMAX}(SE_i). \) From Lemma 3.3, we know that the num-
ber of pattern instances found for a ST co-occurrence SE_i
follows the order J \leq OMAX, thus we get,
\[ \min_j p_j(SE_i, \epsilon_j) \leq \min_j p_{OMAX}(SE_i, \epsilon_j) \] (11)

Since participation index is anti-monotonic as the size of the
pattern increases [9], and from Lemmas 3.1 and 3.2, and
from Eq. 12, we get STCOPJ \subseteq STCOPOMAX [17].

4 FastSTCOPs-Miner Algorithm
In this section, we introduce the FastSTCOPs-Miner algo-

rithm, which is more efficient than our naive STCOPs algo-

rithm [17] in the context of needed memory as well as the
execution time while leading to exactly same results. This is
because we apply a filter-and-refine strategy in each iteration of the algorithm. The FastSTCOPs-Miner algorithm exploits the containment relation between the STCOPs generated using Jaccard ($J$) and $OMAX$ measures (see Sec. 3.6), to filter out candidate patterns that can not form STCOPs with the $J$.

The FastSTCOPs-Miner algorithm first filters STCOPs with $OMAX$, and then uses these filtered STCOPs to find the refined prevalent STCOPs with our standard measure, that is Jaccard ($J$). These refined prevalent STCOPs, like in all Apriori algorithms [1], are used to generate candidate STCOPs in the next iteration of the algorithm. Thus, the FastSTCOPs-Miner algorithm continuously uses a filter-and-refine strategy at each iteration of the algorithm to generate prevalent STCOPs. Even though such duplication of efforts may seem unnecessary, in Sec. 5, we will experimentally show on multitude of real-life and artificial data sets, the impressive effectiveness of our filter-and-refine strategy on ST data with evolving regions. We will also provide detailed explanation for this effectiveness.

Next, we give the pseudocode of the proposed FastSTCOPs-Miner algorithm (see Fig. 4), and explain the algorithm with a running example using the data set already shown in Fig. 2 and Table 1.

For our FastSTCOPs-Miner algorithm shown in Fig. 4, the inputs and outputs are defined as in Sec. 3.5. Steps 1 and 2 of proposed algorithm initialize the data parameters and data structures, steps 3 through 11 give an iterative process to discover the STCOPs of size greater than two. Steps 3 through 11 continue until there are no candidate STCOPs to be discovered as shown by loop condition in step 3. Step 12 returns the union of all prevalent STCOPs (patterns of all sizes). The explanations of functions in the algorithm are:

**Generation of table instances of size-1 (step 2).**
In this function, argument $\Delta t$ represents the size of increment in time. The evolution of instances of our ST events from their birth (start) time is registered using $\Delta t$ as our time sampling frequency. The combination of the event instance ID and time step allows us to identify the appropriate spatial representation of an event at the particular moment. For example, Fig. 5 (a) shows the key columns of table instances of size-1 for our sample ST data set (Fig. 2 and Table 1). Here, the $\Delta t$ value was set to 10 minutes. The column denoted $tab_{ins}(e_1)$ represents the table instance of size-1 for event type $e_1$. Similarly, the columns denoted by $tab_{ins}(e_2)$, $tab_{ins}(e_3)$, and $tab_{ins}(e_4)$ represent the table instances of size-1 for event types $e_2$, $e_3$, and $e_4$. The geometric shapes of instances in each of the presented time instances are not shown in Fig. 5 (a) for simplicity.

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

**Generation of filtered table instances of size-(k+1) (step 5).** This function generates table instances for candidate patterns of size-(k+1). Pattern instances for each table instance can be generated by a ST join query. The geometric shapes of the instances at each time step are saved, as these geometric shapes will be used for finding the cce of STCOPs of size three or more. In this function, we calculate the $cce$ for each pattern instance by using $OMAX$. Pattern instances that have a $cce$ below the user-specified $OMAX$ value are deleted from the table instance, since we know from proofs in Sec. 3.6 that they also cannot satisfy $cce_{th}$ requirement for $J$ measure.

For example, in the table instance $tab_{ins}(e_1e_2)$ shown in Fig. 6 (c), the rows $i_1, i_6, 10:00$ through $i_1, i_6, 10:50$ represent a pattern instance that satisfies the threshold $cce_{th} = 0.01$. As another example, in Fig. 7 (f) we show the filtered table instances generated from candidate patterns of size $k = 3$.

![Figure 5: (a) Table Instances of size-1 and (b) Candidate Patterns for size-2 STCOPs](image)

![Figure 6: (c) Filtered ($T_{OMAX}$) and (d) Refined ($TR_{12}$) Table Instances of size-2](image)

Generation of filtered prevalent patterns size-(k+1) (step 6). This function discovers filtered size-(k+1)
Inputs:
See Sec. 3.5.

Variables:
(1) $k$ & the co-occurrence size (Def. 1).
(2) $CR_{jk}$: the set of candidates for size-$(k)$ STCOPs derived from size-$(k-1)$ refined prevalent STCOPs.
(3) $T_{OMAX}^{k}$: a set of filtered instances of size-$(k)$ ST co-occurrences.
(4) $TR_{jk}$: a set of refined instances of size-$(k)$ ST co-occurrences (see Def. 3).
(5) $P_{OMAX}^{k}$: the set of size-$k$ filtered STCOPs derived from size-$k$ candidate STCOPs.
(6) $PR_{jk}$: the set of size-$k$ refined prevalent STCOPs derived from size-$k$ filtered STCOPs (see Def. 8).
(7) $PR_{final}$: the union of all refined prevalent STCOPs (patterns of all sizes). // This is the final Jaccard-based prevalent patterns.

Algorithm:
1. $k = 1$, $C_k = E$, $PR_{jk} = E$; $PR_{final} = \emptyset$;
2. $TR_{jk} = genJob(C_k, I, \Delta t)$;
3. while ($PR_{jk} \neq \emptyset$) {
   4. $CR_{jk+1} = gen_candidate_coocc(\{PR_{jk}\})$;
   5. $P_{OMAX}^{k+1} = gen_table_instance_filtered(CR_{jk+1}, TR_{jk+1}, cce_{th})$;
   6. $P_{OMAX}^{k+1} = pre_prune_coocc(P_{OMAX}^{k+1}, J_{th})$;
   7. $TR_{jk+1} = gen_table_instance_refined(P_{OMAX}^{k+1}, TR_{jk+1}, cce_{th})$;
   8. $PR_{final} = PR_{final} \cup PR_{jk+1}$;
   9. $k = k + 1$;
10. } return $PR_{final}$;

Figure 4: FastSTCOPs-Miner Algorithm

Figure 7: (e) Candidate Patterns of size-$3$ and (f) Filtered ($T_{OMAX}$) and (g) Refined ($TR_{jk}$) Table Instances of size-$3$.

STCOPs by pruning candidate patterns in $CR_{jk+1}$ that have $pi < pi_{th}$.

For example, we show the $pi$ value (See Def. 7) at each table instance in Fig. 6 (c). As seen from the Fig. 6 (c), the pattern $SE_i = \{e_1, e_4\}$, and $SE_i = \{e_2, e_4\}$ will be pruned if a value of 0.40 is set to $pi_{th}$. Thus, the patterns that satisfy the $pi_{th} = 0.40$ are $\{e_1, e_2\}, \{e_1, e_3\}$, $\{e_2, e_3\}, \{e_3, e_4\}$.

As another example, we show the $pi$ value (See Def. 7) at each table instance in Fig. 7 (f). As seen from the Fig. 7 (f), the pattern $SE_i = \{e_1, e_2, e_3\}$ is a prevalent pattern if a value of 0.40 is set to $pi_{th}$.

Figure 7: (e) Candidate Patterns of size-$3$ and (f) Filtered ($T_{OMAX}$) and (g) Refined ($TR_{jk}$) Table Instances of size-$3$.

STCOPs by pruning candidate patterns in $CR_{jk+1}$ that have $pi < pi_{th}$.

For example, we show the $pi$ value (See Def. 7) at each table instance in Fig. 6 (c). As seen from the Fig. 6 (c), the pattern $SE_i = \{e_1, e_4\}$, and $SE_i = \{e_2, e_4\}$ will be pruned if a value of 0.40 is set to $pi_{th}$. Thus, the patterns that satisfy the $pi_{th} = 0.40$ are $\{e_1, e_2\}, \{e_1, e_3\}, \{e_2, e_3\}, \{e_3, e_4\}$.

As another example, we show the $pi$ value (See Def. 7) at each table instance in Fig. 7 (f). As seen from the Fig. 7 (f), the pattern $SE_i = \{e_1, e_2, e_3\}$ is a prevalent pattern if a value of 0.40 is set to $pi_{th}$.

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

For example, in Fig. 6 (d) we show the refined table instances generated from the refined prevalent patterns obtained in step 6 for a $pi_{th}$ value of 0.40. In each of the table instances shown in Fig. 6 (d), we show the key columns of pattern instances that satisfy the threshold $J_{cco}$ = 0.01 value calculated using the $J$ measure. For example, for the table instance $tab_{ins}(e_1e_2)$, the pattern instances that satisfy the $J_{cco} = 0.01$ for $J$ are $\{i_1, i_6\} \cup \{i_2, i_7\}$. Note the pattern instance $\{i_4, i_8\}$ is dropped from the table instance $tab_{ins}(e_1e_2)$ as it does not satisfy $J_{cco} = 0.01$ (see Fig. 6 (c) and (d) to compare).

As another example, in Fig. 7 (g) we show the refined table instances generated from refined prevalent patterns obtained in step 6 for a $pi_{th}$ value of 0.40. In the table instance $tab_{ins}(e_1e_2e_4)$ shown in Fig. 7 (g), we show the key columns of pattern instances that satisfy the threshold $J_{cco}$ value calculated using $J$. Please note the pattern instance $\{i_1, i_6, i_9\}$ is dropped from the table instance $tab_{ins}(e_1e_2e_4)$ as its cce is smaller than our $J_{cco} = 0.01$ (see Fig. 7 (f) and (g) to compare).

Generation of refined prevalent patterns size-$(k+1)$ (step 8). This function discovers refined size-$(k+1)$ prevalent STCOPs by pruning $P_{OMAX}^{k+1}$ that have $pi < pi_{th}$. As seen from the Fig. 7 (g), the pattern $SE_i = \{e_1, e_2, e_3\}$ will be pruned if $pi_{th}$ is set to 0.40.

In step 9, we calculate the union of refined prevalent patterns. The algorithm runs iteratively until no more STCOPs can be generated (our $PR_{jk+1}$ is empty), and returns all prevalent STCOPs, in step 12. Since we do not have any patterns left that satisfies the threshold $pi_{th} = 0.40$ in our example data set shown in Fig. 2 and Table 1, the algorithm would terminate at $k = 3$ for our running example.
5 Experimental Evaluation

In this section, we compare our FastSTCOPs-Miner algorithm against the classic Apriori-based approach [17] which we will call Naïve STCOPs algorithm. In our experiments, we are using three real-life data sets from the solar physics domain and one artificial data set.

In the real-life data sets, we evaluate our algorithms using six types of evolving solar phenomena. Our real-life data sets contain evolving instances of six different solar event types, which were observed on 01/01/2012 (denoted Data Set A), 01/01/2012 through 01/03/2012 (denoted Data Set B), and 01/01/2012 through 01/05/2012 (denoted Data Set C). We obtained our data sets from the well-known solar data repository called Heliophysics Event Knowledgebase (HEK) [7],[19]. The six different solar event types in our data sets are: Active Region, Filament, Sigmoid, Sunspot, Flare, and Emerging Flux.

The artificial data set (denoted Data Set D) is generated based on the works of Huang et al. in [9]. The artificial data set generator creates a data set of event instances with spatiotemporal features for spatial framework of size $D \times D$. Event types are generated with random size, speed, duration and area change parameters. Number of events to be generated is an input parameter to dataset generator, $M$. We used artificial data set to investigate behaviour of algorithms for larger number of event types. All of our data sets are available on-line to let researchers interested in this topic reproduce our experiments, and maybe even improve on our solution. The website for this paper/research can be found at [8].

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

5.1 Memory Usage

We first investigated the memory usage of the FastSTCOPs-Miner and Naïve STCOPs algorithms for the candidate table instances generated. We report the hard-drive memory usage of candidate table instances with all the pattern instances generated (see bars in solid colors - black and white in Fig. 8), and memory usage of candidate table instances after filtering the pattern instances that do not satisfy threshold $c_{cc}\epsilon_{lth}$ (see bars with pattern markings in Fig. 8). In Fig. 8 bar labelled Naïve J-BCCE (black bars) represents the memory usage of table instances for all the pattern instances generated. The Naïve STCOPs algorithm (from 187 MB in the first black/Naïve J-BCCE bar to 46 MB in the first white/Naïve J-BCCE bar). This benefit continues through remaining steps of our algorithm. As expected, from Fig. 8 we can observe, that there is a drop in the memory usage after the pattern instances are pruned out by applying the threshold $c_{cc}\epsilon_{lth}$ (see and compare Naïve J-BCCE in black color with Naïve J-ACCE marked with diagonal upward stripes in Fig. 8). However, generation of all J-BCCE’s is unnecessary to discover actual STCOPs (i.e. our J-ACCE’s).

Also, in Fig. 8 OMAX-BCCE represents the memory usage of table instances for all the pattern instances generated. OMAX-ACCE represents the memory usage of table instances after filtering the pattern instances that do not satisfy the threshold $c_{cc}\epsilon_{lth}$ in FastSTCOPs-Miner algorithm. This time OMAX-BCCE to OMAX-ACCE ra-
tio represents selectivity (i.e. pruning power) of our filter step in the FastSTCOPs-Miner algorithm. Please note, here the cce value is calculated using $OMAX$, so the $J$-based refine step is fed by the filtered out data (i.e. satisfying $cce_{th}$ and $pi_{th}$ ($pi_{OMAX}$)) but without missing any relevant patterns. Moreover, J-BCCE represents the memory usage of table instances for all the pattern instances generated from the filtered pattern instances (that is from our OMAX-ACCE bars that satisfy $pi_{th}$ ($pi_{OMAX}$)) and J-ACCE represents the memory usage of table instances after filtering the pattern instances that do not satisfy the threshold $cce_{th}$ in FastSTCOPs-Miner algorithm. Please note, here the cce value is calculated using $J$, to find patterns that are relevant. Also, the total number of pattern instances for OMAX-ACCE (bars with checkered pattern) is greater than or equal to J-BCCE (because of filtering effect of threshold $pi_{th}$ ($pi_{OMAX}$)); however, the memory usage increases for J-BCCE because of the union geometries generated for all of the pattern instances in order to calculate cce’s using $J$ measure. Furthermore, from Fig. 8 we can observe a decrease in the memory usage after the pattern instances are filtered by applying threshold $cce_{th}$ (please compare OMAX-BCCE with OMAX-ACCE and J-BCCE with J-ACCE, respectively). This shows the effectiveness of the anti-monotone property of the measures $J$ and $OMAX$ (see Lemma 3.1 and 3.2) and the benefit of our $OMAX$-based pruning strategy (see Lemma 3.4).

5.2 Execution Time

Figure 9: Comparison of execution time for the FastSTCOPs-Miner and Naïve STCOPs.

Next, we show the execution times of our FastSTCOPs-Miner and Naïve STCOPs algorithms. Fig. 9 shows the execution time for patterns of different sizes. As expected, our FastSTCOPs-Miner algorithm outperforms the original Naïve STCOPs algorithm, since it uses a filter-and-refine strategy to find pattern instances that satisfy the threshold $cce_{th}$ for $J$. The Naïve STCOPs algorithm generates computationally expensive Union geometries for all the pattern instances (see the bars labelled as Naïve J-BCCE in Fig. 8) to realize how much memory overhead this process generates for data sets $A$, $B$, $C$, and $D$, while our FastSTCOPs-Miner algorithm generates Union geometries for smaller data set (see the bars labelled as J-BCCE in Fig. 8). This memory overhead causes the execution time of the Naïve STCOPs algorithm to be slower in comparison to our FastSTCOPs-Miner algorithm.

5.3 Pattern Instances

Figure 10: Pattern instances found (filtered with $OMAX$ and refined with $J$) using the FastSTCOPs-Miner and Naïve STCOPs.

Fig. 10 shows the counts of pattern instances that satisfy the threshold value $cce_{th}$. We compare the counts of pattern instances satisfying the threshold $cce_{th}$ with $OMAX$ (FastSTCOPs-Miner (OMAX)) and $J$ (FastSTCOPs-Miner ($J$)) for our FastSTCOPs-Miner algorithm. We also compared the counts of pattern instances of the FastSTCOPs-Miner and Naïve STCOPs algorithms. As shown in Lemma 3.3, we can observe in Fig. 10 that the selectivity of the measures $OMAX$ and $J$ follows the order $OMAX \geq J$. Moreover, the count of pattern instances found for the threshold value $cce_{th}$ is the same for our FastSTCOPs-Miner ($J$) and the Naïve STCOPs ($J$) algorithm in Fig. 10. The identical results between FastSTCOPs-Miner ($J$) and Naïve STCOPs ($J$) confirm the correctness of our implementation of the FastSTCOPs-Miner algorithm.

6 Conclusion and Future Work

Empirical results on three real-life data sets, drawn from the solar physics discipline, and an artificial data set, serve to validate our FastSTCOPs-Miner algorithm. From the result of Fig. 8, we exhibit the memory space usage of candidate table instances for the FastSTCOPs-Miner and the Naïve STCOPs algorithms by showing the memory space comparison before and after pruning the pattern instances with threshold $cce_{th}$. This result is important in order to see the effects of the anti-monotonic property of the measures $J$ and $OMAX$ (see Lemma 3.1 and 3.2). We also showed that our FastSTCOPs-Miner algorithm is faster in comparison to Naïve STCOPs algorithm in Fig. 9. These results show the significance of our filter and refine approach used in FastSTCOPs-Miner algorithm letting us to create the fastest STCOPs mining algorithm from evolving regions that is currently available on our planet.

For the future work, we plan to investigate developing
non-Apriori approaches to discover STCOPs. One approach is to use frequent pattern growth [6] technique to discover STCOPs. Frequent pattern growth approach is shown to outperform Apriori-based approaches [6].

Acknowledgment

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

7 References


