A Novel Breadth-first Strategy Algorithm for Discovering Sequential Patterns from Spatio-temporal Data

Piotr S. Maciąg and Robert Bembenik
Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland

Keywords: Sequential Patterns, Spatio-temporal Data, Crime Incidents.

Abstract: In the paper, we consider the problem of discovering sequential patterns from dataset of event instances and event types. We offer a breadth-first strategy algorithm (spatio-temporal breadth-first miner, STBFM) to search for significant sequential patterns denoting relations between event types in the dataset. We introduce Sequential Pattern Tree (SPTree), a novel structure significantly reducing the time of patterns mining process. Our algorithm is compared with STMiner - the algorithm for discovering sequential patterns from event data. A modification of STBFM allowing to discover Top-N most significant sequential patterns in a given dataset is provided. Experimental studies have been performed on the crime incidents dataset for Boston city.

1 INTRODUCTION

We consider the problem of discovering sequential patterns from a dataset of event instances D and event types F. Each instance $e \in D$ is defined by its event instance identifier, spatial location (e.g. geographical coordinates), occurrence time and event type $f \in F$. The sequential pattern is defined as a sequence of event types $\overrightarrow{s} = f_{i_1} \rightarrow f_{i_2} \rightarrow \cdots \rightarrow f_{i_n}$, where $f_{i_1}, f_{i_2}, \ldots, f_{i_n} \in F$. For any two consecutive event types participating in a significant sequential pattern $f_{i_{j-1}} \rightarrow f_{i_j}$, event instances of the former event type attract in their spatio-temporal neighborhoods occurrences of instances of the latter event type. Let us consider Fig. 1, where we show spatio-temporal dataset $D = \{a1, a2, b1, b2, b3, b4, b5, b6, b7, b8, c1, c2, c3, c4\}$ and $F = \{A, B, C\}$. For the dataset depicted in Fig. 1, a possible significant sequential pattern is $\overrightarrow{s} = A \rightarrow B \rightarrow C$.

Discovering sequential patterns from spatio-temporal event datasets has been introduced in (Huang et al., 2008), where significant sequential pattern is defined as the sequence of event types. The notion of sequential pattern introduced in (Huang et al., 2008) has been extended to the notion of cascading pattern in (Mohan et al., 2012). Algorithms for discovering sequential patterns from databases containing transactions of items such as SPADE or PrefixSpan have been proposed in (Zaki, 2001), (Pei et al., 2004).

Discovering a top set of the most important sequential patterns is the topic well studied in data mining. This idea was introduced in (Han et al., 2002; Tzvetkov et al., 2003). Some recent publications related to mining top high utility sequential patterns are (Yin et al., 2013), which proposes the Top-K Utility Sequences (TUS) algorithm. (Wu et al., 2012), which introduces a similar algorithm but for the problem of frequent itemsets mining. Other publications considering a similar problem are (Lee and Park, 2016; Petitjean et al., 2016; Feremans et al., 2018).

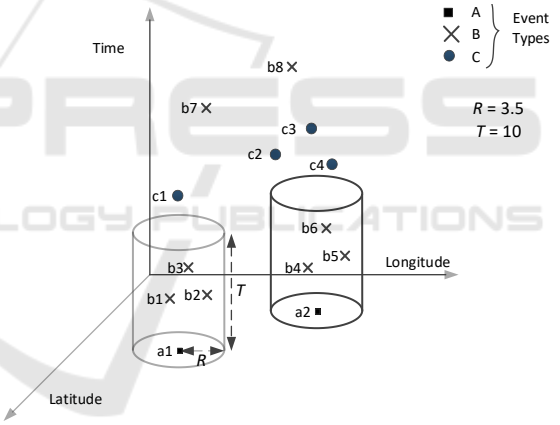

Figure 1: An event dataset with neighborhood spaces marked for instances of type A.

This paper provides the following contributions:

- The definition of a breadth-first search algorithm (Spatio-Temporal Breadth-First Miner, STBFM) for discovering significant sequential patterns in

Maciąg, P. and Bembenik, R.
A Novel Breadth-first Strategy Algorithm for Discovering Sequential Patterns from Spatio-temporal Data.
DOI: 10.5220/0007355804590466
In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 459-466
ISBN: 978-989-758-351-3

event-based spatio-temporal data.

- A Sequential Pattern Tree (SPTree) structure for more efficient candidate generation and the version of STBFM utilizing SPTree is given.
- An important data mining problem is to discover the set of N most significant patterns rather than all patterns with significance measure greater than the given threshold. To do this end, we provide a modification of STBFM discovering Top-N most significant patterns.
- The proposed algorithms have been verified using crime related data for Boston city for year 2014. The performed experiments resulting in interesting patterns and are explained in section 5.
- Proposed algorithms have been compared with STMiner introduced in (Huang et al., 2008) and discovering the same type of patterns. Experimental results show significant improvement in the execution time of our algorithms compared to STMiner.

2 BASIC NOTIONS

In this section, we give the notions and formulate our algorithm discovering all significant sequential patterns using the breadth-first searching strategy.

Definition 1 (Neighborhood Space)**.** By $V_{N(e)}$ we denote the neighborhood space of instance e. For $V_{N(e)}$ having cylindrical shape, R denotes the spatial radius and T the temporal interval of that space.

In Figure 1, we show two neighborhood spaces: $V_{N(a1)}, V_{N(a2)}$.

Definition 2 (Neighborhood with Respect to Event Type (Huang et al., 2008))**.** For a given event instance e, the neighborhood of e is defined as follows:

$$\begin{aligned} N_f(e) = &\{e | p \in D(f) \\ &\wedge distance(p.location, e.location) \leq R \\ &\wedge (p.time - e.time) \in [0,T]\} \end{aligned} \quad (1)$$

where R denotes the spatial radius and T temporal interval of the neighborhood space $V_{N(e)}$ and $D(f)$ is the set of event instances of type f in dataset D.

As the neighborhood $N_f(e)$ of instance e, we denote the set of instances of type f contained inside the neighborhood space $V_{N(e)}$. The neighborhood of instance $a1$ (shown in Figure 1) with respect to event type B is $N_B(a1) = \{b1, b2, b3\}$.

Definition 3 (Set of Instances)**.** For a sequence of event types $\overrightarrow{s} = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m]$ of length m, the sets of instances $I(\overrightarrow{s}[1]), I(\overrightarrow{s}[2]), \ldots, I(\overrightarrow{s}[m])$ participating in the sequence $\overrightarrow{s}$ are defined as follows:

1. For an event type $\overrightarrow{s}[1]$, the set of instances $I(\overrightarrow{s}[1])$ is defined as:

$$I(\overrightarrow{s}[1]) = D(\overrightarrow{s}[1]) \quad (2)$$

2. For event types $\overrightarrow{s}[2] \ldots \overrightarrow{s}[m]$ with $i = 2, 3, \ldots, m$, the sets of instances $I(\overrightarrow{s}[i])$ are defined as:

$$I(\overrightarrow{s}[i]) = distinct\Big(\bigcup_{e \in I(\overrightarrow{s}[i-1])} N_{\overrightarrow{s}[i]}(e) \Big) \quad (3)$$

For the first event type participating in the sequence $\overrightarrow{s}$, the set of instances $I(\overrightarrow{s}[1])$ is equal to the set of instances of type $\overrightarrow{s}[1]$ in D: $D(\overrightarrow{s})$. For the next event types in $\overrightarrow{s}$, the sets $I(\overrightarrow{s}[i])$ are defined as the sets of distinct instances contained in the neighborhoods of instances from $I(\overrightarrow{s}[i-1])$. Let us consider a pattern $\overrightarrow{s} = A \rightarrow B$ for the dataset given in Figure 1. For this pattern we have the following: $I(\overrightarrow{s}[1]) = \{a1, a2\}, I(\overrightarrow{s}[2]) = \{b1, b2, b3, b4, b5, b6\}$.

Definition 4 (Participation Ratio)**.** For a sequence $\overrightarrow{s} = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m]$, the participation ratio between any two consecutive event types contained in $\overrightarrow{s}$ is defined as follows:

$$PR(\overrightarrow{s}[i-1] \rightarrow \overrightarrow{s}[i]) = \frac{\left|I(\overrightarrow{s}[i])\right|}{\left|D(\overrightarrow{s}[i])\right|} \quad (4)$$

that is, as the number of distinct instances of event type $\overrightarrow{s}[i]$ contained in the neighborhoods of instances of event type $\overrightarrow{s}[i-1]$ divided by the number of instances of type $\overrightarrow{s}[i]$ in the dataset D.

The participation ratio between any two consecutive event types $\overrightarrow{s}[i-1], \overrightarrow{s}[i]$ participating in $\overrightarrow{s}$ is defined as the ratio of $|I(\overrightarrow{s}[i])|$ to $|D(\overrightarrow{s}[i])|$ and its value is in the range $[0,1]$.

Definition 5 (Participation Index)**.** For a given m-length sequence $\overrightarrow{s} = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m]$, the participation index is defined as follows:

1. When $m = 2$ then:

$$PI(\overrightarrow{s}) = PR(\overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2]) \quad (5)$$

2. When $m > 2$ then:

$$PI(\overrightarrow{s}) = \min \begin{cases} PI(\overrightarrow{s}^*), \\ PR(\overrightarrow{s}[m-1] \rightarrow \overrightarrow{s}[m]) \end{cases} \quad (6)$$

where sequence $\overrightarrow{s}^* = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m-1]$.

The participation index is the minimum from all participation ratios calculated over any two consecutive event types in $\overrightarrow{s}$. Again, let us consider the pattern $\overrightarrow{s} = A \rightarrow B$ from Figure 1. For such a pattern $PI(\overrightarrow{s}) = 0.75$ ($PI(A \rightarrow B) = PR(A \rightarrow B) = \frac{6}{8} = 0.75$). Participation index preserves the antimonotonicity property (Mohan et al., 2012). For a sequence $\overrightarrow{s}$, participation indexes of any of its subsequences are greater or equal to $PI(\overrightarrow{s})$. This property allows us to define spatio-temporal breadth-first miner (STBFM), the algorithm given in the following section.

3 SPATIO-TEMPORAL BREADTH-FIRST MINER ALGORITHM

We propose to discover all significant sequential patterns with participation indexes greater than threshold θ. The initial value of threshold θ is given by the user. In addition, the user is specifying the dimensions of the neighborhood space: R and T. Algorithm 1 starts with generating candidate patterns of length 2 and verifying their participation indexes. The length 2 candidates are generated by joining every event type with every other event type in the dataset. The set of instances participating in such a candidate pattern is calculated according to Definition 3. To calculate neighborhoods of instances we used the plane sweep algorithm proposed in (Arge et al., 1998). In the next phase, the sets of candidates of length k are generated and verified iteratively, until the set of significant sequential patterns of length $k-1$ is empty. By L_k we denote the set of significant patterns of length k ($\mathcal{L}$ denotes the family of such sets). Similarly C_k denotes a candidate set of patterns with length k.

Algorithm 2 is responsible for verifying candidate patterns. If participation indexes of candidates are greater or equal to the actual θ threshold, then the candidates are inserted into the L_m set.

Algorithm 3 generates a set of candidate patterns of length m based on the discovered significant patterns of length $m-1$. The generation process is as follows: two patterns of length $m-1$ are joined to generate a candidate pattern of length m, if they contain the same event types on the following positions: the second event type of the first pattern is the same as the first event type of the second pattern, the third event type of the first pattern is the same as the second event type of the second pattern and up to the last event type of the first pattern which should be the same as the one

Algorithm 1: Spatio-temporal breadth-first miner for discovering significant sequential patterns (STBFM).

Require: D - a dataset containing event types and their instances, F - a set of event types, R, T - a specification of neighborhood spaces, θ - a significance threshold for the discovered sequences.
Ensure: Top - the set of top N most significant sequential patterns.
1: $C_2 :=$ generate candidate patterns of length 2.
2: $L_2 :=$ VerifyCandidates(C_2).
3: $k := 3$.
4: **while** $L_{k-1} \neq \emptyset$ **do**
5: $C_k :=$ CandidateGen(L_{k-1}).
6: $L_k :=$ VerifyCandidates(C_k).
7: Add L_k to $\mathcal{L}$.
8: $k := k+1$.
9: **end while**
10: **return** $\mathcal{L}$.

Algorithm 2: VerifyCandidates(C_m).

Require: C_m - a set of candidate sequential patterns of length m, θ - a significance threshold.
Ensure: L_m - a set of significant sequential patterns of length m.
1: $L_m := \emptyset$.
2: **for** each $\overrightarrow{s} \in C_m$ **do**
3: CalculatePI($\overrightarrow{s}$).
4: **if** PI($\overrightarrow{s}$) $\geq \theta$ **then**
5: Add $\overrightarrow{s}$ to L_m.
6: **end if**
7: **end for**
8: **return** L_m.

before the last event type of the second pattern (step 4 in Algorithm 3). In such a case, the new candidate sequential pattern contains all event types from the first merged pattern and the last event type from the second merged pattern (step 5 of Algorithm 3). The calculation of event instances participating in the new candidate pattern is defined in step 6. For this calculation these two cases are possible:

1. For the event types from the first to the one before last, the sets of participating instances are sets of instances of respective event types in the first merged pattern.
2. For the last event type, the set of instances is contained in the neighborhoods of instances of the one before the last event type. To calculate such neighborhoods we used the procedure presented in Algorithm 5.

In Algorithm 5, $e \in I(\overrightarrow{s}_i[m-1])$ denotes the set of instances in $I(\overrightarrow{s}_i[m-1])$, and $N_{\overrightarrow{s}_j[m-1]}(e))$ are

Algorithm 3: CandidateGen(L_{m-1}).

Require: L_{m-1} - a set of significant sequential patterns of length $m-1$.
Ensure: C_m - a set of candidate sequential patterns of length m.
1: $C_m := \emptyset$.
2: **for** each $\overrightarrow{s}_i \in L_{m-1}$ **do**
3: **for** each $\overrightarrow{s}_j \in L_{m-1} \wedge \overrightarrow{s}_i \neq \overrightarrow{s}_j$ **do**
4: **if** $\overrightarrow{s}_i[2] = \overrightarrow{s}_j[1] \wedge \overrightarrow{s}_i[3] = \overrightarrow{s}_j[2] \wedge \cdots \wedge \overrightarrow{s}_i[m] = \overrightarrow{s}_j[m-1]$ **then**
5: $\overrightarrow{s} := \overrightarrow{s}_i[1] \rightarrow \overrightarrow{s}_i[2] \rightarrow \cdots \rightarrow \overrightarrow{s}_i[m-1] \rightarrow \overrightarrow{s}_j[m-1]$.
6: $I(\overrightarrow{s}[1]) := I(\overrightarrow{s}_i[1]) \wedge I(\overrightarrow{s}[2]) := I(\overrightarrow{s}_i[2]) \wedge \cdots \wedge I(\overrightarrow{s}[m-1]) := I(\overrightarrow{s}_i[m-1]) \wedge$ $I(\overrightarrow{s}[m]) :=$ CalculateNeighborhood$(I(\overrightarrow{s}_i[m-1]), I(\overrightarrow{s}_j[m-1]))$.
7: Add $\overrightarrow{s}$ to C_m.
8: **end if**
9: **end for**
10: **end for**
11: **return** C_m.

Algorithm 4: CalculatePI($\overrightarrow{s}$).

Require: $\overrightarrow{s} = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m-1] \rightarrow s[m]$ - a sequence of event types.
1: **return** min(PI($\overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m-1]$), PR($\overrightarrow{s}[m-1] \rightarrow \overrightarrow{s}[m]$)).

Algorithm 5: CalculateNeighborhood $(I(\overrightarrow{s}_i[m-1]), I(\overrightarrow{s}_j[m-1]))$.

Require: $I(\overrightarrow{s}_i[m-1])$ - a set of instances of event type $\overrightarrow{s}_i[m]$ of sequence $\overrightarrow{s}_i$, $I(\overrightarrow{s}_j[m-1])$ - a set of instances of event type $\overrightarrow{s}_j[m]$ of sequence $\overrightarrow{s}_j$.
Ensure: $I(\overrightarrow{s}[m])$ - a set of instances of event type $\overrightarrow{s}[m]$ of candidate sequence $\overrightarrow{s}$.
1: **return** $I(\overrightarrow{s}[m])$ as $distinct(\bigcup_{e \in I(\overrightarrow{s}_i[m-1])} N_{\overrightarrow{s}_j[m-1]}(e))$

neighborhoods of such instances with respect to event type $\overrightarrow{s}_j[m-1]$. Such neighborhoods can be calculated using the set $I(\overrightarrow{s}_j[m-1])$ rather than $D(\overrightarrow{s}_j[m-1])$ as it has been provided in Definition 2. In such a case, neighborhoods are calculated according to Eq. 7.

$$N_f(e) = \{e | p \in I(\overrightarrow{s}_j[m-1]) \wedge distance(p.location, e.location) \leq R \wedge (p.time - e.time) \in [0,T]\} \quad (7)$$

3.1 Discovering Significant Sequential Patterns using SPTree

To reduce candidate generation cost we propose to use a new tree structure: Sequence Pattern Tree (SP-Tree). Let us assume that the set of event types $F = \{A,B,C,D,E,F\}$ is given. Additionally, let us assume that the sets of significant patterns $\mathcal{L}$ are as presented in Table 1. The children of the root are all event types in F. Let us assume that the set of significant patterns of length 2 has been generated (for each event type adjoining all other event types in F and verifying participation ratios between them) as presented by level L_2 in SPTree in Figure 2. For each sequence we maintain three data structures: *firstParent*, *secondParent* and *children*.

Table 1: Examples of patterns.

F	A,B,C,D,E,F
$\mathcal{L}$	Patterns set
L_2	$A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, C \rightarrow F$
L_3	$A \rightarrow B \rightarrow C, A \rightarrow B \rightarrow D,$ $B \rightarrow C \rightarrow E, B \rightarrow C \rightarrow F$
L_4	$A \rightarrow B \rightarrow C \rightarrow E, A \rightarrow B \rightarrow C \rightarrow F$

- If sequence $\overrightarrow{s}$ has been created by joining event types f_{i_1} and f_{i_2} to $\overrightarrow{s} = f_{i_1} \rightarrow f_{i_2}$, then $firstParent(\overrightarrow{s}) := f_{i_1}$, $secondParent(\overrightarrow{s}) := f_{i_2}$ and $\overrightarrow{s}$ is added to $children(\overrightarrow{s})$.
- If sequence $\overrightarrow{s}$ has been created from sequences $\overrightarrow{s}_i$ and $\overrightarrow{s}_j$, then $firstParent(\overrightarrow{s}) := \overrightarrow{s}_i$, $secondParent(\overrightarrow{s}) := \overrightarrow{s}_j$ and add $\overrightarrow{s}$ to $children(\overrightarrow{s_i})$.

The procedure for generating candidate patterns of length m using SPTree is given in Algorithm 6. Algorithm 7 is used for the verification of candidates. Compared to Algorithm 2, Algorithm 7 removes $\overrightarrow{s}$ from the children list of the first parent of $\overrightarrow{s}$ if $PI(\overrightarrow{s})$ is less than θ.

Example 1 Let us consider the sequence $A \rightarrow B$ from Figure 2. Let us assume that the algorithm proceeds with the generation of candidates of length 3. In such a case, the sequence $A \rightarrow B$ can be extended with only

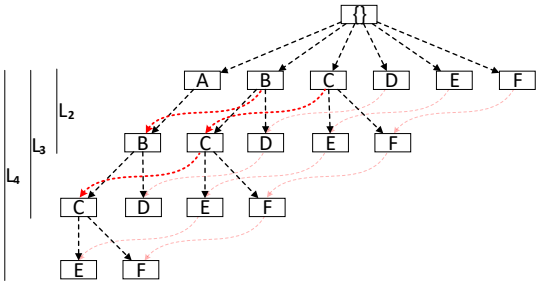

Figure 2: The SPTree created for patterns from Table 1.

two types: C or D, as the sequences $B \rightarrow C$ and $B \rightarrow D$ are children of the event type B being the second parent of $A \rightarrow B$.

4 DISCOVERING TOP-N MOST SIGNIFICANT SEQUENTIAL PATTERNS

An important data mining problem is to discover a set of N most significant patterns from a given dataset rather than all patterns with the significance measure greater than the user given threshold.

Definition 6 (A Top-N Sequential Pattern)**.** A sequential pattern $\overrightarrow{s}$ is the N-th top sequential pattern, if there exist $N-1$ patterns in the Top set with participation indexes equal or greater than $PI(\overrightarrow{s})$.

The method for verification of candidates while discovering Top-N most significant patterns for both naive and SPTree versions is given in Algorithms 8 and 9. In Algorithm 8 and 9, Top denotes the actual set of Top-N patterns, $|Top|$ is used to denote the actual number of patterns in this set and $Top(N)$ returns any top-N sequential pattern according to Definition 6. If participation indexes are greater or equal to the actual θ threshold, then the following scenarios are possible:

1. If the number of patterns in the Top set is less than $N-1$, then the candidate pattern is inserted into the Top and L_m sets.
2. If the number of patterns in the Top set is equal to $N-1$, then the candidate pattern is inserted into the Top and L_m sets and θ is raised to the value of participation index of the actual N-th top pattern.
3. If the number of patterns in the Top set is equal to N, then the candidate pattern is inserted into the Top and L_m sets and if the participation index of the candidate pattern is greater than the actual θ threshold, then θ is set to the value of participation index of the actual top N pattern and all the patterns with participation indexes less than θ are deleted from the Top and L_m sets.

Algorithms 8 and 9 may be used in the candidate verification phase in step 6 of Algorithm 1.

5 EXPERIMENTS

In the performed experiments, we compare our proposed algorithms (in both naive and SPTree versions) with STMiner proposed in (Huang et al., 2008). In comparison to our approach, STMiner utilizes depth first strategy for generating sequential patterns. Rather than using participation ratio and participation index as the significance measure of the discovered patterns, (Huang et al., 2008) proposes density ratio and sequence index for such purpose. Similarly to participation index, sequence index is the minimal density ratio between any two consecutive event types in a pattern. Contrary to the participation index, the sequence index preserves only the weak antimonotonicity property and due to that it can not be used with our strategy.

For the experimental evaluation we used the crime related dataset for Boston city for year 2014 (Boston-Police-Department, 2014). The first 2000 incidents extracted from the dataset are shown in Fig. 3. The whole dataset contains 40544 crime related incidents from 27 crime event types such as: SIMPLE ASSAULT, ARSON, assault with danger of life (AGGRAVATED ASSAULT), DISORDERLY, DRUG CHARGES, HARASSMENT. For the first set of experiments we compare execution times of STMiner ((Huang et al., 2008)) and the proposed algorithms NaiveSTBFM and SPTreeSTBFM. For our experiments we set the following parameters of the proposed algorithm: $R = 500$ meters, $T = 10080$ minutes (7 days).

In Figures 4 and 5, execution times of all three algorithms have been compared in a scenario discovering Top-N of the most important patterns (N is the number of such patterns) using Algorithms 8 and 9. In such a case, both SPTreeSTBFM and Naive STBFM execute 10 times faster than STMiner. A comparison of SPTreeSTBFM and NaiveSTBFM for Top-N patterns with large N using crime dataset containing 10000 instances is given in Figure 6. For large size of Top set (N = 1300), SPTreeSTBFM performs 5 times faster than NaiveSTBFM. For the STMiner algorithm presented in (Huang et al., 2008) we implemented a procedure for discovering Top-N patterns proposed in (Maciag, 2018).

In Figure 7, we compare execution times of SPTreeSTBFM and NaiveSTBFM when participation

Algorithm 6: CandidateGen(L_{m-1}).

Require: L_{m-1} - a set of significant sequential patterns of length $m-1$.
Ensure: C_m - a set of candidate sequential patterns of length m.

$C_m := \emptyset$.
for each $\overrightarrow{s}_i \in L_{m-1}$ **do**
  for each $\overrightarrow{s}_j \in children(secondParent(\overrightarrow{s}_i))$ **do**
    $\overrightarrow{s} := \overrightarrow{s}_i[1] \rightarrow \overrightarrow{s}_i[2] \rightarrow \cdots \rightarrow \overrightarrow{s}_i[m-1] \rightarrow \overrightarrow{s}_j[m-1]$.
    $I(\overrightarrow{s}[1]) := I(\overrightarrow{s}_i[1]) \wedge I(\overrightarrow{s}[2]) := I(\overrightarrow{s}_i[2]) \wedge \cdots \wedge I(\overrightarrow{s}[m-1]) := I(\overrightarrow{s}_i[m-1]) \wedge I(\overrightarrow{s}[m]) :=$ CalculateNeighborhood$(I(\overrightarrow{s}_i[m-1]), I(\overrightarrow{s}_j[m-1]))$.
    $firstParent(\overrightarrow{s}) := \overrightarrow{s}_i$, $secondParent(\overrightarrow{s}) := \overrightarrow{s}_j$.
    Add $\overrightarrow{s}$ to $children(\overrightarrow{s}_i)$, Add $\overrightarrow{s}$ to C_m.
  end for
end for
return C_m.

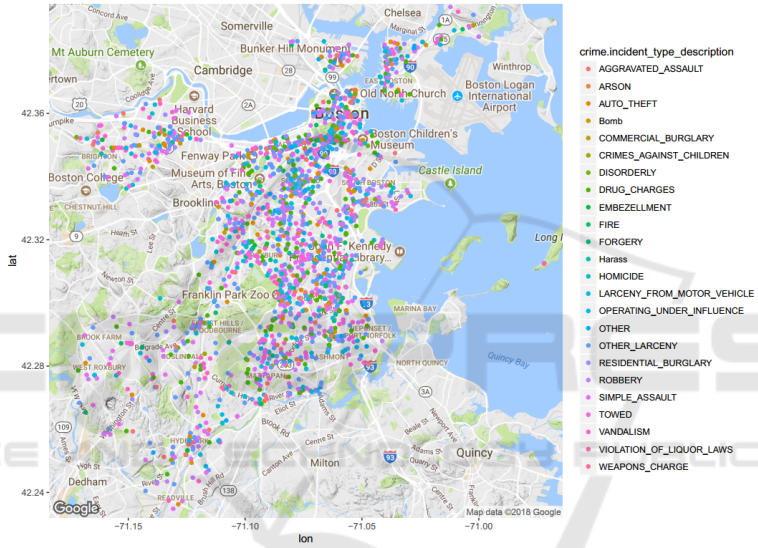

Figure 3: The first 2000 crime incidents and their types occurring in 2014 in Boston city extracted from the dataset used for experiments.

Algorithm 7: VerifyCandidates(C_m).

Require: C_m - a set of candidate sequential patterns of length m, θ - a significance threshold.
Ensure: L_m - a set of significant sequential patterns of length m.

$L_m := \emptyset$.
for each $\overrightarrow{s} \in C_m$ **do**
  CalculatePI($\overrightarrow{s}$).
  if PI($\overrightarrow{s}$) $\geq \theta$ **then**
    Add $\overrightarrow{s}$ to L_m.
  else
    Remove $\overrightarrow{s}$ from $children(firstParent(\overrightarrow{s}))$.
  end if
end for
return L_m.

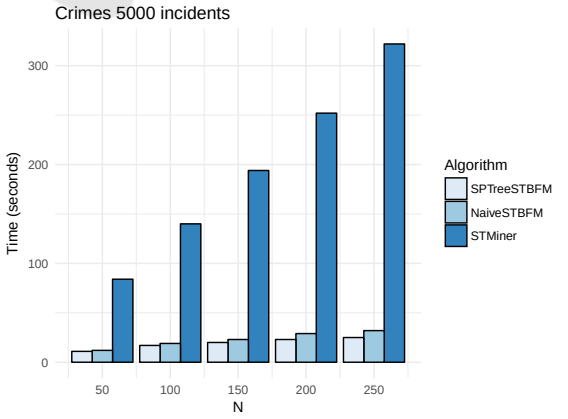

Figure 4: Execution times of STMiner, NaiveSTBFM and SPTreeSTBFM for discovering Top-N sequential patterns for crime dataset with 5000 instances.

Algorithm 8: Top-N-VerifyCandidatesNaive(C_m).

Require: C_m - a set of candidate sequential patterns of length m, θ - a significance threshold.

$L_m := \emptyset$.
for each $\overrightarrow{s} \in C_m$ **do**
  CalculatePI($\overrightarrow{s}$).
  if PI($\overrightarrow{s}$) $\geq \theta$ **then**
    if $|Top| < N-1$ **then**
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
    else if $|Top| = N-1$ **then**
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
      θ := PI($Top(N)$).
    else
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
      if PI($\overrightarrow{s}$) $> \theta$ **then**
        θ := PI($Top(N)$).
        Delete $\overrightarrow{s} \in Top$ with $PI(\overrightarrow{s}) < \theta$.
        Delete $\overrightarrow{s} \in L_m$ with $PI(\overrightarrow{s}) < \theta$.
      end if
    end if
  end if
end for
return L_m.

Algorithm 9: Top-N-VerifyCandidatesSPTree(C_m).

Require: C_m - a set of candidate sequential patterns of length m, θ - a significance threshold.

$L_m := \emptyset$.
for each $\overrightarrow{s} \in C_m$ **do**
  CalculatePI($\overrightarrow{s}$).
  if PI($\overrightarrow{s}$) $\geq \theta$ **then**
    if $|Top| < N-1$ **then**
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
    else if $|Top| = N-1$ **then**
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
      θ := PI($Top(N)$).
    else
      Add $\overrightarrow{s}$ to L_m, add $\overrightarrow{s}$ to Top.
      if PI($\overrightarrow{s}$) $> \theta$ **then**
        θ := PI($Top(N)$).
        Delete all $\overrightarrow{s} \in Top$ with $PI(\overrightarrow{s}) < \theta$.
        Delete all $\overrightarrow{s} \in L_m$ with $PI(\overrightarrow{s}) < \theta$;
        Delete $\overrightarrow{s}$ from $children(firstParent(\overrightarrow{s}))$.
      end if
    end if
  else
    Remove $\overrightarrow{s}$ from $children(firstParent(\overrightarrow{s}))$.
  end if
end for
return L_m.

index threshold is changing. For small θ threshold ($\theta < 0.25$), SPTreeSTBFM performs significantly better than NaiveSTBFM.

The most significant sequences obtained from our experiments are shown in Table 2. We can also conclude that the occurrence of SINGLE ASSAULT as well as OTHER LARCENY are very correlated to many other event types (their participation indexes

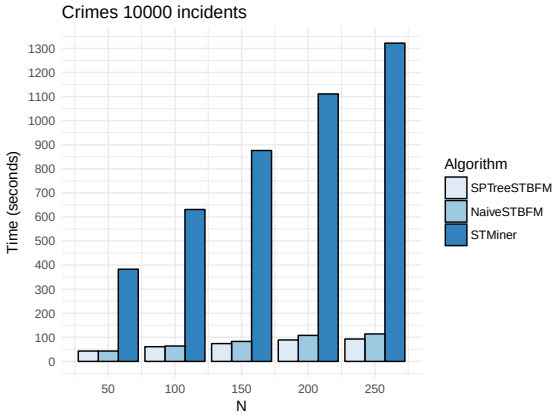

Figure 5: Execution times of STMiner, NaiveSTBFM and SPTreeSTBFM for discovering Top-N sequential patterns for crime dataset with 10000 instances.

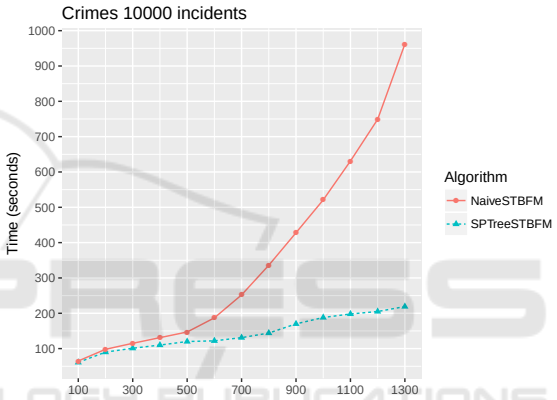

Figure 6: Comparing execution times of NaiveSTBFM and SPTreeSTBFM for discovering Top-N (with large N) for crime dataset with 10000 instances.

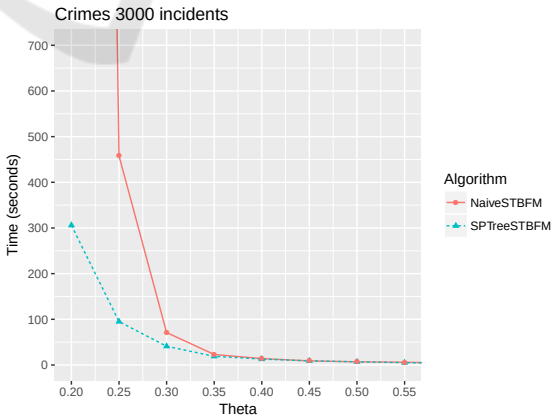

Figure 7: Execution times of NaiveSTBFM and SPTreeSTBFM with changing participation index threshold (theta).

when they precede other event types in the patterns are on average respectively: 0.715159 and 0.795544).

Table 2: Examples of patterns discovered in top-250 set.

Sequential pattern	PI
other_larceny → aggravated_assault → manslaug	1
simple_assault → other_larceny → arson	0.89
drug_charges → simple_assault → gambling_offense	0.83
simple_assault → harass	0.71
simple_assault → other_larceny → disorderly	0.71
vandalism → other_larceny → homicide	0.71
simple_assault → forgery	0.70
simple_assault → towed	0.70
simple_assault → violation_of_liquor_laws	0.63
drug_charges → other_larceny → operating_under_influence	0.63
drug_charges → aggravated_assault	0.59

6 CONCLUSIONS

We proposed a new algorithm called Spatio-Temporal Breadth-First Miner (STBFM) for discovering sequential patterns from spatio-temporal data. STBFM utilizes participation ratio and participation index as significance measures of discovered patterns and discovers all sequential patterns with participation indexes greater or equal to the user given threshold θ. STBFM proceeds with discovering patterns in two phases: candidate generation and candidate verification. For a faster generation of candidates we proposed a new structure: Sequential Pattern Tree (SP-Tree) and modification of STBFM utilizing this structure (SPTreeSTBFM). The proposed algorithms have been compared with the STMiner algorithm introduced in (Huang et al., 2008). As shown in the experimental results, the introduced algorithms are able to discover significant sequences about ten times faster than STMiner.

REFERENCES

Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., and Vitter, J. S. (1998). Scalable sweeping-based spatial join. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 570–581, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Boston-Police-Department (2014). Boston police department: Crime incident reports.

Feremans, L., Cule, B., and Goethals, B. (2018). Mining top-k quantile-based cohesive sequential patterns. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 90–98.

Han, J., Wang, J., Lu, Y., and Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 211–218.

Huang, Y., Zhang, L., and Zhang, P. (2008). A framework for mining sequential patterns from spatio-temporal event data sets. *IEEE Transactions on Knowledge and Data Engineering*, 20(4):433–448.

Lee, S. and Park, J. S. (2016). Top-k high utility itemset mining based on utility-list structures. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 101–108.

Maciag, P. S. (2018). Efficient discovery of top-k sequential patterns in event-based spatio-temporal data. In *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018.*, pages 47–56.

Mohan, P., Shekhar, S., Shine, J. A., and Rogers, J. P. (2012). Cascading spatio-temporal pattern discovery. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):1977–1992.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440.

Petitjean, F., Li, T., Tatti, N., and Webb, G. I. (2016). Skopus: Mining top-k sequential patterns under leverage. *Data Mining and Knowledge Discovery*, 30(5):1086–1111.

Tzvetkov, P., Yan, X., and Han, J. (2003). Tsp: mining top-k closed sequential patterns. In *Third IEEE International Conference on Data Mining*, pages 347–354.

Wu, C. W., Shie, B.-E., Tseng, V. S., and Yu, P. S. (2012). Mining top-k high utility itemsets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 78–86, New York, NY, USA. ACM.

Yin, J., Zheng, Z., Cao, L., Song, Y., and Wei, W. (2013). Efficiently mining top-k high utility sequential patterns. In *2013 IEEE 13th International Conference on Data Mining*, pages 1259–1264.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60.