Maciąg *et al. Journal of Big Data* (2023) 10:98
https://doi.org/10.1186/s40537-023-00780-x

Journal of Big Data

RESEARCH Open Access

Discovery of crime event sequences with constricted spatio-temporal sequential patterns

Piotr S. Maciąg[1*], Robert Bembenik[1] and Artur Dubrawski[2]

*Correspondence:
piotr.maciag@pw.edu.pl

[1] Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
[2] Auton Lab, The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, PA 15213 Pittsburgh, USA

Abstract

In this article, we introduce a novel type of spatio-temporal sequential patterns called Constricted Spatio-Temporal Sequential (CSTS) patterns and thoroughly analyze their properties. We demonstrate that the set of CSTS patterns is a concise representation of all spatio-temporal sequential patterns that can be discovered in a given dataset. To measure significance of the discovered CSTS patterns we adapt the participation index measure. We also provide *CSTS-Miner*: an algorithm that discovers all participation index strong CSTS patterns in event data. We experimentally evaluate the proposed algorithms using two crime-related datasets: Pittsburgh Police Incident Blotter Dataset and Boston Crime Incident Reports Dataset. In the experiments, the CSTS-Miner algorithm is compared with the other four state-of-the-art algorithms: STS-Miner, CSTPM, STBFM and CST-SPMiner. As the results of the experiments suggest, the proposed algorithm discovers much fewer patterns than the other selected algorithms. Finally, we provide the examples of interesting crime-related patterns discovered by the proposed CSTS-Miner algorithm.

Keywords: Data mining, Spatio-temporal sequential patterns, Crime-data analysis, Patterns discovery, Concise representation

Introduction

Discovering knowledge in the form of various types of patterns, inference rules or motifs from spatio-temporal events data is a topic attracting increasing attention of researchers world-wide [1–3]. Specifically, many real-world spatio-temporal datasets consist of a set of event types and their event instances defined by geographical locations and occurrence times. An example of a spatio-temporal event dataset is a set of crime event incidents, such as arson, homicide or vandalism, each of which is assigned an event type, a geographical location and time of occurrence. Discovering sequences of crime types that occur in a spatial area over a time period can contribute to a better understanding of the causes of these crimes and to their elimination [4–8].

In order to discover such sequences of spatio-temporal event types, one can consider applying one of the algorithms for spatio-temporal sequential patterns discovery.

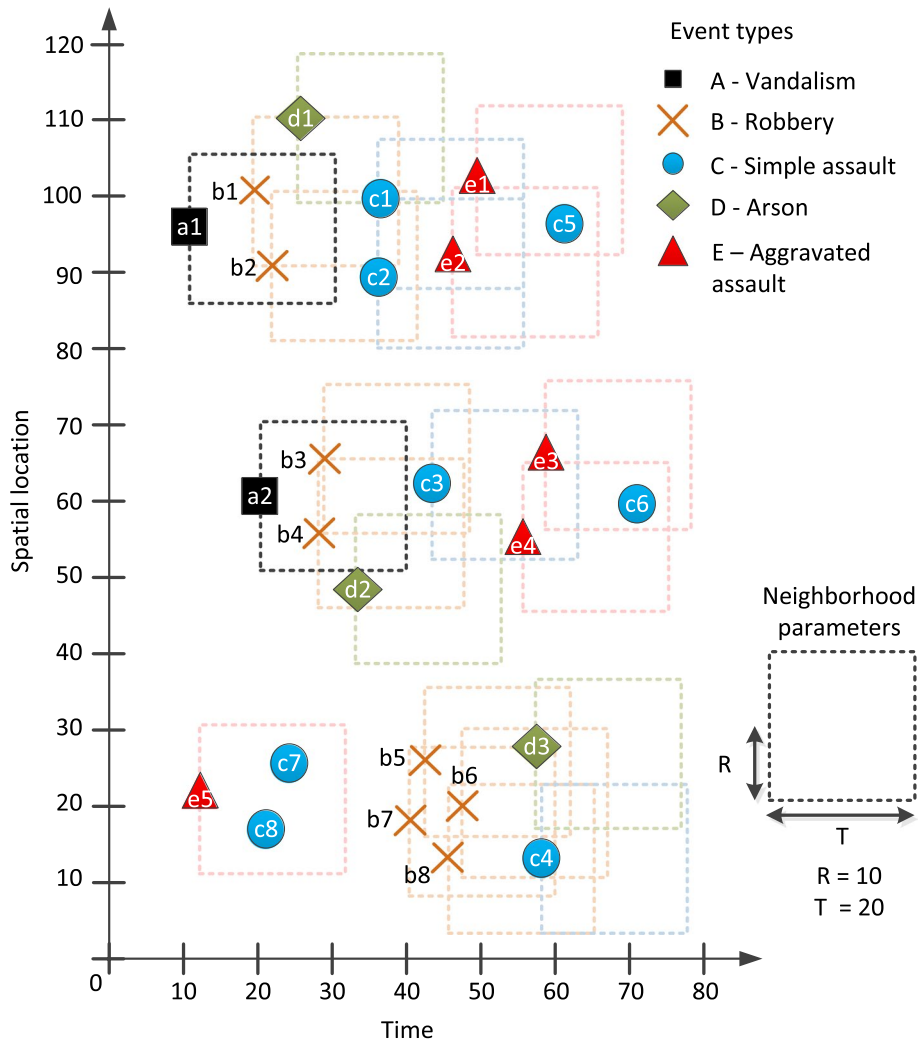

Fig. 1 The example of a spatio-temporal event dataset. The figure also presents neighborhood specification of event instances

A spatio-temporal sequential pattern (in brief, ST sequential pattern), introduced in [9], is defined as a sequence of event types. By discovering ST sequential patterns, one can obtain insight into spatio-temporal relations between various event types. For example, the discovery of an ST sequential pattern *arson* $\rightarrow$ *vandalism* $\rightarrow$ *bomb* can lead to the critical behavior pattern of a dangerous terrorist. The example of another ST sequential pattern that could be discovered from the dataset presented in Fig. 1 is $A \rightarrow B \rightarrow C \rightarrow D$.

Limitations of the existing work

In the literature, several types of methods and algorithms were already developed for the discovery of ST sequential patterns (see, for example, [9–14]). [9] introduced the first algorithm called STS-Miner for the discovery of significant ST sequential patterns. [11, 12] define a significant ST sequential pattern as a pattern, whose participation index

measure is greater than the user-specified threshold PI_{min}. [12] refers to such a pattern as a PI-strong ST sequential pattern (we adapt this naming convention in this paper).

While the already proposed algorithms (such as STS-Miner [9], STBFM [11], CST-SPMiner [12], CSTPM [10], STES [15]) can discover PI-strong (closed) ST sequential patterns in some datasets, however, in practice, the number of discovered redundant patterns can still be too huge to be analyzed by a user of the algorithm. Hence, in this paper, we offer a novel representation of ST sequential patterns which we call *Constricted Spatio-Temporal Sequential patterns* (in brief, CSTS patterns) and analyze their theoretical properties. As presented in the paper, given the set of CSTS patterns one can approximate participation index values of all ST sequential patterns.

To verify the efficiency and effectiveness of the proposed approach, we use two real-world datasets of crime events: the Pittsburgh Police Incident Blotter Dataset and the Boston Crime Incident Reports Dataset. For example, one of the conducted experiments shows that for the Boston Crime Incidents Reports dataset, the proposed approach is able to discover 65,967 CSTS patterns, while the three algorithms discovering all spatio-temporal sequential patterns provide as many as 228,285 patterns. The discovered 65,967 CSTS patterns can be used to derive all 228,285 spatio-temporal sequential patterns and approximate participation index of each of them with a maximal error of $\pm$ 0.025.

Contributions

The contributions of the paper are as follows:

- We introduce the notion of **a Constricted Spatio-Temporal Sequential (CSTS) pattern** that constitutes concise representation of all ST sequential patterns.
- We thoroughly **analyze theoretical properties of CSTS patterns**. Specifically, we show that the set of CSTS patterns is a subset of the set of closed ST sequential patterns and that each CSTS pattern is also a closed ST sequential pattern. Moreover, we show that given the set of Participation Index (PI-)strong CSTS patterns one can obtain the set of all PI-strong ST sequential patterns and approximate participation index of each of them with an approximation margin of $\pm\ \varepsilon$.
- We offer **a new algorithm called CSTS-Miner** that discovers PI-strong CSTS patterns. CSTS-Miner applies **an introduced MAX-Tree structure** for more efficient candidate patterns generation. The proposed MAX-Tree is generated in two main phases of CSTS-Miner: the first phase called "top-down", in which all PI-strong ST sequential patterns are generated using the breadth-first strategy, and the second phase called "bottom-up", which calculates PI-strong CSTS patterns. We also offer the analysis of **computational and memory complexity** of CSTS-Miner.
- **We experimentally compare the results obtained with the CSTS-Miner algorithm with three other state-of-the-art algorithms** discovering ST sequential patterns: the adapted version of STS-Miner [9], STBFM [11], CSTPM [10], which discover PI-strong ST sequential patterns. Moreover, we also compare CSTS-Miner with the CST-SPMiner algorithm [12], which discovers PI-strong closed ST sequential patterns. Similarly to our CSTS patterns, closed spatio-temporal sequential patterns discovered by CST-SPMiner also constitute a concise representation of all ST sequential

patterns. For the purpose of experiments, we selected and preprocessed two crime events datasets: the Pittsburgh Police Incident Blotter Dataset and the Boston Crime Incident Reports Dataset. As we show, CSTS-Miner discovers much fewer redundant patterns than the other selected algorithms. Specifically, as the results of the experiments confirm, CSTS-Miner provides up to 60% fewer patterns compared to the STS-Miner, STBFM and CSTPM algorithms and up to 40% fewer patterns compared to the CST-SPMiner algorithm.

- We **provide experimental comparison of the effectiveness and efficiency of the above-mentioned STS-Miner, CSTPM, STBFM and CST-SPMiner algorithms** discovering (closed) spatio-temporal sequential patterns. In our comparison, we showed the number of discovered patterns and execution time of each of algorithms for the same parameters of participation index threshold and neighborhood specification. Our implementations (prepared in the C++ language) of the selected algorithms (STS-Miner, STBFM, CSTPM, CST-SPMiner) as well as the proposed CSTS-Miner algorithm are available at the GitHub repository.[1]
- Eventually, **we provide examples of interesting crime-related patterns discovered by CSTS-Miner** from the Pittsburgh Police Incident Blotter Dataset.

Structure

The structure of the article is as follows. In Sect. Related work, we offer a brief review of the related work. Section Basic notions offers basic notions of ST sequential patterns. In Sect. Discovery of constricted ST sequential patterns, we introduce the notion of a CSTS pattern. In Sect. Theoretical properties of CSTS patterns, we analyze theoretical properties of CSTS patterns. Section Constricted ST sequential patterns miner describes the proposed CSTS-Miner algorithm. In Sect. Experiments, we provide the results of experiments and in Sect. Conclusion we conclude the article.

Related work

The discovery of concise representations of various patterns (especially frequent patterns and sequential patterns) attracts researchers' attention. In [16], *closed sequential patterns* representation was introduced for the first time. Following [16], numerous works were dedicated to the problem of more efficient mining of closed sequential patterns. The examples include methods and algorithms offered in [17, 18] or, more recently, in [19] and [20]. Other related research directions include discovery of top closed sequential patterns with the highest support [21, 22] and the parallel discovery of closed sequential patterns [23]. A survey of the current methods for the discovery of closed sequential patterns can be found in [24].

Our proposed notion of CSTS patterns is similar to the notion of delta closed sequential patterns [25] in that they both avoid returning patterns whose significance measure is no greater than the significance measure of their supersequence patterns plus the approximation margin (which in the case of delta closed patterns is called δ-tolerance).

[1] https://github.com/piotrMaciag32/CSTS-Miner.

However, unlike our CSTS patterns, delta closed sequential patterns are not designed to work with spatio-temporal event data.

While many methods and algorithms were offered to discover various types of spatio-temporal patterns, relatively few of them focused on discovering ST sequential patterns. To this end, in our experiments the proposed CSTS-Miner algorithm is compared with the most representative algorithms mining ST sequential patterns, namely:

- **STS-Miner** [9]—the first algorithm offered for the discovery of ST sequential patterns. STS-Miner uses the depth-first patterns generation strategy. In this work, we adapted STS-Miner to use the participation index measure instead of the sequence index measure introduced in [9]. Thus, the adapted version of STS-Miner is capable of discovering PI-strong ST sequential patterns.
- **STBFM** [11]—the algorithm discovering PI-strong ST sequential patterns by means of the breadth-first pattern generation strategy. Maciąg and Bembenik [11] introduced a structure called SP-Tree that allows to efficiently generate candidate patterns using their first and second parent patterns and a children list of the first parent pattern. In addition, [11] presented the two variations of STBFM that can discover top-K PI-strong ST sequential patterns. The experiments of [11] showed that STBFM is capable of discovering some interesting crime-related patterns.
- **CST-SPMiner** [12]—the algorithm discovering closed PI-strong ST sequential patterns. CST-SPMiner applies the breadth-first candidate patterns generation strategy to obtain all PI-strong closed ST sequential patterns. For each obtained PI-strong ST sequential pattern $\overrightarrow{s}$, CST-SPMiner determines if this pattern is closed or not using a check condition verifying if $\overrightarrow{s}$ is a closure pattern of any of its parent patterns.
- **CSTPM** [10]—the algorithm discovering Cascading Spatio-Temporal Patterns (CSTP). CSTP patterns consist not only of ST sequential patterns but also of cascades of event types. In our work, to directly compare CSTPM with the proposed CSTS-Miner, we adapted the CSTPM algorithm for the discovery of ST sequential patterns rather than cascading ST patterns.

The reviews of methods and algorithms for (spatio-temporal) patterns discovery with particular emphasis on spatio-temporal event datasets can be found in [15, 26–33].

Basic notions

The definitions presented in this section are formulated based on the works [9, 11, 12].

ST sequential patterns

Let $\mathbf{F}$ denote a set of n event types and $\mathbf{D}$ denote a dataset of event instances in which each $e \in \mathbf{D}$ consists of a spatial location, an occurrence time (timestamp) and an event type $F \in \mathbf{F}$. $\mathbf{D}$ will be called a *spatio-temporal event dataset*. Moreover, let $|\mathbf{D}|$ denote the number of event instances in $\mathbf{D}$. The set of all event instances of type F in dataset $\mathbf{D}$ will be denoted by $\mathbf{D}(F)$.

Spatio-temporal event datasets often occur in real-world. The possible examples include: a dataset of crime instances and crime event types or a dataset of conflict incidents and their types. Let us consider an example of a spatio-temporal event dataset presented in Fig. 1. This dataset consists of event instances $\mathbf{D} = \{a_1, a_2, b_1, \ldots, b_8, c_1, \ldots, c_8, d_1, \ldots, d_3, e_1, \ldots, e_5\}$ and a set of five event types $\mathbf{F} = \{A, B, C, D, E\}$.[2] To better illustrate the notions introduced in this section, let us assign the set of event types $\mathbf{F} = \{A, B, C, D, E\}$ of the dataset presented in Fig. 1 real-world crime event types as follows:

- A—Vandalism,
- B—Robbery,
- C—Simple assault,
- D—Arson,
- E—Aggravated assault.

Spatio-temporal sequential pattern (ST sequential pattern) $\overrightarrow{s}$ is defined as a sequence of elements, each of which is an event type from $\mathbf{F}$ [9]. Please note that an event type $F \in \mathbf{F}$ can occur multiple times in an ST sequential pattern $\overrightarrow{s}$[3]. Before we provide a formal definition of an ST sequential pattern, we recall the definition of a spatio-temporal neighborhood of an event instance with respect to an event type:

Definition 1 (Neighborhood of an event instance with respect to an event type [9, 12]) For an event instance e, the *neighborhood of e with respect to an event type* $F \in \mathbf{F}$ is denoted by $\mathbf{N}(e, F)$ and is defined as follows:

$$\begin{aligned}\mathbf{N}(e, F) = \{p \mid p \in \mathbf{D}(F) \\ \wedge\, distance(p.location, e.location) \leq R \\ \wedge\, (p.time - e.time) \in (0, T]\},\end{aligned} \tag{1}$$

where R and T are user-given spatial distance and time window thresholds, respectively.

In Fig. 1, the neighborhood $\mathbf{N}(a_1, B)$ consists of event instances $\{b_1, b_2\}$. (We would say that the neighborhood of a Vandalism crime event a_1 contains with respect to event type B two crime events of Robbery b_1, b_2). Similarly, the neighborhood $\mathbf{N}(c_1, E) = \{e_1, e_2\}$ (thus, crime instance c_1 of Simple assault contains with respect to Aggravated assault two crime events e_1, e_2).

The spatial distance threshold of neighborhoods given in Fig. 1 is $R = 10$, while the time window threshold is equal to $T = 20$. In our experiments, the spatial distance between locations of two event instances is calculated as the Euclidean distance between these locations. Please note that according to Definition 1, an event instance e_j can be located in a neighborhood of an event instance e_i, only if the occurrence time of e_j is

[2] Please note that the spatial location of each event instance presented in Fig. 1 is for simplicity specified using only one dimension. However, in real datasets, the spatial location of event instances is usually defined by coordinates of two dimensions (for example, in the datasets selected for the experiments, spatial location is defined using longitude and latitude coordinates).

[3] An example of such an ST sequential pattern to be discovered for the dataset in Fig. 1 is $A \rightarrow B \rightarrow C \rightarrow E \rightarrow C$ (or *Vandalism* → *Robbery* → *Simple assault* → *Aggravated assault* → *Simple assault*).

greater than the occurrence time of e_i (i.e., the two event instances with the same occurrence time can not mutually belong to their neighborhoods because the difference between their occurrence time would be $e_i.time - e_j.time = 0$).

Definition 2 (Spatio-temporal sequential pattern) A *spatio-temporal sequential pattern* (in brief, *ST sequential pattern*) is a sequence of event types in $\mathbf{F}$. i-th element of sequence $\overrightarrow{s}$ is denoted by $\overrightarrow{s}[i]$. Sequence $\overrightarrow{s}$ which consists of m elements is denoted as $\overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m]$. The number of elements of sequence $\overrightarrow{s}$ is defined as its length.

An example of an ST sequential pattern for the dataset presented in Fig. 1 is $\overrightarrow{s} = A \rightarrow B \rightarrow C$, the length of which is 3. The important question is how to efficiently calculate neighborhoods of event instances. In our implementation, we adapted the computationally efficient plane sweep algorithm (see e.g. [34]).

Definition 3 (Set of event instances supporting an element of an ST sequential pattern [9, 12]) *A Set of event instances supporting i-th element of ST sequential pattern* $\overrightarrow{s}$ is denoted by $\mathbf{I}(\overrightarrow{s}, i)$ and is defined as follows:

$$\mathbf{I}(\overrightarrow{s}, i) = \begin{cases} \mathbf{D}(\overrightarrow{s}[1]) & \text{when } i = 1, \\ \bigcup\limits_{e \in \mathbf{I}(\overrightarrow{s}, i-1)} \mathbf{N}(e, \overrightarrow{s}[i]) & \text{when } i > 1. \end{cases} \tag{2}$$

For each ST sequential pattern $\overrightarrow{s}$, we can unambiguously distinguish sets of event instances supporting elements of that pattern. For the first element of a pattern, the set of event instances $\mathbf{I}(\overrightarrow{s}, 1)$ supporting that element is defined simply as all event instances of event type $\overrightarrow{s}[1]$ in $\mathbf{D}$. For every next element of $\overrightarrow{s}$ (say i), the set of supporting event instances $\mathbf{I}(\overrightarrow{s}, i)$ consists of all those event instances of event type $\overrightarrow{s}[i]$ which belong to neighborhoods of instances contained in the supporting set $\mathbf{I}(\overrightarrow{s}, i-1)$.

Let us consider an example of a previously given ST sequential pattern $\overrightarrow{s} = A \rightarrow B \rightarrow C$ (*Vandalism* $\rightarrow$ *Robbery* $\rightarrow$ *Simple assault*) of the dataset in Fig. 1. The sets of event instances supporting $\overrightarrow{s}$ are as follows:

- $\mathbf{I}(\overrightarrow{s}, 1) = \mathbf{D}(A) = \{a_1, a_2\}$;
- $\mathbf{I}(\overrightarrow{s}, 2) = \bigcup\limits_{a \in \mathbf{I}(\overrightarrow{s}, 1)} \mathbf{N}(a, B) = \{b_1, b_2, b_3, b_4\}$;
- $\mathbf{I}(\overrightarrow{s}, 3) = \bigcup\limits_{b \in \mathbf{I}(\overrightarrow{s}, 2)} \mathbf{N}(b, C) = \{c_1, c_2, c_3\}$;

In this article, we apply the previously introduced in [10, 11, 15] *participation ratio* and *participation index* measures of significance of discovered patterns. The participation ratio of an i-th element of an ST sequential pattern $\overrightarrow{s}$ expresses the quotient

of the number of event instances supporting i-th element of $\overrightarrow{s}$ to the number of event instances of $\overrightarrow{s}[i]$ event type in **D**.

Definition 4 (Participation Ratio (PR) and Participation Index (PI) [12]) The participation ratio of an i-th element of ST sequential pattern $\overrightarrow{s}$, where $i \geq 1$, is denoted by $PR(\overrightarrow{s}, i)$ and is defined as the ratio of the cardinality of the set of event instances supporting i-th element of $\overrightarrow{s}$ to the number of all instances of type $\overrightarrow{s}[i]$ in the dataset **D**; that is: $PR(\overrightarrow{s}, i) = \frac{\left|I(\overrightarrow{s}, i)\right|}{\left|\mathbf{D}(\overrightarrow{s}[i])\right|}$.

The *participation index of ST sequential pattern* $\overrightarrow{s} = \overrightarrow{s}[1] \rightarrow \overrightarrow{s}[2] \rightarrow \cdots \rightarrow \overrightarrow{s}[m]$ is denoted by $PI(\overrightarrow{s})$ and is defined as the minimum from the participation ratios of all elements of $\overrightarrow{s}$; that is, $PI(\overrightarrow{s}) = \min\big(\{PR(\overrightarrow{s}, i) \mid i = 1, 2, \ldots, m\}\big)$.

By Definition 4, the value of participation ratio is always in the range [0,1]. Participation index is defined as the minimum of participation ratios of all elements of an ST sequential pattern.

Let us consider the pattern $\overrightarrow{s} = A \rightarrow B \rightarrow C$ (*Vandalism* $\rightarrow$ *Robbery* $\rightarrow$ *Simple assault*) and let us calculate PR and PI values of this pattern given the dataset and neighborhoods parameters of Fig. 1.

- $PR(\overrightarrow{s}, 1) = 1$,
- $PR(\overrightarrow{s}, 2) = \frac{4}{8} = 0.5$ (that is half of all Robbery event instances occur in neighborhoods of Vandalism event instances),
- $PR(\overrightarrow{s}, 3) = \frac{3}{8} = 0.375$ (only three instances of Simple assault event type occur in neighborhoods of event instances of the set $\mathbf{I}(\overrightarrow{s}, 2)$.

Thus, the participation index of $\overrightarrow{s} = A \rightarrow B \rightarrow C$ equals to $PI(\overrightarrow{s}) = 0.375$.

In Table 1, we listed all ST sequential patterns whose participation indexes are greater than 0 that can be discovered in the dataset of Fig. 1.

Definition 5 (PI-strong ST sequential pattern) A candidate ST sequential pattern $\overrightarrow{s}$ is called PI-strong if its participation index $PI(\overrightarrow{s})$ is greater than the participation index threshold PI_{min}.

Table 1 All ST sequential patterns and their participation indexes (provided in parentheses next to the patterns) for the dataset of Fig. 1

Pattern length	Patterns set
$L_1(F)$	$A(1), B(1), C(1), D(1), E(1)$
L_2	$A \rightarrow B(0.5), B \rightarrow B(0.625), B \rightarrow C(0.5), B \rightarrow D(1), C \rightarrow E(0.8), E \rightarrow C(0.5)$
L_3	$A \rightarrow B \rightarrow B(0.25), A \rightarrow B \rightarrow C(0.375), A \rightarrow B \rightarrow D(0.5), B \rightarrow B \rightarrow C(0.5), B \rightarrow B \rightarrow D(0.33),$ $B \rightarrow C \rightarrow E(0.375), C \rightarrow E \rightarrow C(0.25)$
L_4	$A \rightarrow B \rightarrow B \rightarrow C(0.25), A \rightarrow B \rightarrow C \rightarrow E(0.375), B \rightarrow B \rightarrow C \rightarrow E(0.375),$ $B \rightarrow C \rightarrow E \rightarrow C(0.25)$
L_5	$A \rightarrow B \rightarrow B \rightarrow C \rightarrow E(0.25), A \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25), B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$
L_6	$A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$

Discovery of all PI-strong ST sequential patterns can be performed, for example, using the STBFM algorithm introduced in [11].

Let us assume that $PI_{min} = 0.5$. The set of all PI-strong ST sequential patterns for the dataset of Fig. 1 is: $A(1)$, $B(1)$, $C(1)$, $D(1)$, $E(1)$, $B \rightarrow B(0.625)$, $B \rightarrow D(1)$, $C \rightarrow E(0.8)$.

Definition 6 Let $\overrightarrow{s_1} = \overrightarrow{s_1}[1] \rightarrow \overrightarrow{s_1}[2] \rightarrow \cdots \rightarrow \overrightarrow{s_1}[m_1]$ and $\overrightarrow{s_2} = \overrightarrow{s_2}[1] \rightarrow \overrightarrow{s_2}[2] \rightarrow \cdots \rightarrow \overrightarrow{s_2}[m_2]$ be ST sequential patterns. $\overrightarrow{s_1}$ is a subsequence of $\overrightarrow{s_2}$ and $\overrightarrow{s_2}$ is a supersequence of $\overrightarrow{s_1}$ if $m_1 \leq m_2$ and there exists an integer k, where $0 \leq k \leq m_2 - m_1$, such that $\overrightarrow{s_1}[1] = \overrightarrow{s_2}[1+k] \wedge \overrightarrow{s_1}[2] = \overrightarrow{s_2}[2+k] \wedge \cdots \wedge \overrightarrow{s_1}[m_1] = \overrightarrow{s_2}[m_1+k]$.

If $m_1 < m_2$, then $\overrightarrow{s_1}$ is a proper subsequence of $\overrightarrow{s_2}$ and $\overrightarrow{s_2}$ is a proper supersequence of $\overrightarrow{s_1}$.

Theorem 1 (Anti-monotonicity property of the participation index for supersequences [12]) *Let $\overrightarrow{s_1}$ and $\overrightarrow{s_2}$ be ST sequential patterns. If $\overrightarrow{s_1}$ is a subsequence of $\overrightarrow{s_2}$, then $PI(\overrightarrow{s_1}) \geq PI(\overrightarrow{s_2})$.*[4]

For the dataset presented in Fig. 1, $\overrightarrow{s_1} = A \rightarrow B$ is a proper subsequence of $\overrightarrow{s_2} = A \rightarrow B \rightarrow C \rightarrow E$ ($\overrightarrow{s_2}$ is a proper supersequence of $\overrightarrow{s_1}$).

As follows from Theorem 1, the *PI* value of ST sequential pattern $\overrightarrow{s_2}$ is always less than or equal to the *PI* value of any of its proper subsequence $\overrightarrow{s_1}$. The STBFM, CST-SPMiner and CSTPM algorithms apply Theorem 1 to efficiently generate candidate ST sequential patterns using the breadth-first search strategy.

Closed ST sequential patterns

Maciag 12] introduced a concise and lossless representation of ST sequential patterns called **closed ST sequential patterns**. The important property of closed ST sequential patterns is that one can obtain the value of participation index of any ST sequential pattern given only the set of all closed ST sequential patterns. In Sect. "Discovery of constricted ST sequential patterns" of this work, we theoretically and experimentally compare the proposed constricted ST sequential patterns to the closed ST sequential patterns. Thus, Definition 7 recalls the notions of closed ST sequential pattern, closure of an ST sequential pattern and PI-strong closed ST sequential pattern.

Definition 7 (Closed ST sequential pattern and closure of an ST sequential pattern [12]) ST sequential pattern $\overrightarrow{s_1}$ is closed if there exists no proper supersequence $\overrightarrow{s_2}$ of $\overrightarrow{s_1}$, such that the participation index $PI(\overrightarrow{s_2}) = PI(\overrightarrow{s_1})$.

A closure of ST sequential pattern $\overrightarrow{s_1}$ is a supersequence $\overrightarrow{s_2}$ of $\overrightarrow{s_1}$, such that $\overrightarrow{s_2}$ is a closed ST sequential pattern and $PI(\overrightarrow{s_2}) = PI(\overrightarrow{s_1})$.

A PI-strong closed ST sequential pattern is a closed ST sequential pattern whose participation index is greater than the threshold PI_{min}.

[4] We refer the reader to [12] for the proof of the theorem.

For example, for the set of ST sequential patterns of Table 1, $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$ is a closed ST sequential pattern. $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$ is also a closure of the following patterns:

- $A \rightarrow B \rightarrow B(0.25)$,
- $C \rightarrow E \rightarrow C(0.25)$,
- $A \rightarrow B \rightarrow B \rightarrow C(0.25)$,
- $B \rightarrow C \rightarrow E \rightarrow C(0.25)$,
- $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E(0.25)$,
- $A \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$,
- $B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$.

An ST sequential pattern can be closed and be its own closure. For example, $B \rightarrow B(0.625)$ is a closed ST sequential pattern and it is also its own closure. Please note that according to Definition 7 an ST sequential pattern can have more than one closure. For example, pattern $B \rightarrow C \rightarrow E(0.375)$ of the dataset in Fig. 1 has two closures:

- $A \rightarrow B \rightarrow C \rightarrow E(0.375)$,
- $B \rightarrow B \rightarrow C \rightarrow E(0.375)$.

Similarly to Table 1, in Table 2, we provide the set of all closed ST sequential patterns that can be discovered from the dataset of Fig. 1.

Discovery of constricted ST sequential patterns

In this section, we first present our motivation for introducing Constricted ST Sequential patterns. Next, we provide the elementary notions of such type of patterns.

Motivation

Closed ST sequential patterns of Definition 7 are a concise representation of all ST sequential patterns: hence, the set of closed ST sequential patterns can be used to derive all ST sequential patterns. However, in the case of many real-world spatio-temporal event data, participation indexes of ST sequential patterns strictly depend on the specification of the neighbourhoods of event instances as well as spatial and temporal distribution of the locations of event instances in the dataset **D**. Specifically, for a given ST sequential pattern $\overrightarrow{s}$ of length k, its proper supersequence patterns of length greater

Table 2 All closed ST sequential patterns and their participation indexes (provided in parentheses next to the patterns) for the dataset of Fig. 1

Pattern length	Patterns set
$L_1(F)$	$A(1), C(1), E(1)$
L_2	$B \rightarrow B(0.625), B \rightarrow D(1), C \rightarrow E(0.8), E \rightarrow C(0.5)$
L_3	$A \rightarrow B \rightarrow D(0.5), B \rightarrow B \rightarrow C(0.5), B \rightarrow B \rightarrow D(0.33)$
L_4	$A \rightarrow B \rightarrow C \rightarrow E(0.375), B \rightarrow B \rightarrow C \rightarrow E(0.375)$
L_5	$A \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$
L_6	$A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$

than k usually have only slightly smaller values of participation indexes than $\overrightarrow{s}$. According to Definition 7, none of such supersequences can constitute a closure of $\overrightarrow{s}$ and thus be a closed ST sequential pattern.

To illustrate such a situation, let us consider pattern $\overrightarrow{s_1} = B \rightarrow B \rightarrow C(0.5)$ from Table 1. As follows from Table 2, pattern $B \rightarrow B \rightarrow C(0.5)$ is a closed pattern (and thus as follows from Definition 7 it is its own closure). Let us now consider pattern $\overrightarrow{s_2} = B \rightarrow B \rightarrow C \rightarrow E(0.375)$. $\overrightarrow{s_2}$ is a proper supersequence of $\overrightarrow{s_1}$ (as follows from Definition 6). However, because PI value of $\overrightarrow{s_2}$ is less than PI value $\overrightarrow{s_1}$, $\overrightarrow{s_2}$ cannot be a potential closure of $\overrightarrow{s_1}$ despite the fact that the difference between participation indices of both is only 0.75.

Nevertheless, providing only supersequences of $\overrightarrow{s}$ can often allow us to approximate the participation index of $\overrightarrow{s}$.

Hence, in this paper, we offer a notion of a constricted ST sequential pattern. We define a constricted ST sequential pattern $\overrightarrow{s_1}$ as such a maximal (that is the longest) supersequence of pattern $\overrightarrow{s}$ for which (i) the difference between participation indexes of $\overrightarrow{s}$ and $\overrightarrow{s_1}$ is minimal and (ii) participation index of $\overrightarrow{s_1}$ is greater than or equal to the participation index of $\overrightarrow{s}$ minus approximation margin ε[5]. We show in Sect. "Theoretical properties of CSTS patterns" that given a set of PI-strong constricted ST sequential patterns, one can obtain a set of all PI-strong ST sequential patterns and approximate participation indexes of each of them with an approximation margin $\pm\ \varepsilon$.

Elementary notions

Let us begin with the definitions of a maximal supersequence of an ST sequential pattern and a minimal proper supersequence of an ST sequential pattern.

Definition 8 (Maximal supersequence of an ST sequential pattern) For an ST sequential pattern $\overrightarrow{s_1}$ of a dataset **D**, its maximal ST supersequnce pattern $\overrightarrow{s_2}$ is such a supersequence of $\overrightarrow{s_1}$ whose length is the greatest.

Please note that $\overrightarrow{s_1}$ can have more than one maximal ST supersequence pattern.

Definition 9 (Minimal proper supersequence of an ST sequential pattern) For an ST sequential pattern $\overrightarrow{s_1}$ of a dataset **D**, its minimal proper supersequnce pattern $\overrightarrow{s_2}$ is such a proper supersequence of $\overrightarrow{s_1}$ whose length is the smallest.

Please note that $\overrightarrow{s_1}$ can have more than one minimal proper ST supersequence patterns.

Let us consider the set of all supersequences of the pattern $\overrightarrow{s_1} = B \rightarrow B$ presented in Table 1:

- $B \rightarrow B$,
- $A \rightarrow B \rightarrow B$,

[5] As we show in Sect. 5, for approximation margin $\varepsilon = 0$, the proposed notion of a constricted ST sequential pattern is equivalent to the notion of a closed ST sequential pattern.

- $B \rightarrow B \rightarrow C$,
- $B \rightarrow B \rightarrow D$,
- $A \rightarrow B \rightarrow B \rightarrow C$,
- $B \rightarrow B \rightarrow C \rightarrow E$,
- $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E$,
- $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C$.

The maximal supersequence of the pattern $\overrightarrow{s_1}$ is $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C$ and the minimal proper supersequences of $\overrightarrow{s_1}$ are $A \rightarrow B \rightarrow B, B \rightarrow B \rightarrow C, B \rightarrow B \rightarrow D$.

Definition 10 (ε-constricted maximal supersequence of an ST sequential pattern) ε -constricted maximal supersequence $\overrightarrow{s_1}$ of an ST sequential pattern $\overrightarrow{s}$ (in brief, Constricted ST Sequential pattern, CSTS pattern) is such a supersequence of $\overrightarrow{s}$ which preserves the two conditions:

1 $PI(\overrightarrow{s_1}) \geq PI(\overrightarrow{s}) - \varepsilon$, and
2 the difference $PI(\overrightarrow{s}) - PI(\overrightarrow{s_1})$ is minimal over the set of all maximal supersequences of $\overrightarrow{s}$.

We denote the set of all ε-constricted maximal supersequences of pattern $\overrightarrow{s}$ as $\mathcal{C}^{max}(\overrightarrow{s})$. ε is an approximation margin parameter, whose value is user-specified and is in the range [0,1].

To illustrate Definition 10, let us consider again the pattern $\overrightarrow{s} = B \rightarrow B$ presented in Table 1 and let us assume that $\varepsilon = 0.25$. The participation index of $\overrightarrow{s}$ equals 0.675. The ε-constricted supersequence of $\overrightarrow{s}$ is the ST sequential pattern $\mathcal{C}^{max}(\overrightarrow{s}) = \{B \rightarrow B \rightarrow C \rightarrow E\}$, whose participation index equals 0.325.

Please note that an ST sequential pattern can have more than one ε-constricted maximal supersequence. For example, let us consider ST sequential pattern $\overrightarrow{s} = B \rightarrow C \rightarrow E$ presented in Table 1 and let us assume that approximation margin $\varepsilon = 0.1$. The two ε-constricted maximal supersequences of $\overrightarrow{s}$ are: $\mathcal{C}^{max}(\overrightarrow{s}) = \{A \rightarrow B \rightarrow C \rightarrow E, B \rightarrow B \rightarrow C \rightarrow E\}$, whose PI values are both equal to 0.375. Please also note that according to Definition 10 an ST sequential pattern $\overrightarrow{s}$ can be its own ε-constricted maximal supersequence[6].

From Definition 10 follows that an ST sequential pattern $\overrightarrow{s}$ can be a CSTS pattern of another ST sequential pattern, but also can have its own CSTS patterns. For example, let us consider the pattern $\overrightarrow{s} = B \rightarrow B \rightarrow C \rightarrow E$ from Table 1 and let us assume that $\varepsilon = 0.25$. $\overrightarrow{s}$ is a CSTS pattern of the ST sequential pattern $B \rightarrow B$, but also has its own CSTS pattern $\mathcal{C}^{max}(\overrightarrow{s}) = \{A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C\}$.

Definition 11 By $\mathcal{RC}^{max}(\overrightarrow{s_1})$ (reverse maximal closure set) we denote a set of ST sequential patterns for which $\overrightarrow{s_1}$ is the ε-constricted maximal supersequence ($\overrightarrow{s_1}$ is a CSTS pattern).

[6] In fact, as we present in Sect. 6, each ST sequential pattern of the maximal length patterns set (say L_k) is its own ε-constricted maximal supersequence.

Let us consider an ST sequential pattern $\overrightarrow{s_1} = A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C$ from Table 1, whose $PI = 0.25$. Additionally, let us assume that approximation margin is $\varepsilon = 0.25$. The set $\mathcal{RC}^{max}(\overrightarrow{s_1})$ is (the numbers in parentheses specify participation indexes):

- $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$,
- $A \rightarrow B \rightarrow B \rightarrow C \rightarrow E(0.25)$,
- $B \rightarrow B \rightarrow C \rightarrow E \rightarrow C(0.25)$,
- $A \rightarrow B \rightarrow B \rightarrow C(0.25)$,
- $B \rightarrow B \rightarrow C \rightarrow E(0.375)$,
- $B \rightarrow C \rightarrow E \rightarrow C(0.25)$,
- $A \rightarrow B \rightarrow B(0.25)$,
- $B \rightarrow B \rightarrow C(0.5)$,
- $B \rightarrow C \rightarrow E(0.375)$,
- $A \rightarrow B(0.5), B \rightarrow C(0.5), E \rightarrow C(0.5)$.

The set $\mathcal{RC}^{max}(\overrightarrow{s_1})$ is applied by the proposed CSTS-Miner algorithm to identify CSTS patterns.

Definition 12 (The set of all PI-strong CSTS patterns) The set of all PI-strong CSTS patterns is defined as the set of all ST sequential patterns that are PI-strong (according to Definition 5) and are CSTS patterns (according to Definition 10).

The proposed CSTS-Miner algorithm first discovers all PI-strong ST sequential patterns and then subsequently returns only those of them which are PI-strong CSTS patterns.

Theoretical properties of CSTS patterns

In this section, we derive theoretical properties of the introduced notions of (PI-strong) CSTS patterns. Specifically, we show that:

- for the parameter $\varepsilon = 0$ the set of all CSTS patterns is equivalent to the set of all closed ST sequential patterns (Lemma 1);
- each CSTS pattern is a closed ST sequential pattern regardless of the value of the approximation margin ε (Lemma 2);
- the set of CSTS patterns is a subset of the set of closed ST patterns for any value of the approximation margin parameter ε (Theorem 2);
- it can be verified if an ST sequential pattern $\overrightarrow{s}$ is PI-strong given the set of PI-strong CSTS patterns and how to approximate PI value of $\overrightarrow{s}$ (Lemma 3);
- one can approximate the value of PI of $\overrightarrow{s}$ with an error no greater than $\pm\,\varepsilon$ (Lemma 4).

Lemma 1 shows that given an ST sequential pattern $\overrightarrow{s}$ and its CSTS pattern $\overrightarrow{s_1}$ as well as for $\varepsilon = 0$, $\overrightarrow{s_1}$ is a closure of $\overrightarrow{s}$ and $\overrightarrow{s_1}$ is a closed ST sequential pattern.

Lemma 1 *Let $\overrightarrow{s}$ be an ST sequential pattern, $\varepsilon = 0$ and let $\overrightarrow{s_1}$ be a CSTS pattern of $\overrightarrow{s}$. The CSTS pattern $\overrightarrow{s_1}$ is a closure of $\overrightarrow{s}$ and $\overrightarrow{s_1}$ is a closed ST sequential pattern.*

Proof

The proof of lemma follows from Definitions 7 and 10. By Definition 7, a closure of ST sequential pattern $\overrightarrow{s}$ is such a closed ST sequential pattern $\overrightarrow{s*}$ that is a supersequence of $\overrightarrow{s}$ and whose $PI(\overrightarrow{s*}) = PI(\overrightarrow{s})$. For $\varepsilon = 0$ we have:

- the *PI* value of $\overrightarrow{s_1}$ being a CSTS pattern of $\overrightarrow{s}$ equals *PI* value of $\overrightarrow{s}$,
- CSTS pattern $\overrightarrow{s_1}$ is a maximal supersequence of $\overrightarrow{s}$.

Thus, for $\varepsilon = 0$ the CSTS pattern $\overrightarrow{s_1}$ is a closed ST sequential pattern. □

It follows from Lemma 1 that for the parameter $\varepsilon = 0$ the set of all PI-strong CSTS patterns is equivalent to the set of all PI-strong closed ST sequential patterns (in other words, for $\varepsilon = 0$ the proposed algorithm CSTS-Miner will return the same patterns set as the CST-SPMiner of [12]).

In Lemma 2 we show that each CSTS pattern is a closed ST sequential pattern regardless of the value of the approximation margin ε.

Lemma 2 *Each CSTS pattern is a closed ST sequential pattern regardless of the value of approximation margin ε.*

Proof

Let us assume that $\overrightarrow{s_1}$ is a CSTS pattern of an ST sequential pattern $\overrightarrow{s}$ given any value of ε. Now, let us assume that there exists a proper maximal supersequence $\overrightarrow{s_2}$ of $\overrightarrow{s_1}$ whose participation index $PI(\overrightarrow{s_2})$ equals the participation index $PI(\overrightarrow{s_1})$, that is $\overrightarrow{s_2}$ is a closure of $\overrightarrow{s_1}$. By Definition 7 $\overrightarrow{s_2}$ has to be a closed ST sequential pattern. However, as follows from Definition 10 this would contradict that $\overrightarrow{s_1}$ is a CSTS pattern. Hence, CSTS pattern $\overrightarrow{s_1}$ is a closed ST sequential pattern. □

In Theorem 2, we show that the set of CSTS patterns is a subset of the set of closed ST patterns for any value of the approximation margin parameter ε.

Theorem 2 *For any $\varepsilon \in [0, 1]$, the set of all CSTS patterns is a subset of the set of all closed ST sequential patterns.*

Proof

We already showed in Lemma 1 that for $\varepsilon = 0$, the set of all CSTS patterns is equal to the set of all closed ST sequential patterns. We also presented in Lemma 2 that each CSTS pattern is a closed ST sequential pattern regardless of the value of ε. Let us assume that $\varepsilon > 0$. If there exists a closed ST sequential pattern $\overrightarrow{s}$ that has a CSTS pattern and $\overrightarrow{s}$ is not a CSTS pattern itself, then $\overrightarrow{s}$ will not be included in the set of CSTS patterns (in such a case, the CSTS patterns set is a proper subset of the set of closed ST sequential patterns). Otherwise, if each closed ST sequential pattern is also a CSTS pattern, then the set of CSTS patterns is equal to the set of all closed ST sequential patterns. In either case, the CSTS patterns set is a subset of the closed ST sequential patterns set.

□

Theorem 2 shows that the number of discovered CSTS patterns is always less than or equal to the number of closed ST sequential patterns.

Let us illustrated Lemma 1, 2 as well as Theorem 2 with example patterns of the dataset presented in Fig. 1:

- For $\varepsilon = 0.5$, pattern $B \rightarrow B \rightarrow C \rightarrow E(PI = 0.375)$ is a CSTS pattern (as it is ε -constricted maximal supersequence of e.g. pattern $C \rightarrow E(PI = 0.8)$). At the same time pattern $B \rightarrow B \rightarrow C \rightarrow E$ is also a closed ST sequential pattern (as follows from Table 2).
- For $\varepsilon = 0$, the set of CSTS patterns is the same as the set of closed ST sequential patterns. Thus, each closed ST sequential pattern presented in Table 2 is also a CSTS pattern (e.g. pattern $B \rightarrow D(PI = 1)$ is a closed ST sequential pattern and CSTS patterns derived from singleton patterns $B(PI = 1)$ and $D(PI = 1)$).

Lemma 3 shows how to verify if an ST sequential pattern $\overrightarrow{s}$ is PI-strong given the set of PI-strong CSTS patterns and how to approximate its *PI* value.

Lemma 3 *For each ST sequential pattern the following hold*:

(i) *An ST sequential pattern $\overrightarrow{s}$ is PI-strong only if there exists a supersequence of $\overrightarrow{s}$ in the set of PI-strong CSTS patterns.*

(ii) *The participation index value of a PI-strong ST sequential pattern $\overrightarrow{s}$ is equal to or less than $PI(\overrightarrow{s_1}) + \varepsilon$, where $\overrightarrow{s_1}$ is such a minimal proper supersequence of $\overrightarrow{s}$ in PI-strong CSTS patterns, whose $PI(\overrightarrow{s_1})$ is the greatest.*

Proof

Ad(i) Follows immediately from Theorem 1.

Ad(ii) **Case 1.** Let us first assume that there is only one supersequence $\overrightarrow{s_1}$ of $\overrightarrow{s}$ in the set of PI-strong CSTS patterns. In this case, $\overrightarrow{s_1}$ is a ε-constricted maximal ST supersequence of $\overrightarrow{s}$ and by Definition 10 $PI(\overrightarrow{s}) \in \left[PI(\overrightarrow{s_1}), PI(\overrightarrow{s_1}) + \varepsilon\right]$. **Case 2.** Now let us assume that there is more than one proper supersequence of $\overrightarrow{s}$ in the set of PI-strong CSTS patterns. Since we can not indicate which one of them is ε-constricted maximal ST supersequence, then $PI(\overrightarrow{s}) \in \left[PI(\overrightarrow{s_1}), PI(\overrightarrow{s_1}) + \varepsilon\right]$, where $\overrightarrow{s_1}$ is a minimal proper supersequence from all proper supersequences of $\overrightarrow{s}$ in the set of PI-strong CSTS patterns.

□

Lemma 3 indicates that the set of PI-strong CSTS patterns (unlike the set of all PI-strong closed ST sequential patterns) is a *lossless* representation of all PI-strong ST sequential patterns but not *informative* (in a sense that the PI value of a PI-strong ST sequential pattern can be obtained from the set of PI-strong CSTS patterns only with a certain approximation). The question is how much the approximation of such PI value of a PI-strong ST sequential pattern differs from its exact PI value. In Lemma 4, we provide the value of such maximal approximation.

Lemma 4 *Given an ST sequential pattern $\overrightarrow{s}$ that has a supersequence in the CSTS patterns set one can not*:

- underestimate the exact *PI* value of $\overrightarrow{s}$ by less than $PI(\overrightarrow{s}) - \varepsilon$, and
- overestimate the exact *PI* value of $\overrightarrow{s}$ by more than $PI(\overrightarrow{s}) + \varepsilon$.

Proof

Let us assume that $\overrightarrow{s_1}$ is any proper supersequence of $\overrightarrow{s}$ in the set of CSTS patterns. We will consider two extreme cases:

Case 1. The value of $PI(\overrightarrow{s_1})$ equals $PI(\overrightarrow{s}) - \varepsilon$. In such a case, one can not underestimate $PI(\overrightarrow{s})$ by less than $PI(\overrightarrow{s}) - \varepsilon$.

Case 2. The value of $PI(\overrightarrow{s_1})$ equals $PI(\overrightarrow{s})$. In such a case, one can not overestimate $PI(\overrightarrow{s})$ by more than $PI(\overrightarrow{s}) + \varepsilon$.

Thus, the estimation of the $PI(\overrightarrow{s})$ value of the pattern $\overrightarrow{s}$ given the set of CSTS patters is always in the range $\left[PI(\overrightarrow{s}) - \varepsilon, PI(\overrightarrow{s}) + \varepsilon\right]$. □

Constricted ST sequential patterns miner

This section introduces our algorithm called CSTS-Miner for discovering the set of all PI-strong CSTS patterns. In Table 3, we present the notation used in the algorithms of this section. The main CSTS-Miner procedure is presented in Algorithm 1. The algorithm consists of two phases: *(i)* "top-down"—iterative generation of all PI-strong ST sequential patterns of length k from patterns of length $k-1$ until it is impossible to generate new patterns; *(ii)* "bottom-up"—calculation of PI-strong CSTS patterns. The "top-down" phase adapts the STBFM algorithm [11] for the discovery of PI-strong ST sequential patterns. Specifically, to efficiently generate new PI-strong patterns of length k from PI-strong patterns of length $k-1$, we adapted the SP-tree structure and extended it to the proposed MAX-Tree structure. MAX-Tree is used to not only iteratively generate new patterns but also, unlike SP-Tree offered in [11], to identify all PI-strong CSTS patterns. The "bottom-up" phase calculates PI-strong CSTS patterns in a recursive way starting with the set of the longest PI-strong ST sequential patterns L_k obtained in the "top-down" phase.

"Top-down" phase of the CSTS-miner algorithm

The "top-down" phase of Algorithm 1 is conducted by executing steps 1–18 of this algorithm. In step 1 of Algorithm 1, a singular candidate ST sequential pattern is generated from each event type in **F** and remembered in the set L_1 (patterns in L_1 constitute the first level of MAX-Tree). By Definition 4, singular patterns are always PI-strong since their PI values are equal to 1.

Subsequently, the PI-strong ST sequential patterns of length 2 are generated and remembered as L_2. The generation of such PI-strong ST sequential patterns is conducted in steps 3–13 using two nested loops, each of which iterates over all patterns in L_1. A new candidate pattern $\overrightarrow{s}$ of length 2 is always generated by concatenating the

Table 3 Notations and parameters used in the pseudocode listings

Notation	Description
D	A dataset of spatio-temporal event instances
F	A set of event types
$\overrightarrow{s}$	An ST sequential pattern
$\mathbf{N}(e, F)$	Neighborhood of event instance e with respect to event type $F \in \mathbf{F}$
$\mathbf{I}(\overrightarrow{s}, i)$	A set of event instances supporting i-th element of $\overrightarrow{s}$
$PR(\overrightarrow{s}, i)$	Participation ratio of i-th element of $\overrightarrow{s}$
$PI(\overrightarrow{s})$	Participation index of pattern $\overrightarrow{s}$
$\text{children}(\overrightarrow{s})$	Children patterns of $\overrightarrow{s}$
$\text{parent}_1(\overrightarrow{s})$	First parent of pattern $\overrightarrow{s}$
$\text{parent}_2(\overrightarrow{s})$	Second parent of pattern $\overrightarrow{s}$
$\mathcal{C}^{max}(\overrightarrow{s})$	A set of ε-constricted maximal supersequences of $\overrightarrow{s}$
$\mathcal{RC}(\overrightarrow{s})$	Set of ST sequential patterns, for which $\overrightarrow{s}$ is a ε-constricted maximal supersequence
PI_{min}	Participation index threshold of discovered patterns
ε	Approximation margin

two event types of singular patterns $\overrightarrow{s_i}$ and $\overrightarrow{s_j}$. We will refer to the patterns $\overrightarrow{s_i}$ and $\overrightarrow{s_j}$ as the first and the second parent of $\overrightarrow{s}$, respectively. Please note that $\overrightarrow{s}$ can consist of two the same event types (in such a case, $\overrightarrow{s_i} = \overrightarrow{s_j}$). The set of instances supporting the second element of $\overrightarrow{s}$ is calculated in step 6 and consists of event instances of type $\overrightarrow{s}[2]$ in $\mathbf{D}$ which belong to neighborhoods of event instances in the set $\mathbf{I}(\overrightarrow{s_i}, 1)$.

The participation index of the candidate generated pattern $\overrightarrow{s}$ is calculated and verified in steps 7 and 8 of Algorithm 1. If $\overrightarrow{s}$ occurred to be PI-strong, then $\overrightarrow{s}$ is inserted into the list of children of its first parent children($\overrightarrow{s_i}$) and into the set L_2.

Algorithm 1 CSTS-Miner($\mathbf{D}, \mathbf{F}, R, T, PI_{min}, \varepsilon$)

Input: $\mathbf{D}$ - spatio-temporal event dataset, $\mathbf{F}$ - set of event types, R - spatial threshold value, T - time window threshold value, PI_{min} - participation index threshold value, ε - approximation margin.

Output: - $\bigcup_k \{\overrightarrow{s} \in L_k \mid \overrightarrow{s}.\mathcal{RC} \neq \emptyset \vee (\overrightarrow{s}.\mathcal{RC} = \emptyset \wedge \overrightarrow{s}.\mathcal{C}^{max} = \emptyset)\}$ - a set of PI-strong CSTS patterns.

{*———- CSTS-Miner "top-down" phase ———-*}
1: $L_1 :=$ generate patterns of length 1 from all event types in $\mathbf{F}$
2: $L_2 := \emptyset$
3: **for each** $\overrightarrow{s_i} \in L_1$ **do**
4: **for each** $\overrightarrow{s_j} \in L_1$ **do**
5: $\overrightarrow{s} := \overrightarrow{s_i}[1] \rightarrow \overrightarrow{s_j}[1]$
6: $\mathbf{I}(\overrightarrow{s}, 2) := \bigcup_{e \in \mathbf{I}(\overrightarrow{s_i}, 1)} N_{\overrightarrow{s}[2]}(e)$
7: $\overrightarrow{s}.PI := PR(\overrightarrow{s}, 2)$
8: **if** $\overrightarrow{s}.PI > PI_{min}$ **then**
9: parent$_1(\overrightarrow{s}) := \overrightarrow{s_i}$; parent$_2(\overrightarrow{s}) := \overrightarrow{s_j}$
10: Add $\overrightarrow{s}$ to children(parent$_1(\overrightarrow{s})$); add $\overrightarrow{s}$ to L_2
11: **end if**
12: **end for**
13: **end for**
14: $k := 2$
15: **while** $L_k \neq \emptyset$ **do** ▷ L_k - a set of PI-strong ST sequential patterns of length k.
16: $k := k + 1$
17: $L_k :=$ GenAndVerify(L_{k-1})
18: **end while**

{*———- CSTS-Miner "bottom-up" phase ———-*}
19: **while** $k > 1$ **do**
20: **for** $\overrightarrow{s} \in L_k$ **do**
21: VerifySupersequence($\overrightarrow{s}$, parent$_1(\overrightarrow{s})$)
22: VerifySupersequence($\overrightarrow{s}$, parent$_2(\overrightarrow{s})$)
23: **end for**
24: $k := k - 1$
25: **end while**
26: **return** $\bigcup_k \{\overrightarrow{s} \in L_k \mid \mathcal{RC}^{max}(\overrightarrow{s}) \neq \emptyset \vee (\mathcal{RC}^{max}(\overrightarrow{s}) = \emptyset \wedge \mathcal{C}^{max}(\overrightarrow{s}) = \emptyset)\}$ ▷ Return all CSTS patterns.

Algorithm 2 GenAndVerify(L_{k-1}) [12]

Input: L_{k-1} - a set of PI-strong ST sequential patterns of length $k-1$.
Output: L_k - a set of PI-strong ST sequential patterns of length k that are generated by this procedure from the set of PI-strong ST sequential patterns of length $k-1$.

1: $L_k := \emptyset$
2: **for each** $\overrightarrow{s_i} \in L_{k-1}$ **do**
3: $\quad \overrightarrow{s_l} := \text{parent}_2(\overrightarrow{s_i})$
4: $\quad$ **for each** $\overrightarrow{s_j} \in \text{children}(\overrightarrow{s_l})$ **do**
5: $\quad\quad \overrightarrow{s} := \overrightarrow{s_i}[1] \rightarrow \overrightarrow{s_i}[2] \rightarrow \cdots \rightarrow \overrightarrow{s_i}[k-1] \rightarrow \overrightarrow{s_j}[k-1]$
6: $\quad\quad \mathbf{I}(\overrightarrow{s}, k) := \bigcup_{e \in \mathbf{I}(\overrightarrow{s_i}, k-1)} N_{\overrightarrow{s}[k]}(e)$
7: $\quad\quad \overrightarrow{s}.PI := \min\big(PI(\overrightarrow{s_i}), PR(\overrightarrow{s}, k)\big)$
8: $\quad\quad$ **if** $PI(\overrightarrow{s}) > PI_{min}$ **then**
9: $\quad\quad\quad \text{parent}_1(\overrightarrow{s}) := \overrightarrow{s_i}$; $\text{parent}_2(\overrightarrow{s}) := \overrightarrow{s_j}$
10: $\quad\quad\quad$ Add $\overrightarrow{s}$ to $\text{children}\big(\text{parent}_1(\overrightarrow{s})\big)$; add $\overrightarrow{s}$ to L_k
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end for**
14: **return** L_k

Steps from 15 to 18 of the Algorithm 1 consist of the iterative generation and verification of PI-strong ST sequential patterns of length greater than 2 using the function *GenAndVerify* shown in Algorithm 2. Specifically, the *GenAndVerify* function generates all PI-strong ST sequential patterns L_k from PI-strong ST sequential patterns L_{k-1}. To this end, the function uses the first parent, the second parent as well as the children list of patterns in L_{k-1}. As follows from Lemma 5 (presented in [12] as Lemma 5),[7] a candidate ST sequential pattern $\overrightarrow{s}$ of length ≥ 3 can be obtained by concatenating all elements of its first parent with the last element of its second parent.

Lemma 5 [*Construction of an ST sequential pattern from its first and second parent* [12]] *Let* $\overrightarrow{s}$ *be an ST sequential pattern of length* $m \geq 3$ *and* $\overrightarrow{s_1}$ *be the first parent of* $\overrightarrow{s}$. *Then,* $\overrightarrow{s} = \overrightarrow{s_1} \rightarrow F_m$, *where* F_m *is the last element of* $\text{parent}_1(\overrightarrow{s})$, *and* $\text{parent}_2(\overrightarrow{s}) = \text{parent}_2(\overrightarrow{s_1}) \rightarrow F_m$.

Algorithm 2 proceeds as follows. First, the set L_k is initialized. Subsequently, the loop in step 2 iterates over all patterns in $\overrightarrow{s_i} \in L_{k-1}$ ($\overrightarrow{s_i}$ is the first parent of a new candidate pattern $\overrightarrow{s}$) and the loop in step 4 iterates over all children patterns $\overrightarrow{s_j} \in \text{parent}_2(\overrightarrow{s_i})$ of the second parent of $\overrightarrow{s_i}$. Each of such $\overrightarrow{s_j}$ child patterns constitutes the second parent of a new candidate pattern $\overrightarrow{s}$. Thus, the elements of $\overrightarrow{s}$ are: $\overrightarrow{s} := \overrightarrow{s_i}[1] \rightarrow \overrightarrow{s_i}[2] \rightarrow \cdots \rightarrow \overrightarrow{s_i}[k-1] \rightarrow \overrightarrow{s_j}[k-1]$.

After the elements of $\overrightarrow{s}$ are obtained, the set of instances supporting its last element is computed according to step 6 of Algorithm 2 and the *PI* value of $\overrightarrow{s}$ is calculated according to step 7 of Algorithm 2. If the *PI* value is greater than the participation index threshold PI_{min}, then $\overrightarrow{s}$ is inserted to L_k and appended to the children list of $\overrightarrow{s_i}$: $\text{children}(\overrightarrow{s_i})$. Otherwise, $\overrightarrow{s}$ is discarded as not being a PI-strong pattern.

[7] We refer the reader to [12] for the proof of Lemma 5.

The generation of new candidate ST sequential patterns ends when the *GenAndVerify* function can not generate any new patterns.

"Bottom-up" phase of the CSTS-miner algorithm

While the "top-down" phase generates all PI-strong ST sequential patterns, the "bottom-up" phase of Algorithm 1 starts in step 19 and is entirely dedicated to calculation of these PI-strong ST sequential patterns which are PI-strong CSTS patterns. The verification of PI-strong ST sequential patterns as being CSTS patterns starts from the set L_k of the longest patterns and is iteratively continued for the subsequent sets of patterns $L_{k-1}, L_{k-2}, \ldots, L_2$.

Algorithm 3 VERIFYSUPERSEQUENCE($\overrightarrow{s}$, $\overrightarrow{s_i}$)

Input: $\overrightarrow{s}$ - PI-strong ST sequential pattern, whose $\mathcal{RC}^{max}$ set needs to be obtained.

```
if PI(s) ≥ (PI s_i) − ϵ then
    if s_i ∉ RC^max(s) then
        if C^max(s_i) = ∅ ∨ len(s) = len(C^max(s_i)) then
            if C^max(s_i) = ∅ ∨ PI(s) = PI(C^max(s_i)) then
                Insert s to C^max(s_i)
                Insert s_i to RC^max(s)
            else if PI(s) > PI(C^max(s_i)) then
                Remove s_i from
                    RC^max(s_k) of all s_k ∈ C^max(s_i)
                C^max(s_i) := ∅
                Insert s to C^max(s_i)
                Insert s_i to RC^max(s)
            end if
        end if
    end if
    if parent_1(s_i) ≠ NULL then
        VERIFYSUPERSEQUENCE(s, parent_1(s_i))
    end if
    if parent_2(s_i) ≠ NULL then
        VERIFYSUPERSEQUENCE(s, parent_2(s_i))
    end if
end if
return
```

The function *VerifySupersequence* presented in Algorithm 3 obtains two ST sequential patterns $\overrightarrow{s}$ and $\overrightarrow{s_i}$ and verifies if $\overrightarrow{s}$ is a CSTS pattern of $\overrightarrow{s_i}$. To this end, function *VerifySupersequence* applies Definition 10 and Theorem 1. Specifically, *VerifySupersequence* conducts the following steps:

1 Checks if the *PI* value of $\overrightarrow{s}$ is greater than the *PI* value of $\overrightarrow{s_i}$ minus approximation margin ε (in step 1). This fulfills the first condition of Definition 10.

2 Checks if $\overrightarrow{s_i}$ already belongs to the RC^{max} list of $\overrightarrow{s}$ (in step 2). Due to construction of MAX-Tree, it can happen that the function *VerifySupersequence* will be invoked multiple times for the same two sequences $\overrightarrow{s}$ and $\overrightarrow{s_i}$. Thus, the check prevents the situation when $\overrightarrow{s_i}$ is added multiple times to the list $RC^{max}(\overrightarrow{s})$.

3 Verifies whether there is no pattern of $\overrightarrow{s_i}$ in the set $C^{max}(\overrightarrow{s_i})$ or the length of $\overrightarrow{s}$ equals the length of the patterns in $C^{max}(\overrightarrow{s_i})$. In such a case, the two situations are possible:

- Either $C^{max}(\overrightarrow{s_i}) = 0$ or the *PI* value of $\overrightarrow{s}$ equals the *PI* values of $C^{max}(\overrightarrow{s_i})$ patterns. In any case, $\overrightarrow{s}$ is inserted to $C^{max}(\overrightarrow{s_i})$ and $\overrightarrow{s_i}$ is inserted to $\mathcal{RC}^{max}(\overrightarrow{s_i})$.
- Alternatively, if the *PI* value of $\overrightarrow{s}$ is greater than the *PI* values of patterns in $C^{max}(\overrightarrow{s_i})$, then $\overrightarrow{s_i}$ is removed from the $\mathcal{RC}^{max}$ lists of all patterns in $C^{max}(\overrightarrow{s_i})$ and $C^{max}(\overrightarrow{s_i})$ is set to be empty. Next, $\overrightarrow{s}$ is added to $C^{max}(\overrightarrow{s_i})$ and $\overrightarrow{s_i}$ is added to $\mathcal{RC}^{max}(\overrightarrow{s})$. These operations fulfill the second condition of Definition 10.

If the *PI* value of $\overrightarrow{s}$ is greater than the *PI* value of $\overrightarrow{s_i}$ minus approximation margin ε, then *VerifySupersequence* function is recursively invoked for $\overrightarrow{s}$ and the parent patterns $\text{parent}_1(\overrightarrow{s_i})$, $\text{parent}_2(\overrightarrow{s_i})$ of $\overrightarrow{s_i}$ to verify whether $\overrightarrow{s}$ is also their CSTS pattern. Otherwise, as follows from Theorem 1, $\overrightarrow{s}$ can not be a CSTS pattern of any of the parent patterns $\text{parent}_1(\overrightarrow{s_i})$, $\text{parent}_2(\overrightarrow{s_i})$ of $\overrightarrow{s_i}$. Thus, invokes of *VerifySupersequence* $\overrightarrow{s}$, $\text{parent}_1(\overrightarrow{s_i})$ and *VerifySupersequence* $\overrightarrow{s}$, $\text{parent}_2(\overrightarrow{s_i})$ are skipped.

The complete MAX-Tree created for the dataset shown in Fig. 1 is presented in Fig. 2. The tree is created by Algorithm 1 using the following input parameters: spatial threshold value $R = 10$, time window threshold value $T = 20$, participation index threshold value $PI_{min} = 0.25$, approximation margin value $\varepsilon = 0.25$. All patterns in the tree are PI-strong ST sequential patterns. However, only blue boxes represent those of them, which are also PI-strong CSTS patterns.

For the pattern $\overrightarrow{s} = B \rightarrow B \rightarrow C$, the set of its CSTS patterns is $C^{max}(\overrightarrow{s}) = \{A \rightarrow B \rightarrow B \rightarrow C \rightarrow E \rightarrow C\}$, while the minimal proper supersequence of $\overrightarrow{s}$ among the set of PI-strong CSTS patterns returned by Algorithm 1 is $\overrightarrow{s_1} = B \rightarrow B \rightarrow C \rightarrow E$. Let us consider how one can approximate the participation index value of pattern $\overrightarrow{s} = B \rightarrow B \rightarrow C$ given the set of all PI-strong CSTS patterns presented in Fig. 2. Since the set of PI-strong CSTS patterns contains more than one supersequence of $\overrightarrow{s}$, then the value of participation index of $PI(\overrightarrow{s}) \leq PI(\overrightarrow{s_1}) + \varepsilon = 0.375 + 0.25 = 0.625$. In fact, the exact PI value of $\overrightarrow{s}$ equals $PI(\overrightarrow{s}) = 0.5$.

CSTS-miner complexity analysis

Let us start with the analysis of computational cost of the "Top-down" phase of Algorithm 1. Let us assume that $|\mathbf{F}|$ and $|\mathbf{D}|$ denote the number of event types and event instances in the dataset, respectively. Furthermore, let us assume that A_{Chl} is the average number of children of a node in the MAX-Tree and A_I is the average number of instances supporting an element of a sequential pattern. The plane sweep algorithm of [34] applied in this study requires on average $A_I \cdot \log |\mathbf{D}|$ number of operations to obtain the neighborhoods of event instances $I_{\overrightarrow{s}[i]}$ supporting an i-th element of a sequential pattern assuming that event instances in $\mathbf{D}$ are initially sorted in non-decreasing order according to the occurrence times. Thus, the average number of computations needed to obtain children patterns of a sequential pattern in MAX-Tree equals $A_{chl} \cdot A_I \cdot \log |\mathbf{D}|$. Let us assume that $|L_k|$ denotes the average number of PI-strong ST sequential patterns of length $k = 2, 3, \ldots$. The

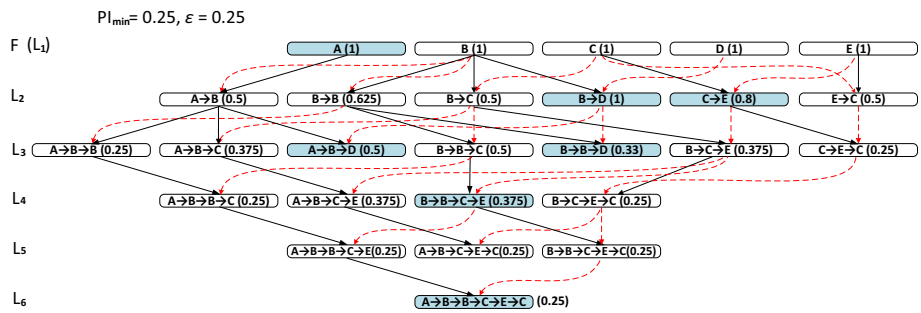

Fig. 2 The complete MAX-Tree generated with parameters $PI_{min} = 0.25$ and $\varepsilon = 0.25$ for the dataset presented in Fig. 1 (all boxes represent PI-strong ST sequential patterns, while blue boxes represent patterns being also PI-strong CSTS patterns; PI values are given in parentheses next to the patterns)

number of computations needed to generate patterns of MAX-Tree of length $3, 4, \ldots$ equals $|L_k| \cdot A_{chl} \cdot A_I \cdot \log |\mathbf{D}|$. Additionally, let $T - 1$ denote the number of levels of MAX-Tree excluding level 1 which contains all event types in $\mathbf{F}$. Since Algorithm 1 generates all patterns of length L_2 in two nested loops iterating over patterns L_1 that consist of singleton event types in $\mathbf{F}$, we can approximate the calculation numbers of "Top-down" phase as

$$O\Big((T-1) \cdot |L_k| \cdot A_{chl} \cdot A_I \cdot \log |\mathbf{D}| + |\mathbf{F}|^2 \cdot A_I \cdot \log |\mathbf{D}|\Big) =$$
$$O\Big(((T-1) \cdot |L_k| \cdot A_{chl} + |\mathbf{F}|^2) \cdot A_I \cdot \log |\mathbf{D}|\Big).$$

Let us now analyse computational complexity of the "Bottom-up" phase of Algorithm 1. CSTS-Miner calls Algorithm 3 for both parents of each pattern of levels $k = 2, 3, \ldots$. Given the height of MAX-Tree equal to T, the computational cost of phase "Bottom-up" is strictly dependent on the specified value of ε parameter. Let assume that for a generated MAX-Tree, the participation indexes changes from 1 for singleton patterns L_1 to PI_{min} for patterns in L_T. Given that each pattern has two parent patterns, we can assume that the number of ascendant patterns to be verified for a given ST sequential pattern equals: $2^{(1-PI_{min}) \cdot \varepsilon \cdot T}$. Thus, the computational cost of the "Bottom-up" phase equals:

$$O\big((T-1) \cdot |L_k| \cdot 2^{(1-PI_{min}) \cdot \varepsilon \cdot T} \cdot |\mathcal{C}^{max}(\overrightarrow{s})|\big).$$

$|\mathcal{C}^{max}(\overrightarrow{s})|$ is the average number of CSTS patterns of sequence $\overrightarrow{s}$. Algorithm 3 in a pessimistic cases require to scan over such list when a detection of a CSTS pattern is detected.

To analyse memory complexity of Algorithm 1 let us note that in order to restore a sequence of elements constituting a pattern, at each level $k = 1, 2, \ldots$, of the tree, it is enough to store only the last element of a sequence of event types constituting a pattern. Thus, the memory required to store elements of sequences of the discovered PI-strong ST-sequential patterns along with their participation indexes equals $T \cdot |L_k|$. Also, in order to generate new candidate patterns it is enough to store in the computer's memory only event instances of the last elements of patterns of two subsequent lengths $k - 1$ and k. Hence, the memory complexity of "Top-down" phase of Algorithm 1 equals:

$$O\big(T \cdot |L_k| + |L_k| \cdot |L_{k-1}| \cdot A_I\big) = O\big(|L_k| \cdot (T + |L_k| \cdot A_I)\big).$$

In order to asses the memory complexity of the "Bottom-up" phase let us assume that $|\mathcal{C}^{max}(\overrightarrow{s})|$ and $|\mathcal{RC}^{max}(\overrightarrow{s})|$ are the average numbers of CSTS patterns of $\overrightarrow{s}$ and the average number of patterns for which $\overrightarrow{s}$ is a CSTS pattern, respectively. For each pattern $\overrightarrow{s}$ of MAX-tree, Algorithm 3 maintains the list $\mathcal{C}^{max}(\overrightarrow{s})$. However, the list $\mathcal{RC}^{max}(\overrightarrow{s})$ needs to be maintained for all patterns exept the singleton patterns. Thus, we can asses the memory complexity of the second phase of Algorithm 1 as:

$$O\big(T \cdot |L_k| \cdot |\mathcal{C}^{max}(\overrightarrow{s})| + (T-1) \cdot |L_k| \cdot |\mathcal{RC}^{max}(\overrightarrow{s})|\big).$$

Experiments

In this section, we first review datasets used for the experiments and describe our experimental setup. Then we provide results of the comparison of the proposed CSTS-Miner algorithm with the STS-Miner [9], CST-SPMiner [12], STBFM [12] and CSTPM [10] algorithms.

Selected datasets

For the experiments with the proposed CSTS-Miner algorithm we selected two publicly available datasets, each of which consists of crime event incidents.

Pittsburgh police incident blotter dataset

The first dataset selected for the experiments is *Pittsburgh Police Incident Blotter Dataset* that contains crime incidents collected by the Police Department of Pittsburgh City over the period 31.12.1989–31.12.2019 [35]. The dataset was validated according to the Uniform Crime Reporting (UCR) standards [36] and consists of such attributes as: *incident time*, *incident location* (defined by street name and number), *incident neighborhood*, *incident type*, *description of offense*, incident *longitude* and *latitude* locations. For the purposes of experiments we selected the following attributes of the dataset: *incident type*, *incident time*, *longitude* and *latitude* which are directly used by the proposed CSTS-Miner algorithm. In the experiments, we use only crime incidents reported between 01.01.2017 and 31.12.2019. The number of crime incidents reported over this time period is 122,895. However, approximately 40% of them contain missing values for one of the selected attributes: incident type, geographical location or incident time. Thus, we decided to remove such incidents. The resultant dataset contains 72,867 crime event incidents of 236 unique incident types. All of these 236 incident types are selected as event type set **F**. To the most frequent incident types belong: *theft from auto*, *simple assault*, *public drunkenness*, *criminal mischief* or *harassment*. The attribute *incident time* (which specifies incident occurrence time with exact occurrence date and a timestamp given in hours, minutes and seconds) is transformed into the number of minutes that passed from the timestamp *01.01.2017 00:00*. Thus, the time window parameter T used by the CSTS-Miner algorithm is specified in minutes. In Table 4, we present the characteristic of the resultant dataset.

In Fig. 3, we present the location of the first two thousand crime incidents of ten most frequent incident types from the resultant Pittsburgh Police Incident Blotter Dataset.

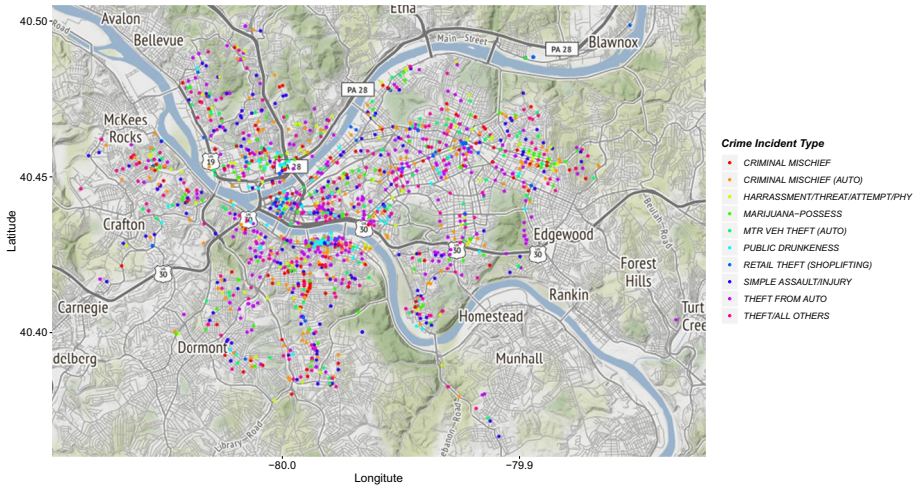

Fig. 3 The first two thousand crime incidents of ten most frequent incident types from the resultant Pittsburgh Police Incident Blotter Dataset

Boston crime incident reports dataset

The second of the selected datasets is *Boston Crime Incidents Reports Dataset* provided by the Boston Police Department [37]. The dataset was collected over the period 08.07.2012–10.08.2015. However, in the experiments, we extracted only crime incidents that occurred between 01.01.2014–31.12.2014. The Boston Crime Incidents Reports Dataset contains several attributes, such as: *incident location*, *incident time*, *incident type*, *used weapon type* (such as, for example, unarmed or firearm) *shooting presence*, *police shift* or *occurrence district* and *occurrence area*. Similarly to the Pittsburgh Police Incident Blotter Dataset, for the experiments we selected only attributes: *incident location* (given by longitute and latitude), *incident time* and *incident type*. Since the attribute *incident type* does not contain only crime

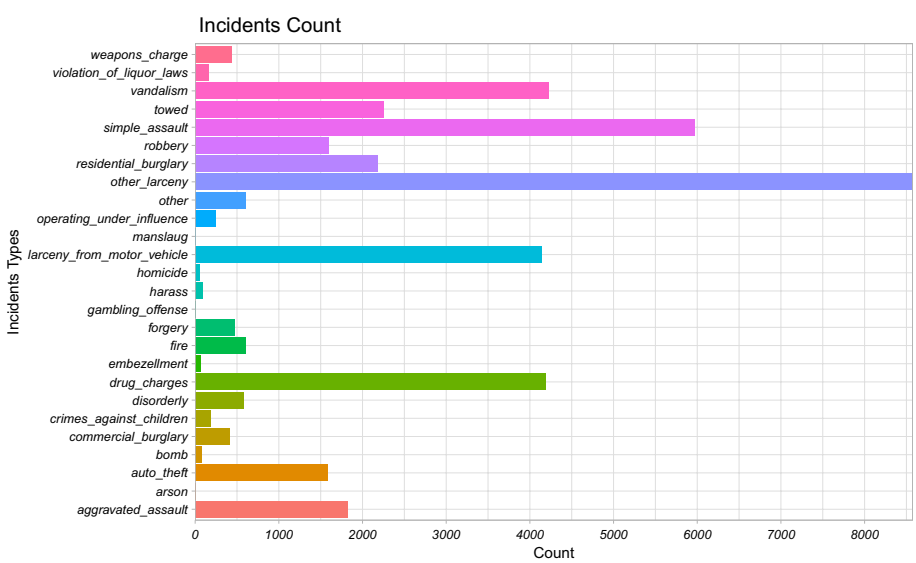

Fig. 4 Incidents count per crime type in the *complete* Boston Crime Incident Reports Dataset used for experiments

Table 4 The characteristic of the resultant Pittsburgh Police Incident Blotter Dataset

Parameter	Value
No. of incident types (|**F**|)	236
No. of incident instances (|**D**|)	72 867
Avg. no. of incidents per type	309
Median no. of incidents per type	26
Std. of no. of incidents per type	784
Min. no. of instances in a type	1
Max. no. of instances in a type	6201

incident types, but also other incident types, such as *medical assist* or *property found*, we preprocessed the dataset to obtain incidents of all 26 crime event types present in the dataset. The examples of event types are: *aggravated assault*, *arson*, *auto theft*, *drug charges*.

Additionally, to better analyze the results that can be obtained with CSTS-Miner, we selected incidents of only the ten least frequent crime types in the complete dataset to create the *reduced* dataset. These crime types are *violation of liquor laws, operating under influence, manslaughter, homicide, harassment, gambling offense, embezzlement, crimes against children, bomb, arson*.

In Table 5, we present the characteristic of the complete dataset, while in Table 6 the characteristic of the reduced dataset is shown. Since there are only 26 crime event types in the complete dataset, we decided to present the number of incidents of each type in the histogram shown in Fig. 4.

Experimental setup

In our experiments, the spatial distance threshold R of a neighborhood of an event instance e is specified in meters. However, the locations of event instances in the obtained datasets are specified using the longitude and latitude coordinates. Thus, we apply the following procedure to transform the distance between two event instances $e_1, e_2 \in \mathbf{D}$ into meters. First, each coordinate (either longitude or latitude) of an instance e is converted into radians according to Eq. (3) in which, $e.lat$ refers to the latitude coordinate and $e.lon$ refers to the longitude coordinate of instance e, respectively. Next, the distance in meters between two instances is obtained according to Eq. (4). In Eq. (4), earthRadius denotes the radius of Earth in kilometers.

$$rad_{e.lat} = \frac{e.lat \cdot \pi}{180}; \quad rad_{e.lon} = \frac{e.lon \cdot \pi}{180}. \tag{3}$$

$$\begin{aligned} &D_{slat} = \sin\big((rad_{e_2.lat} - rad_{e_1.lat})/2\big). \\ &D_{slon} = \sin\big((rad_{e_2.lon} - rad_{e_1.lon})/2\big). \\ &Dist_{(e_1,e_2)} = 2 \cdot \text{earthRadius} \cdot 1000. \\ &\arcsin\left(\sqrt{D_{slat}^2 + D_{slon}^2 \cdot \cos(rad_{e_1.lat}) \cdot \cos(rad_{e_2.lat})}\right). \end{aligned} \tag{4}$$

Table 5 The characteristic of the *complete* Boston Crime Incident Reports Dataset

Parameter	Value
No. of incident types (|**F**|)	26
No. of incident instances (|**D**|)	40 545
Avg. no. of incidents per type	1559
Median no. of incidents per type	527
Std. of no. of incidents per type	2176
Min. no. of instances in a type	1
Max. no. of instances in a type	8575

Table 6 The characteristic of the *reduced* Boston Crime Incident Reports Dataset

Parameter	Value
No. of incident types (|**F**|)	10
No. of incident instances (|**D**|)	896
Avg. no. of incidents per type	90
Median no. of incidents per type	72
Std. of no. of incidents per type	83
Min. no. of instances in a type	1
Max. no. of instances in a type	245

The implementations of all algorithms selected for the experiments are prepared in C++. We ran the experiments using a computer equipped with Apple M1 processor and 16 GB of RAM memory. Our implementations of the algorithms (STS-Miner, STBFM, CSTPM, CST-SPMiner, CSTS-Miner) are available at the GitHub repository.[8]

Results of the experiments

In the experiments, we compare our proposed CSTS-Miner with the four other algorithms: STS-Miner applying participation index measure [9], STBFM [11], CSTPM [10]. All of them discover PI-strong ST sequential patterns. CSTS-Miner is also compared with CST-SPMiner [12], which discovers PI-strong closed ST sequential patterns.

We aim to measure the number of discovered patterns by each algorithm and its computation time for each of the selected datasets. The obtained results for each dataset are presented in Tables 7, 8 and 9. The results presented in these tables were obtained for the following input parameters of the five compared algorithms:

- For the Pittsburgh Police Incident Blotter Dataset: $R = 350$ m, $T = 11{,}520$ (8 days), $PI_{min} = \{0.33, 0.32, \ldots, 0.25\}$, $\varepsilon = \{0.025, 0.05, 0.075\}$.
- For the *complete* Boston Crime Incidents Report Dataset: $R = 300$ m, $T = 5760$ min (4 days), $PI_{min} = \{0.055, 0.05, \ldots, 0.015\}$, $\varepsilon = \{0.05, 0.1, 0.15\}$.
- For the *reduced* Boston Crime Incidents Report Dataset: $R = 500$ m, $T = 43{,}200$ min (30 days), $PI_{min} = \{0.01, 0.0095, \ldots, 0.005\}$, $\varepsilon = \{0.05, 0.1, 0.15\}$.

[8] https://github.com/piotrMaciag32/CSTS-Miner.

Table 7 Computation time (in sec) and the number of discovered patterns for the Pittsburgh Police Incident Blotter Dataset

R = 350 meters, T = 11520 (8 days)

PI_{min}	PI-strong ST seq. patterns						PI-strong closed ST seq. patterns		PI-strong CSTS patterns					
	STS-Miner [9]		*CSTPM* [10]		*STBFM* [11]		*CST-SPMiner* [12]		*CSTS-Miner* ($\varepsilon = 0.025$)		*CSTS-Miner* ($\varepsilon = 0.05$)		*CSTS-Miner* ($\varepsilon = 0.075$)	
	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns
0.33	26	796	9	796	9	796	9	657	9	594	9	576	9	556
0.32	31	1002	9	1002	9	1002	9	819	9	712	9	681	9	666
0.31	39	1320	9	1320	9	1320	9	1065	9	883	9	827	9	802
0.3	56	1990	9	1990	9	1990	9	1573	9	1161	9	1050	9	997
0.29	116	4371	11	4371	10	4371	10	3323	10	2235	10	2018	10	1932
0.28	269	10,434	17	10,434	12	10,434	12	7741	12	4516	12	4206	12	4104
0.27	898	35,166	100	35,166	19	35,166	19	25,446	19	14,956	20	14,182	22	13,939
0.26	–	–	1658	143,666	47	143,666	47	100,235	51	60,977	67	58,907	99	58,519
0.25	–	–	–	–	206	1,073,917	206	683,860	253	515,128	712	509,232	1744	508,492

Table 8 Computation time (in sec) and the number of discovered patterns for the *complete* Boston Crime Incidents Dataset (- denotes that the computation times were too long to obtain the results)

R = 300 meters, T = 5760 (4 days)														
PI_{min}	PI-strong ST seq. patterns						PI-strong closed ST seq. patterns		PI-strong CSTS patterns					
	STS-Miner [9]		*CSTPM*[10]		*STBFM* [11]		*CST-SPMiner* [12]		*CSTS-Miner* ($\varepsilon = 0.1$)		*CSTS-Miner* ($\varepsilon = 0.05$)		*CSTS-Miner* ($\varepsilon = 0.15$)	
	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns
0.055	69	3982	4	3982	3	3982	3	3346	3	2472	3	2232	3	2170
0.05	92	5386	4	5386	3	5386	4	4484	3	3265	3	2977	3	2913
0.045	129	7550	6	7550	4	7550	4	6248	4	4410	4	4060	4	3991
0.04	199	12,014	9	12,014	5	12,014	5	9899	5	6713	5	6292	5	6211
0.035	350	22,191	23	22,191	6	22,191	6	17,732	6	11,210	6	10,663	6	10,576
0.03	676	43,327	77	43,327	8	43,327	8	33,954	9	21,091	8	20,449	8	20,328
0.025	–	–	440	102,731	14	102,731	14	79,701	14	48,775	14	47964	14	47797
0.02	–	–	–	–	33	367,400	32	282,156	33	168,511	33	167,361	34	167,139
0.015	–	–	–	–	145	2,819,490	142	2,040,303	160	1,173,939	176	1,172,294	190	1,171,955

Table 9 Computation time (in sec) and the number of discovered patterns for the *reduced* Boston Crime Incidents Dataset

R = 500 meters, T = 43200 (30 days)														
PI_{min}	PI-strong ST seq. patterns						PI-strong closed ST seq. patterns		PI-strong CSTS patterns					
	STS-Miner [9]		*CSTPM* [10]		*STBFM* [11]		*CST-SPMiner* [12]		*CSTS-Miner* ($\varepsilon = 0.05$)		*CSTS-Miner* ($\varepsilon = 0.1$)		*CSTS-Miner* ($\varepsilon = 0.15$)	
	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns	Time	# Patterns
0.01	6	34,102	46	34,102	0	34,102	0	9987	9	8629	13	8580	17	8575
0.0095	6	34,102	46	34,102	0	34,102	0	9987	9	8629	13	8580	17	8575
0.009	6	34,102	46	34,102	0	34,102	0	9987	9	8629	13	8580	17	8575
0.0085	6	34,102	46	34,102	0	34,102	0	9987	9	8629	13	8580	17	8575
0.008	27	141,860	1055	141,860	0	141,860	0	51,219	217	44,228	318	44,165	393	44,165
0.0075	27	141,860	1048	141,860	0	141,860	0	51,219	215	44,228	324	44,165	393	44,165
0.007	27	141,860	1050	141,860	0	141,860	0	51,219	216	44,228	318	44,165	393	44,165
0.0065	27	141,860	1099	141,860	0	141,860	0	51,219	216	44,228	317	44,165	393	44,165
0.006	30	161,238	1394	161,238	0	161,238	0	56,812	277	47,407	405	47,347	487	47,342
0.0055	30	161,238	1381	161,238	0	161,238	0	56,812	278	47,407	405	47,347	487	47,342
0.005	43	228,285	2787	228,285	1	228,285	1	76,894	369	65,967	530	65,903	624	65,899

Table 7 presents the results for the Pittsburgh Police Incident Blotter dataset. As it can be noted from the table, CSTS-Miner can discover much fewer patterns for all three selected values of its approximation margin ε parameter than the other four selected algorithms. For example, for the participation index threshold $PI_{min} = 0.26$, there are 143,666 PI-strong ST sequential patterns and 100,235 PI-strong closed ST sequential patterns, while the number of PI-strong CSTS patterns discovered for the approximation threshold value $\varepsilon = 0.1$ is only 58,519. As it can be noted from Table 7, the increasing values of approximation margin ε can result in a significant increase of computation times of CSTS-Miner. For example, for $PI_{min} = 0.25$ STBFM and CST-SPMiner both executed in 207 s, while CSTS-Miner for $\varepsilon = 0.025$ executed in 253 s and for $\varepsilon = 0.075$ it executed in 1744s.

Slightly different results were presented in Table 8 for the *complete* Boston Crime Incident Report dataset. In the case of this dataset, it was possible to obtain a similar reduction in the number of discovered patterns as in the case of the Pittsburgh Police Incident Blotter dataset. For example, for the participation index threshold PI_{min} equal to 0.015, the numbers of PI-strong ST sequential patterns and PI-strong closed ST sequential patterns are 2,819,490 and 2,040,303, respectively. For the same value of the PI_{min} parameter and $\varepsilon = 0.15$, CSTS-Miner provided 1,171,955 PI-strong CSTS patterns. Thus, the reduction in the number of patterns is 58% when compared to the STBFM algorithm as well as 43% when compared to the CST-SPMiner algorithm.

Finally, in Table 9 we present the results obtained for the *reduced* Boston Crime Incident Report dataset. The reduction in the number of discovered patterns is even more impressive in the case of this dataset. For the values of participation index threshold $PI_{min} = 0.005$ and approximation margin $\varepsilon = 0.15$, CSTS-Miner provided 65 899 PI-strong CSTS patterns. For the same PI_{min} value, the numbers of PI-strong ST sequential patterns and PI-strong closed ST sequential patterns discovered are 228 285 and 76 894 patterns, respectively. Thus, the reduction in the number of discovered patterns by CSTS-Miner when compared to STS-Miner, CSTPM, STBFM is 71% and when compared to CST-SPMiner is 24%. However, it is worth noting that the computation time for the reduced dataset can be significantly higher in the case of the CSTS-Miner algorithm than in the cases of the STBFM and CST-SPMiner algorithms. Also, for the reduced Boston Crime Incident Report dataset, CSTPM executes much longer than the other algorithms.

In Fig. 5, we provide the plots presenting the percent of the number of the discovered PI-strong CSTS patterns to the number of the discovered PI-strong ST sequential patterns. As it can be noted from the presented plots, the percent ranges between 15% and 90%. It was possible to obtain minimal values of the percent (around 15%) for the reduced Boston Crime Incidents Report dataset when the spatial threshold $R = 600$ meters as well as temporal window $T = 28800$ minutes were applied and the PI_{min} threshold was equal to the values in the range $0.29 - -0.28$. In the case of both Pittsburgh Crime Incident Blotter and complete Boston Crime Incidents datasets, for the smaller values of the PI_{min} threshold (< 0.3) and the approximation margin ε equal to 0.2, the obtained percent is usually below 50%. Interestingly, not always the smaller value of the PI_{min} threshold resulted in a more significant reduction of the number of

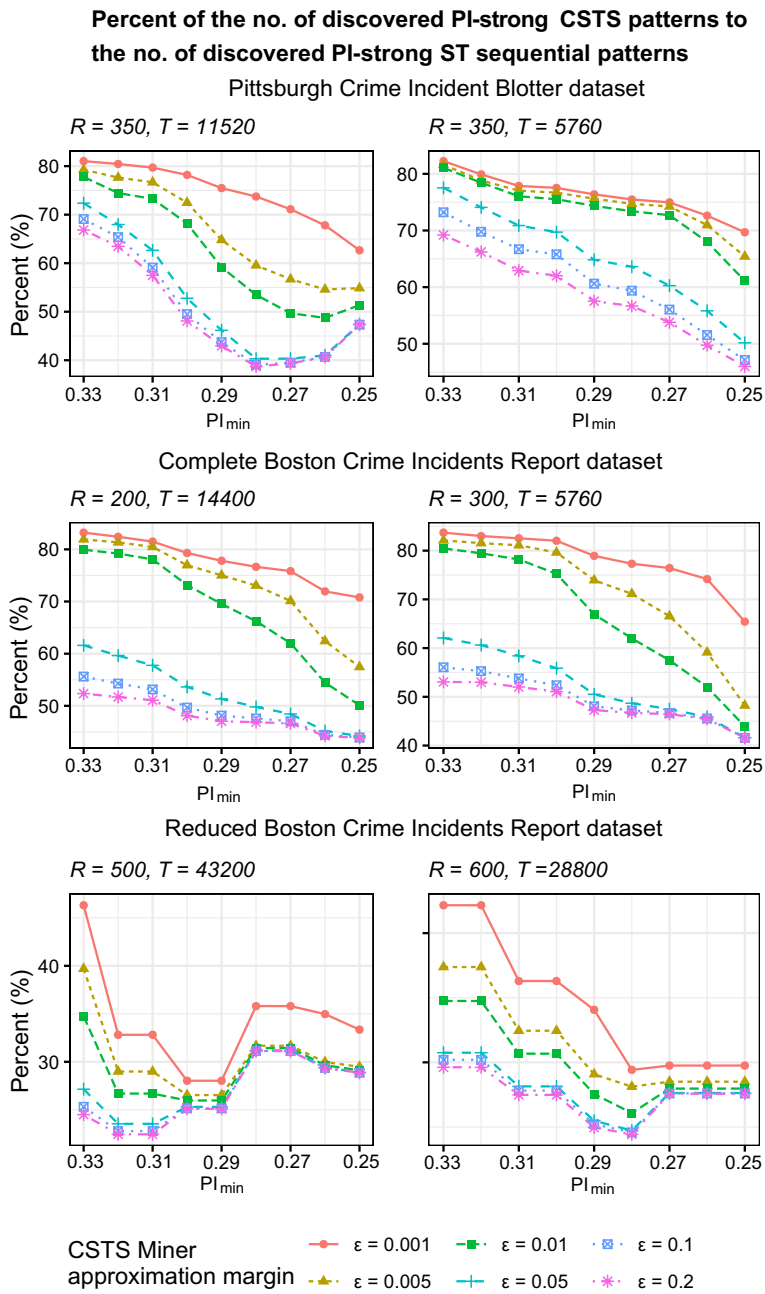

Fig. 5 Percent of the number of PI-strong CSTS patterns discovered by the CSTS-Miner algorithm to the number of PI-strong ST sequential patterns discovered by the STS-Miner, STBFM and CSTPM algorithms

discovered patterns. For example, for the Pittsburgh Crime Incident Blotter dataset and parameters $R = 350$ meters, $T = 11,520$ minutes as well as approximation margin ε equal to 0.2 or 0.1 the smallest percent was obtained for $PI_{min} = 0.28$.

In Fig. 6, we provide the plots presenting the percent of the number of PI-strong CSTS patterns to the number of PI-strong closed ST sequential patterns. As follows from Lemma 1, for $\varepsilon = 0$ the set of PI-strong CSTS patterns is equal to the set of PI-strong closed ST sequential patterns. However, for the greater values of ε (such as, for example

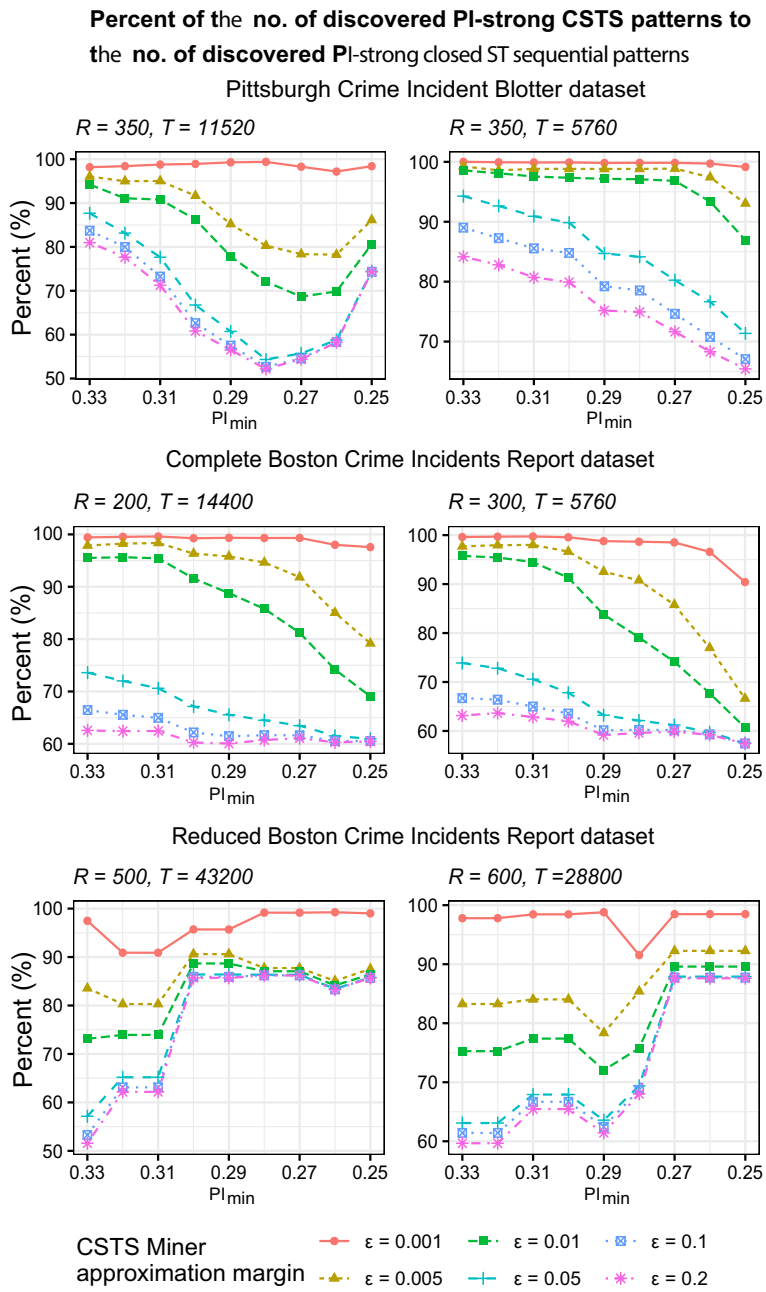

Fig. 6 Percent of the number of PI-strong CSTS patterns patterns discovered by the CSTS-Miner algorithm to the number of PI-strong closed significant ST sequential patterns discovered by the CST-SPMiner algorithm

0.1 or 0.2), CSTS-Miner is capable of providing as few as 50% of the number of PI-strong closed ST sequential patterns discovered by CST-SPMiner. Importantly, even for the smaller values of the ε parameter (such as, for example, 0.01), CSTS-Miner provided as few as 70% of the number of patterns provided by CST-SPMiner (as it is shown, for example, in the plots for the complete Boston Crime Incidents Report dataset presented in Fig. 6).

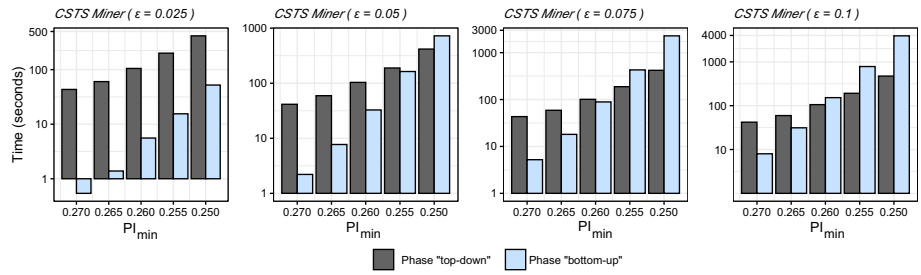

Fig. 7 Computation time in seconds of the "top-down" phase (steps 1–17) and the "bottom-up" phase (steps 18–24) of Algorithm 1 for the Pittsburgh Police Incident Blotter Dataset. Please note the logarithmic scale for the computation time

To summarize, the plots presented in Figs. 5 and 6 show that the CSTS-Miner algorithm can discover significantly fewer PI-strong CSTS patterns than PI-strong ST sequential patterns and PI-strong closed ST sequential patterns, even when the small values of the ε parameter are applied.

In our next experiment, we aimed to compare the computation times of steps 1–18 (phase "top-down") and steps 19–25 (phase "bottom-up") of Algorithm 1. As previously noted, the "top-down" phase is responsible for generating all PI-strong ST sequential patterns, while the "bottom-up" phase finds those patterns which are also PI-strong CSTS patterns.

In Fig. 7, we present the comparison of computation times (using the logarithmic scale) of the both phases obtained for the Pittsburgh Police Incident Blotter dataset. Please note that for the smaller values of the approximation margin ε (such as $\varepsilon = 0.025$), the computation times for the "top-down" phase are more significant than for the "bottom-up" phase. However, with the increasing values of ε and for the smaller values of PI_{min}, the computation time of the "bottom-up" phase can increase significantly. For example, for the parameters ε equal to 0.1 and PI_{min} equal to 0.25, the computation time of the "bottom-up" phase of Algorithm 1 can be up to four times longer than the computation time of the "top-down" phase for the same values of these parameters.

The results presented in Fig. 7 are in-line with the computational and space complexity analysis presented in Sect. Theoretical properties of CSTS patterns. In particular, for the increasing values of ε the computational cost of the "Bottom-up" phase of Algorithm 1 can be much higher than the computational cost of phase "Top-down". This is the result of the fact that the "Bottom-up" time complexity: $O\big((T-1)\cdot|L_k|\cdot 2^{(1-PI_{min})\cdot\varepsilon\cdot T}\cdot|\mathcal{C}^{max}(\overrightarrow{s})|\big)$ contains exponential function with the base of the function equal to 2 and exponent assessed by us as $(1-PI_{min})\cdot\varepsilon\cdot T$.

Representative patterns selection

In this subsection, we provide some interesting examples of discovered sequential patterns of different types of crimes. To this end, we ran CSTS-Miner using the Pittsburgh Police Incident Blotter Dataset with the following parameters: $R = 300$, $T = 11,520$ minutes (8 days), $\varepsilon = 0.05$, $PI_{min} = 0.3$.

The examples of interesting resultant patterns include:

1 *public_drunkenness* → *robbery/ bank/ knife* (PI = 0.5).
2 *public_drunkenness* → *robbery/ bank/ strongarm* (PI = 0.44).
3 *simple_assault/ injury* → *public_drunkenness* → *public_drunkenness* → *all_other_offenses (expt_traff)* → *fail_ disord_per_to_disperse* (PI = 0.30).
4 *simple_assault/ injury* → *public_drunkenness* → *robbery/ bank/ _strongarm* (PI = 0.33).
5 *robbery/ highway/ gun* → *sale/ use_of_air_rifles* (PI = 0.5).

Patterns 1 and 2 could provide essential information about types of banks robberies. Pattern 1 states that half of the bank robberies using a knife were conducted within 300 ms from the reported public drunkenness incidents and up to 8 days after they occurred. Similarly, pattern 2 communicates that 44% of all bank robberies using weapon (strongarm) occurred within 300 ms from the reported public drunkenness incidents and up to 8 days after they occurred. Another interesting example is pattern 4, which states that half of the usage of air rifles occurred within 300 ms from the highway robbery incidents and up to 8 days after they happened.

Conclusion

In this article, we offered a new type of ST sequential patterns called ε-constricted ST sequential patterns (CSTS patterns) and we thoroughly analyzed their theoretical properties. Specifically, we showed that a set of CSTS patterns is a subset of the set of closed ST sequential patterns and that each CSTS pattern is also a closed ST sequential pattern. Moreover, we showed that given the set of PI-strong CSTS patterns one can obtain the set of all PI-strong ST sequential patterns and approximate participation index of each of them with the approximation margin $\pm\,\varepsilon$. We also offered a new algorithm called CSTS-Miner that discovers all PI-strong CSTS patterns. CSTS-Miner adapts the MAX-Tree structure for more efficient candidate patterns generation. The proposed MAX-Tree is generated in two main phases of CSTS-Miner: the first one called "top-down" in which all PI-strong ST sequential patterns are generated using the breadth-first strategy, and the second one called "bottom-up" which calculates PI-strong ST sequential patterns being CSTS patterns. We analyzed properties and computation times of CSTS-Miner.

The experiments with the CSTS-Miner algorithm were conducted using two crime-related datasets for the cities of Boston and Pittsburgh: the Pittsburgh Police Incident Blotter Dataset and the Boston Crime Incident Reports Dataset. Each of the selected datasets consists of various types of crime and numerous event instances. To better verify the capabilities of the proposed algorithm, we also extracted a reduced dataset from the complete Boston Crime Indecent Reports dataset. The resultant reduced dataset contains 10 least frequent crime event types and 896 event instances.

During the experimental evaluation, we compared the results obtained with the proposed CSTS-Miner algorithm to four other state-of-the-art algorithms: STS-Miner [9], CSTPM [10], STBFM [11] and CST-SPMiner [12]. The STS-Miner, CSTPM and STBFM algorithms discover PI-strong ST sequential patterns, whereas CST-SPMiner discovers PI-strong closed ST sequential patterns. Each of the selected algorithms, as well as the proposed algorithm, use the participation index to measure the significance of the discovered patterns. As we presented in the experiments, in the case of the Pittsburgh

Police Incident Blotter Dataset and in the cases of the complete and reduced Boston Crime Incident Reports Dataset, CSTS-Miner can return much fewer patterns than the other selected algorithms. In particular, in the case of the Pittsburgh Police Incident Blotter Dataset, CSTS-Miner provides up to 60% fewer patterns compared to STBFM and up to 50% fewer patterns compared to CST-SPMiner. Similarly, for the complete Boston Crime Incident Reports Dataset, CSTS-Miner provides up to 60% fewer patterns compared to STBFM and up to 40% fewer patterns compared to CST-SPMiner. For the reduced Boston Crime Incident Reports Dataset, CSTS-Miner provides up to 85% fewer patterns than STBFM and up to 50% fewer patterns than CST-SPMiner.

Author contributions

PM proposed the notion of CSTS patterns, analyzed its theoretical properties, designed Algorithm 3 and provided its description, selected datasets for the experiments and conducted the experiments as well as their analysis. RB contributed to the writing of the manuscript, analysis of the proposed notions, algorithms and results of the experiments. AD contributed to the writing of the manuscript and analysis of the results of the experiments. All authors read and approved the final manuscript.

Funding

This work was supported by the RENOIR (Reverse Engineering of Social Information Processing) program (Grant no. 691,152) and by the Institute of Computer Science of Warsaw University of Technology.

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on a reasonable request.

Declarations

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 5 April 2022 Accepted: 17 May 2023
Published online: 11 June 2023

References

1. Han J, Pei J, Kamber M. Data mining: concepts and techniques. Elsevier; 2011.
2. Zaki MJ, Meira W Jr, Meira W. Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press; 2014.
3. Hu Z, Wang L, Tran V, Chen H. Efficiently mining spatial co-location patterns utilizing fuzzy grid cliques. Inf Sci. 2022;592:361–88.
4. Buczak AL, Gifford CM. Fuzzy Association Rule Mining for Community Crime Pattern Discovery. In: ACM SIGKDD Workshop on Intelligence and Security Informatics. ISI-KDD '10. New York, NY, USA: Association for Computing Machinery; 2010. Available from: https://doi.org/10.1145/1938606.1938608.
5. Yu CH, Ding W, Morabito M, Chen P. Hierarchical spatio-temporal pattern discovery and predictive modeling. IEEE Trans Knowl Data Eng. 2016;28(4):979–93.
6. He J, Zheng H. Prediction of crime rate in urban neighborhoods based on machine learning. Eng Appl Artif Intell. 2021;106: 104460.
7. Wu J, Abrar SM, Awasthi N, Frias-Martinez E, Frias-Martinez V. Enhancing short-term crime prediction with human mobility flows and deep learning architectures. EPJ Data Sci. 2022;11(1):53.
8. Dao THD, Thill JC. CrimeScape: analysis of socio-spatial associations of urban residential motor vehicle theft. Soc Sci Res. 2022;101: 102618.
9. Huang Y, Zhang L, Zhang P. A framework for mining sequential patterns from spatio-temporal event data sets. IEEE Trans Knowl Data Eng. 2008;20(4):433–48.
10. Mohan P, Shekhar S, Shine JA, Rogers JP. Cascading spatio-temporal pattern discovery. IEEE Trans Knowl Data Eng. 2012;24(11):1977–92.
11. Maciąg PS, Bembenik R. A Novel Breadth-first Strategy Algorithm for Discovering Sequential Patterns from Spatio-temporal Data. In: Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods—Volume 1: ICPRAM, INSTICC. SciTePress; 2019. p. 459–466.

12. Maciąg PS, Kryszkiewicz M, Bembenik R. Discovery of closed spatio-temporal sequential patterns from event data. In: Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary; 2019. p. 707–716. https://doi.org/10.1016/j.procs.2019.09.226.
13. He Z, Deng M, Cai J, Xie Z, Guan Q, Yang C. Mining spatiotemporal association patterns from complex geographic phenomena. Int J Geogr Inf Sci. 2020;34(6):1162–87. https://doi.org/10.1080/13658816.2019.1566549.
14. Andrzejewski W, Boinski P. Maximal mixed-drove co-occurrence patterns. Inf Syst Front. 2022;p. 1–24.
15. Aydin B, Angryk RA. Spatiotemporal event sequence mining from evolving regions. In: 2016 23rd International Conference on Pattern Recognition (ICPR); 2016. p. 4172–4177.
16. Yan X, Han J, Afshar R. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In: Proceedings of the 2003 SIAM International Conference on Data Mining; 2003. p. 166–177.
17. Wang J, Han J, Li C. Frequent closed sequence mining without candidate maintenance. IEEE Trans Knowl Data Eng. 2007;19(8):1042–56.
18. Wang J, Han J. BIDE: efficient mining of frequent closed sequences. In: Proceedings. 20th International Conference on Data Engineering; 2004. p. 79–90.
19. Fumarola F, Lanotte PF, Ceci M, Malerba D. CloFAST: closed sequential pattern mining using sparse and vertical id-lists. Knowl Inf Syst. 2016;48(2):429–63.
20. Gomariz A, Campos M, Marin R, Goethals B. ClaSP: an efficient algorithm for mining frequent closed sequences. In: Pei J, Tseng VS, Cao L, Motoda H, Xu G, editors. Advances in knowledge discovery and data mining. Berlin, Heidelberg: Springer, Berlin Heidelberg; 2013. p. 50–61.
21. Tzvetkov P, Yan X, Han J. TSP: mining top-K closed sequential patterns. In: Third IEEE International Conference on Data Mining; 2003. p. 347–354.
22. Zhang J, Wang Y, Yang D. CCSpan: mining closed contiguous sequential patterns. Knowl-Based Syst. 2015;89:1–13.
23. Cong S, Han J, Padua D. Parallel Mining of Closed Sequential Patterns. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. KDD '05. New York, NY, USA: Association for Computing Machinery; 2005. p. 562-567.
24. Fournier-Viger P, Lin JCW, Kiran RU, Koh YS, Thomas R. A survey of sequential pattern mining. Data Sci Pattern Recogn. 2017;1(1):54–77.
25. Wong AKC, Zhuang D, Li GCL, Lee ESA. Discovery of Delta closed patterns and noninduced patterns from sequences. IEEE Trans Knowl Data Eng. 2012;24(8):1408–21.
26. Kisilevich S, Mansmann F, Nanni M, Rinzivillo S. In: Maimon O, Rokach L, editors. Spatio-temporal clustering. Boston: Springer; 2010; 855–74.
27. Li Z. Spatiotemporal pattern mining: algorithms and applications. Cham: Springer International Publishing; 2014. p. 283–306.
28. Sunitha G, Reddy M, Rama A. Mining frequent patterns from spatio-temporal data sets: a survey. J Theor Appl Inf Technol. 2014;68(2).
29. Maciąg PS. A survey on data mining methods for clustering complex spatiotemporal data. In: Kozielski S, Mrozek D, Kasprowski P, Małysiak-Mrozek B, Kostrzewa D, editors. Beyond databases, architectures and structures towards efficient solutions for data analysis and knowledge representation. Cham: Springer International Publishing; 2017. p. 115–26.
30. Maciąg PS. Efficient Discovery of Top-K Sequential Patterns in Event-Based Spatia-Temporal Data. In: 2018 Federated Conference on Computer Science and Information Systems (FedCSIS); 2018. p. 47–56.
31. Atluri G, Karpatne A, Kumar V. Spatio-temporal data mining: a survey of problems and methods. ACM Comput Surv. 2018;51(4):83:1-83:41.
32. Ansari MY, Ahmad A, Khan SS, Bhushan G, et al. Spatiotemporal clustering: a review. Artif Intell Rev. 2019;1–43.
33. Aydin B, Boubrahimi SF, Kucuk A, Nezamdoust B, Angryk RA. Spatiotemporal event sequence discovery without thresholds. Geoinformatica. 2020;1–29.
34. Arge L, Procopiuc O, Ramaswamy S, Suel T, Vitter JS. Scalable Sweeping-Based Spatial Join. In: Proceedings of the 24rd International Conference on Very Large Data Bases. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1998. p. 570–581.
35. Pittsburgh Police Department. Pittsburgh Police Incident Blotter; 2020. Date accessed: 28.10.2020. http://plenar.io/explore/event/police_incident_blotter_archive.
36. UCR program. Uniform Crime Reporting program; 2020. Date accessed: 28.10.2020. https://www.fbi.gov/services/cjis/ucr.
37. Boston Police Department. Crime Incident Reports; 2014. Date accessed: 25.05.2018. http://plenar.io/explore.

Publisher's Note