Discovering Tight Space-Time Sequences

Riccardo Campisano, Heraldo Borges, Fábio Porto, Fabio Perosi, Esther Pacitti, Florent Masseglia, Eduardo Ogasawara

Discovering Tight Space-Time Sequences

Riccardo Campisano[1,2], Heraldo Borges[1], Fabio Porto[3], Fabio Perosi[4], Esther Pacitti[5], Florent Masseglia[5], and Eduardo Ogasawara[1]

[1] CEFET/RJ - Federal Center for Technological Education of Rio de Janeiro
[2] LIneA - Inter-institutional Laboratory for e-Astronomy
[3] LNCC - National Laboratory of Scientific Computing
[4] UFRJ - Federal University of Rio de Janeiro
[5] Inria & University of Montpellier

Abstract. The problem of discovering spatiotemporal sequential patterns affects a broad range of applications. Many initiatives find sequences constrained by space and time. This paper addresses an appealing new challenge for this domain: find tight space-time sequences, *i.e.*, find within the same process: i) frequent sequences constrained in space and time that may not be frequent in the entire dataset and ii) the time interval and space range where these sequences are frequent. The discovery of such patterns along with their constraints may lead to extract valuable knowledge that can remain hidden using traditional methods since their support is extremely low over the entire dataset. We introduce a new *Spatio-Temporal Sequence Miner* ($STSM$) algorithm to discover tight space-time sequences. We evaluate $STSM$ using a proof of concept use case. When compared with general spatial-time sequence mining algorithms ($GSTSM$), $STSM$ allows for new insights by detecting maximal space-time areas where each pattern is frequent. To the best of our knowledge, this is the first solution to tackle the problem of identifying tight space-time sequences.

1 Introduction

Space and time are pervasive in our day-to-day lives. As many datasets that include both time and space data are becoming available, new opportunities to discover interesting spatiotemporal patterns arise. An event may be classified as an occurrence of a phenomenon in a given space and time. A spatiotemporal sequential pattern is a sequence of events that are constrained in space and time [8]. Due to that, spatiotemporal sequence mining is gaining attention [13,14].

In this work, we investigate a new problem related to spatiotemporal pattern identification. We are interested in finding tight space-time sequences, *i.e.*, sequences that are constrained in space and time that may not be frequent in the entire dataset but are frequent inside a time interval and space range (spatiotemporal blocks). The primary challenge is to discover these blocks and the frequent sequences they contain. Solving this problem has a valuable impact on many applications.

Consider, for instance, the quantified-self movement, where people wear connected bracelets giving their position and inferring their activities. By analyzing the activities of their clients' bracelets, a brand might discover some habits regarding sports and food at coarse grain (say, "people who jog in the morning step by a vegan shop once a week"). However, there might be fine-grained behaviors that cannot be extracted, using existing methods, because they concern a niche (*i.e.*, they have extremely low support, that hinders their discovery). Such a niche could be, for instance, that "people in Manhattan who jog at 7 am and have lunch near Time Square, spend 1 to 2 minutes in front of the buildings displays". This kind of pattern is of high interest for marketing and personalized recommendations. In the meantime, the accumulation of such niches, particularly in massive data, makes their discovery a challenging and valuable target for data analytics. The challenge is to extract both the pattern (*e.g.*, jog, lunch, buildings displays), its occurrence time (*e.g.*, from 7 am to lunchtime), and the location where it occurs (*e.g.*, Time Square).

This paper introduces the problem of discovering tight space-time sequences and proposes an effective and efficient solution with the *Spatio-Temporal Sequence Miner* ($STSM$) algorithm to address it. $STSM$ applies a sequential pattern mining algorithm to detect spatial ranges where sequences are frequent. Next, it composes all detected sequences inside each spatial range to discover time intervals in which these sequences are frequent.

Our group first experienced the spatiotemporal sequence mine problem while analyzing seismic surveys [5]. Since then we have developed the $STSM$ algorithm and used the seismic motivation as our use case for validating the technique. By applying constrained space-time sequence mining techniques, we managed to identify 1,500 tight space-time sequences under high support threshold, which would not have been found using general spatial-time sequence mining algorithms ($GSTSM$) [14] for the same frequency threshold. Moreover, the algorithm was able to detect candidate areas for seismic *horizons* (which stand for kind of layers) and *bright spots* (potential areas of hydrocarbon accumulation), which are known as important characteristics for the domain experts [17].

In addition to this introduction, this paper is organized into five more sections. Section 2 discusses related works. Sections 3 and 4 formalize the problem and present the definitions needed to describe our algorithm. Section 5 gives the $STSM$ algorithm to identify sequential pattern constrained in space and time. Section 6 details the experimental evaluation. Finally, Section 7 concludes the paper.

2 Related works

Sequential pattern mining is a broad field of research embracing several approaches [13,7]. Over the last decades, some works have been developed for sequential mining of spatiotemporal datasets [14]. We have conducted a systematic map study querying Scopus using the keywords ("*sequence mining*" $\vee$ "*sequential pattern*") $\wedge$ ("*space-time*" $\vee$ "*spatiotemporal*"). We identified 98

articles that were related to spatiotemporal sequential pattern mining, which are grouped according to different types of datasets: (i) trajectory datasets, (ii) event-based datasets, and (iii) emerging patterns datasets.

Trajectories datasets can be classified as a collection of events of the same moving object at different spatial locations and times. The goal is to retrieve similar trajectories that might reveal underlying traveling patterns of moving objects in the dataset [8]. It includes works that search for routes frequently used [6]. Since commonly the position of moving objects may be inaccurate, some approaches address the position uncertainty to better recognize common trajectories [11]. Although trajectory problems are relevant, they differ from our problem. We are not interested in moving objects that have the same behavior. Instead, we are interested in regions and time intervals when some events are related (constrained) and relevant (with high support).

Event-based datasets correspond to the majority of related work. The goal is to find sequences constrained by space and time. For that, data is partitioned according to spatial or temporal dimensions, and events are related whenever they preserve certain proximity [16]. Huang et al. [8] define sequence index as the significance measure for evaluating identified spatiotemporal sequential patterns. Such approach improves the mining algorithm to identify constrained sequences. Julea et al. [9] studied satellite images and target patterns with a high connectivity measure. Alatrista-Salas et al. [1] expand the search-space for identifying sequences by looking at neighbor partitions. Li et al. [10] address the problem of event prediction from identified spatiotemporal sequence patterns. Finally, Batu et al. [3] avoid partitioning space and time by computing the most relevant sequences in the dataset. Although approaches for event-based datasets are in the same category of our paper, they differ since all identified sequences are frequent in the entire dataset. In our work, sequences may only be frequent inside spatiotemporal blocks (*i.e.*, a time interval and space range).

Finally, emerging pattern datasets correspond to solve the problem where data are continuously added to the database. Consider the patterns identified in the initial dataset and the patterns that are identified in an updated dataset. Previously identified patterns may become irrelevant, and new patterns may emerge. Some initiatives in emerging spatiotemporal datasets have been developed so far [4,15,2]. These problems are complementary to ours since all identified patterns in both datasets (initial or updated) have high support. The problem tackled in this paper may find tight space-time sequence in both datasets, which may emerge or not.

Considering the aforementioned related works, to the best of our knowledge, we are not aware of similar works that studied tight space-time sequences from spatiotemporal datasets.

3 Formalization

Let $t = <v_1, v_2, \cdots, v_n>$ be a **time-stamped sequence (TS)**, *i.e.*, a **sequence** of items, where $|t| = n$ is the number of items in t. A time index

j is an integer value between 1 and n that is related to item v_j. A sequence $s = <w_1, w_2, \cdots, w_k>$ is **included** in another sequence $t = <v_1, v_2, \cdots, v_n>$ if there exist integers $i_1 < i_2 < \cdots < i_k$ such that $w_1 = v_{i_1}, w_2 = v_{i_2}, \cdots, w_k = v_{i_k}$. Let $P = \{p_1, p_2, ..., p_m\}$ be a set of positions, a **spatial time-stamped sequence (STS)** d is a couple (p, t) where $p \in P$ is a position and t is the associated TS. A **STS dataset** D is a set of STS.

An STS $d = (p, t)$ is said to support a sequence s if s is included in t. The **support** of a sequence s in D is the number of STS in D in which s is included. The **frequency** of a sequence s in D is the fraction of STS in D that supports s: $freq(s, D) = \frac{sup(s,D)}{|D|}$. Given a user's minimum threshold $\gamma \in]0..1]$, a sequence s is said to be **frequent** if $freq(s, D) \geq \gamma$. Let I be the set of all possible sequences s included in D, **the set F of frequent sequences** in D concerning γ is denoted by $F(D, \gamma) = \{s \in I : freq(s, D) \geq \gamma\}$.

A **spatial range** (or simply **range**) $r = (p_s, p_e)$ is defined by a start position p_s and an end position p_e. We define the set of all potential ranges over D as PR. The set of STS that belong to range r is defined as $Tr(r) = \{d : d \subseteq D, p_s \leq d.p \leq p_e\}$. The frequency of s over $Tr(r)$ in STS of a range r is denoted by $freq(s, r)$ whenever it is clear from the context. The same approach is valid for support of s over $Tr(r)$, which is denoted by $sup(s, r)$. A **ranged sequence** sr is a triple (s, r, fr) where s is a *sequence*, r is a *range*, and fr is the *frequency* of the *sequence* s over the *range* r: $fr = freq(s, r)$. Let k be the size of s; then s is called a $k-$sequence.

A **time interval** (or simply **interval**) $i = (i_s, i_e)$ is defined by a start time i_s and an end time i_e. The length of an interval i is given by: $|i| = i_e - i_s + 1$. Given a interval i, a sequence $s = <w_1, w_2, \cdots, w_k>$ is a **subsequence** of another sequence $t = <v_1, v_2, \cdots, v_n>$: $s = subseq(t, i)$ iff $i_s \geq 1 \wedge i_e \leq n$, $|i| = k$ and $\forall j \in [1..k], w_j = v_{i_s+j-1}$. By definition, s is also included in t. We define the set of all possible intervals over D as PI. A **block** b is a couple (r, i) where r is a range ($r \in PR$) and i is an interval ($i \in PI$). The size of a block b is the product of range size with interval length: $|b| = |b.r| \cdot |b.i|$. We define the set of all possible blocks over a range r as $PB(r)$. Moreover, a **bounding box** between two blocks b_i, b_j: $bb(b_i, b_j)$ is the minimum block b that contains them.

The **occurrences** of a *sequence* s in a block b refer to the positions in which all instances of s is present in b. Thus, $occur(s, b)$ is a set of blocks o, such that $\forall b_j \in o, b_j \subseteq b \mid |Tr(b_j.r)| = 1 \wedge \forall d \in Tr(b_j.r), subseq(d, b_j.i) = s$. The **support** of a *sequence* s in a block b, $sup(s, b)$ is the number of occurrences of s in b: $|occur(s, b)|$. The **item-support** of a *sequence* s in a block b, is the product of support of a sequence s in a block b with the length of the sequence s: $isup(s, b) = sup(s, b) \cdot |s|$. The **item-frequency** of a sequence s in a block b is the fraction of item-support over the size of b: $ifreq(s, b) = \frac{isup(s,b)}{|b|}$. Given a user's minimum threshold $\delta \in]0..1]$, a sequence s is said to be **item-frequent** in a block b if $ifreq(s, b) \geq \delta$. A **blocked sequence** sb is a triple (s, b, ifr) where s is a *sequence*, b is a *block*, and ifr is the *item-frequency* of s over b: $ifr = ifreq(s, b)$.

4 Problem statement

Considering an STS dataset D, the problem we address is to find sequences in D that are frequent in constrained spatial range and time interval. The goal is to discover these ranges and intervals and the frequent sequences they contain.

In the following definitions, we introduce the notions of solid-ranged sequence and solid-blocked sequence that are fundamental for STSM algorithm. Their intuition is to respectively support the identification of spatial range and time-space blocks where a pattern is frequent.

5 STSM

This section is devoted to the presentation of $STSM$ (Spatio-Temporal Sequence Miner), our algorithm designed for the identification of solid-blocked sequences in the spatiotemporal dataset. The notion of kernels (range-kernels and block-kernels) introduced in this section allows for extracting solid-blocked sequences efficiently. First, we give an overview of our main algorithm in Section 5.2. Later, each relevant function is described in the following sections.

5.1 STS definitions

Given γ and δ, user's minimum thresholds, respectively, for frequency and item-frequency, we introduce the characteristics of solid-ranged sequence in Definition 1 and solid-blocked sequence in Definition 2.

Definition 1 *Let sr be a ranged sequence of range r, sequence s, and frequency fr. Then, sr is called a* **solid-ranged sequence** *iff the following conditions hold:*
1) $fr \geq \gamma$
2) $\forall r_2 \in PR \mid r \subseteq r_2$*, we have either a) or b) or both:*
a) $freq(s, r_2) < \gamma$
b) $sup(s, r_2) = sup(s, r)$
3) $\forall r_2 \in PR$ *such that* $r_2 \subset r$, $sup(s, r_2) < sup(s, r)$

The first condition in Definition 1 ensures that sr represents a sequence that is frequent over its associated range. The second condition ensures that the size of r is maximal. If a larger range exists, then, in this range, s is not frequent, or the support of s is the same (*i.e.*, it is not worth extending the range from r to r_2, since the extension is not going to contribute to the support of s). Finally, the third condition ensures that the size of r is minimal. In fact, s is supported by the first and last STS in r, so if a smaller range exists where s is frequent, the support is going to be lower anyway (*i.e.*, relevant STS supporting s would have been dropped from the range and should be kept).

Let k be the size of s then sr is a **k-solid-ranged sequence**. $\mathbf{SR_k}$ is the set of all k-solid-ranged sequences.

Definition 2 *Let sb be a blocked sequence with a block b, sequence s, and item-frequency ifr. Then, sb is called a* **solid-blocked sequence** *iff the following conditions hold:*
1) $\exists\ sr \in \mathbf{SR}_{|\mathbf{s}|} \mid b.r \subseteq sr.r$ *and* $s = sr.s$
2) $ifr \geq \delta$
3) $\forall b_2 \in PB(r) \mid b \subseteq b_2$*, we have either a) or b) or both:*
a) $ifreq(s, b_2) < \delta$
b) $isup(s, b_2) = isup(s, b)$
4) $\forall b_2 \in PB(r) \mid b_2 \subset b$, $isup(s, b_2) < isup(s, b)$
5) $sup(s, b) > 1$

The first condition ensures that the range of *sb* is within the range of a solid-ranged sequence *sr*. In other words, candidate blocks for solid-blocked sequences are identified from computed solid-ranged sequences. The second condition ensures that s corresponds to a sequence that is item-frequent in b. The third condition ensures that the size of b is maximal. If a larger block exists that contains b, s is not item-frequent, or the item-support of s is the same (*i.e.*, it is not worth extending the block from b to b_2, since the extension is not going to contribute to the item-support of s). The fourth condition ensures that the size of b is minimal. In fact, an item of s is present in the first and last range in $b.r$, and in the first and last interval in $b.i$, so if a smaller range or interval exists where s is solid-item-frequent, the support is going to be lower anyway. Finally, the fifth condition avoids trivial solid-blocked sequences that contain only a single occurrence of s in b.

Let k be the size of s; then *sb* is a **k-solid-blocked sequence**. $\mathbf{SB_k}$ is the set of all k-solid-blocked sequences.

5.2 General principle

Algorithm 1 aims to generate solid-blocked sequences (SB) from size 1 to k. It receives as input an STS dataset D, a solid range threshold γ and a solid block threshold δ. It has three main functions: (i) candidate generation, (ii) identification of solid-ranged sequences, (iii) identification of solid-blocked sequences. The algorithm starts from candidates ranged sequences of size 1 (detailed in Section 5.3). They are built from all distinct items presented in D considering its entire range.

It starts a repeat-until loop that computes all solid-ranged sequences SR_k with a frequency greater than or equal to γ (detailed in Section 5.4). Then, candidate ranged sequences of size $k+1$ are computed from solid ranges SR_k. Once we get empty candidates ranged sequences of size $k+1$, the loop stops.

Finally, all solid-blocked sequences SB_k with item-frequency greater than or equal to δ are computed (detailed in Section 5.5) from identified solid-ranged sequences.

Algorithm 1 Spatio-Temporal Sequence Miner

```
1: function STSM(D, γ, δ)
2:     C_1 ← generateCandidates(D, nil)
3:     k ← 0
4:     repeat
5:         k ← k + 1
6:         SR_k ← solidRangedSequences(D, C_k, γ)
7:         C_{k+1} ← generateCandidates(D, SR_k)
8:     until C_{k+1} ≠ ∅
9:     for (i ∈ {1 ··· k}) do
10:        SB_i ← solidBlockedSequences(D, SR_i, δ)
11:    return {SB_1, ··· , SB_k}
```

5.3 Generation of candidates

Frequent pattern mining algorithms aim at providing efficient algorithms on larger datasets. The generation of candidates should accomplish the mission: start with solid-ranged sequences of size 1 (SR_1) and explore the support of larger solid-ranged sequences with a limited number of scans over the dataset. To this end, we compute range frequencies for candidate solid-ranged sequences of size k from computed solid-ranged sequences of size $k-1$ in only one scan. At the end of each scan, solid-ranged sequences of size k are identified.

$STSM$ introduces a spatial aware counting step for generated candidates. Let us consider $xr(s_x, r_x, fr_x)$ to be a ranged sequence that is not a solid-ranged sequence. Any ranged sequence superset $yr = (s_y, r_y, fr_y)$, such that $s_x \subseteq s_y \wedge r_x \subseteq r_y$ cannot be a solid-ranged sequence (*i.e.*, since $fr_x < \gamma$ then $fr_y < \gamma$). Thus, the generate candidates follows an apriori-like principle with an additional filter on the possible intersection of the candidates (*i.e.*, if two solid-ranged sequences of size k have a common subsequence, but their ranges do not intersect, they are not considered for generating a new candidate).

5.4 Selection of solid-ranged sequences

In this section, we define a range kernel, which is the basis of solid-ranged sequences described in Algorithm 2. A **range kernel** for a sequence s is a range where its frequency is greater or equal to γ. Let $K(s, r, \gamma)$ be the set of range kernels for the sequence s over a range r concerning the minimum threshold γ. $K(s, r, \gamma)$ is defined as follows: Let $k \subseteq r$ be a subrange such that s is included in $Tr(k)$, *i.e.*, $sup(s, k) > 0$. Let k_s be the first position in which s is present in r. If k_s does not exist, then $K = \emptyset$. If k_s exists, then let P be the set of positions such that $\forall p \in P, p \in r \wedge p > k_s \wedge frequency(s, [k_s..p]) < \gamma$ (in other words, P is the set of positions in r such that extending the range k up to any of those positions leads to a frequency less than γ for s). If P is empty, then k_e is defined as the last position in which s is present in r, and $K(x, r, \gamma) = \{k\}$. Otherwise (*i.e.* $P \neq \emptyset$), let $q \in P$ such that $\forall p \in P, p > q$ (q is the first position such that

frequency of s is lost on $[k_s..q]$). Then, k_e is defined as the last occurrence of s in $[k_s..q]$ and $K(s, r, \gamma) = \{k\} \cup K(s, r - [k_s..k_e], \gamma)$.

The mechanics of computing range kernels is encapsulated by function $rangeK$ of Algorithm 2. Intuitively, range kernels of a sequence are the longest ranges such that: (i) The first and last records support the sequence; (ii) The frequency of the sequence is always greater than or equal to γ when it is counted from the beginning until the end of the range.

Algorithm 2 solid-ranged Sequences

```
1: function solidRangedSequences(D, C_k, γ)
2:     SR_k ← ∅
3:     for (c ∈ C_k) do
4:         c.sker ← rangeK(D, c, γ)
5:         c.sker ← mergeK(c.s, c.sker, mRS, freq, γ)
6:         for (r ∈ c.sker) do
7:             SR_k ← SR_k + (c.s, r, frequency(c.s, r))
8:     return SR_k
```

After extracting range kernels for all candidate sequences C_k, whenever possible, Algorithm 2 merges them. Algorithm $mergeK$ finds solid-ranged sequences of s over r concerning threshold γ. Given two range kernels t and u, Algorithm $mergeK$ tries to merge them into v using function $mRS(t, u)$ as long as $freq(s, v) \geq \gamma$ is satisfied, which simply combines their ranges: t, u. In this case, v is included in the $sker$ list, and both t and u are marked for removal. Such approach is repeated until no more pairs of kernels can be merged. Finally, in lines 6-8 of Algorithm 2, all merged kernels are used to produce solid-ranged sequences SR_k.

5.5 Selection of solid-blocked sequences

In this section, we define a block kernel, which is the basis of solid-blocked sequences described in Algorithm 3. A **block kernel** is a block b in a solid range sr that contains a set of occurrences of s in b such that the item-frequency for s in b is greater or equal to δ. Let $B(sr, \delta)$ be the set of all possible **block kernels** in sr that has item-frequency not less than δ: $B(sr, b) : b_k \in PB(sr) \wedge ifreq(s, b_k) \geq \delta$.

Given a block kernel $b \in B(sr, \delta)$, let o be the set of occurrences of sr in b: $o = occur(s, b)$. If $|o| = 1$, then according to this definition, $ifreq(s, o) \geq \delta$. We name this case as trivial block kernel. However, if $|o| > 1$, then there exists two block kernels b_i and $b_j (b_i \subset b, b_j \subset b)$, such that $ifreq(s, b_i) \geq \delta \wedge ifreq(s, b_j) \geq \delta \wedge o = occur(s, b_i) \cup occur(s, b_j) \wedge occur(s, b_i) \cap occur(s, b_j) = \emptyset$. Let $o = occur(s, b)$ of s in a block kernel b. Whenever $|o| > 1$, $(s, b, ifreq(s, b))$ is a solid-blocked sequence.

From the previous definition, Algorithm 3 computes all solid-blocked sequences. It starts computing all trivial block kernels. Then, Algorithm $mergeK$ finds solid-blocked sequences of s over sr concerning threshold δ. Given two block kernels t and u, Algorithm $mergeK$ tries to merge them into v using function $bb(t, u)$, which produces a bounding box for t and u. If v has an $ifreq$ for s greater than or equal to δ, v is included in the $stkernel$ list, and both t and u are marked for removal. Such approach is repeated until no more pairs of kernels can be merged. Finally, in lines 6-8 of Algorithm 3, all merged block kernels are used to produce solid-blocked sequences SB_k.

Algorithm 3 solid-blocked Sequences

```
1: function solidBlockedSequences(D, SR_k, δ)
2:     B ← ∅
3:     for (sr ∈ SR_k) do
4:         sr.stker ← spatialTimeK(D, sr, δ)
5:         sr.stker ← mergeK(sr.stker, bb, ifreq, δ)
6:         for (b ∈ sr.stker) do
7:             if (|occur(s, b)| > 1) then
8:                 B ← B + (s, b, freq(s, b.r), ifreq(s, b))
9:     return B
```

5.6 Merging kernels

Let us consider that we are provided with a sequence s and L, the set of kernels of s concerning a threshold σ. Algorithm $mergeK$ can be used to compute both solid-ranged sequences of s over r concerning γ and solid-blocked sequences of s over sr concerning δ. To do so, Algorithm $mergeK$ receives as input $mfunc$, which is a function that merges kernels, and $thresfunc$, which is a function that computes the frequency used to evaluate if a merged kernel respects threshold σ.

Proof for merge kernels Lemma: Let K be the set of range kernels of s on r concerning γ. Algorithm $mergeK$ makes it possible to find all the solid-ranged sequences $sr = (s, r, fr)$ on r.

Proof Let $k \in K$, be a range kernel of s after Algorithm $mergeK$ (*i.e.* k cannot be merged with any other range kernel in K), then:

1. $freq(s, k) \geq \gamma$. According to solid range definition, s is frequent in each kernel. Furthermore, if k is the result of a merging process, then Algorithm $mergeK$ checks the frequency of s on the resulting range.

2. $\forall q$ such that $k \subseteq q$, we have one of the following cases:

Algorithm 4 Algorithm Merge Kernels

```
1: function mergeK(s, L, mfunc, thresfunc, σ)
2:     mergeable ← true
3:     while mergeable do
4:         mergeable ← false
5:         for t, u ∈ L | u > t do
6:             v ← mfunc(t, u)
7:             if thresfunc(s, v) ≥ σ then
8:                 L ← L + v
9:                 K ← K + t + u
10:                mergeable ← true
11:        for k ∈ K do
12:            L ← L − k
13:        K ← ∅
```

- $sup(s, q-k) > 0$, then s is not frequent in q (otherwise, let us consider k' the kernel to which s is present in q, then k and k' would have been merged).
- $sup(s, q-k) = 0$, then $sup(x, q)$=$sup(x, k)$ (in this case, x may remain frequent in q or not, depending on the size of q).

3. According to Definition 2.1, s is supported by the first and the last STS in k. Then, s has lower support on any sub-range of k.

Based on the three observations above, let $SR_x = \{(s, r, fr), \forall r \in K)\}$ be the set of ranged sequences corresponding to all the merged kernels of s on PR concerning γ, then SR_x is the set of all solid-ranged sequences $sr = (s, r, fr)$ on PR concerning γ.

A similar proof is analogous to $mergeK$ applied in solid-blocked sequences.

5.7 Toy Example

Figure 1 shows an example of spatiotemporal dataset D to better illustrate the definitions and mechanics of $STSM$ algorithm. Each position p_i is associated with an STS d_i in dataset D. Let us consider a frequency threshold $\gamma = \frac{1}{2}$ given by the user. In this context, the only frequent sequence for D is $<a>$ with a frequency of $\frac{1}{2}$ using $GSTSM$.

This dataset presents other sequences that can be identified by $STSM$ according to our definitions. Consider the ranges $r_1([d_1..d_2])$, $r_2([d_8..d_{10}])$, $r_3([d_2..d_5])$, and $r_4([d_7..d_8])$. Both r_3 and r_4 are, respectively part of solid-ranged sequences $sr_3(<e, o>, r_3, \frac{3}{4})$ and $sr_4(<i, u>, r_4, 1)$. In fact, considering our generate candidate algorithm, solid-ranged sequence sr_3 is obtained from $(<e>, r_3, \frac{3}{4})$ and $(<o>, r_3, \frac{3}{4})$.

Additional, r_1 and r_2 are not solid-ranged sequences. They correspond to range kernels for $<a>$ identified by algorithm *solidRangedSequences*. Later, they are merged to build a solid-ranged sequence $(<a>, [d_1..d_{10}], \frac{1}{2})$. As it is,

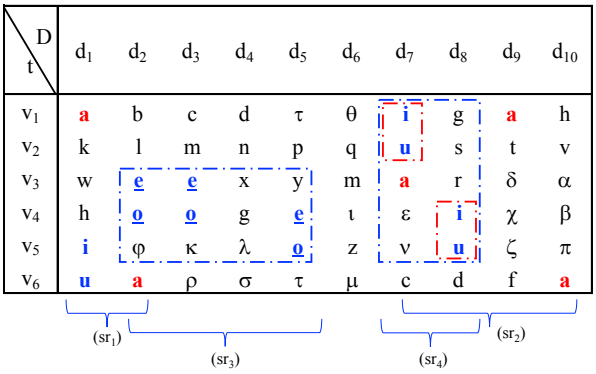


Fig. 1. Dataset D for ten spatial-time series with six observations in each.

the sequence $<a>$ may not appear to be interesting, but if frequency threshold is increased to $\gamma = \frac{2}{3}$, both r_1 and r_4 become part of a solid-ranged sequence for $<a>$, whereas no sequence is identified by general spatial-time sequence mining algorithms ($GSTSM$).

Taking into account the relationship between time and space, intuitively, sequence $<a>$ still does not appear to be interesting. In fact, the goal of $STSM$ is to identify tight space-time sequences. Consider blocks $b_1 = ([d_2..d_5], [v_3..v_5])$ and $b_2 = ([d_7..d_8], [v_1..v_5])$. The occurrences of $<i, u>$ in b_2 are $\{([d_7], [v_1..v_2]), ([d_8], [v_4..v_5])\}$. Adopting an item-frequency threshold $\delta = \frac{2}{5}$, both blocks are part of solid-blocked sequences $sb_1(<e, o>, b_1, \frac{1}{2})$ and $sb_2(<i, u>, b_2, \frac{2}{5})$. Additionally, sequence a is not produced by $STSM$ algorithm as a solid-blocked sequence for such threshold.

We can observe that using solid-ranged sequences and solid-blocked sequences definitions together with $STSM$ algorithm, it is now possible to find sequences that would not be found using higher frequency thresholds. According to the example, the sequence $<e, o>$ is supported by the entire dataset with a frequency of $\frac{3}{10}$ occurs on a particular range (*i.e.* $[d_2..d_5]$) and interval (*i.e.* $[v_3..v_5]$) with a frequency of $\frac{3}{4}$ and item-frequency of $\frac{1}{2}$. Also, it filters unwanted sequences that are not tight in space-time, such as $<i, u>$ in block $([d_1], [v_5..v_6])$. Sequence $<i, u>$ i is only solid-blocked sequence in b_2.

6 Experimental evaluation

This section presents the experimental evaluation. Section 6.1 starts presenting the dataset used as a proof of concept. Section 6.2 presents the exploratory analysis of $STSM$. Finally, Section 6.3 discusses the behavior of the algorithm $STSM$ when compared to $GSTSM$.

6.1 Dataset

The dataset chosen as input data is the inline 401 of the *Netherlands Offshore F3 Block* reflection seismic survey [5]. It is composed of 651 *inlines*. The *inline* contains 951 spatiotemporal series of timely-ordered observations collected in different positions on the surface. Each of spatiotemporal series contains 462 observations, a sample of every four milliseconds, representing the different depths of the subsurface.

Seismic surveys are commonly collected by producing pertubations (shots) on the ground or in the ocean. Such shots generate elastic waves that are propagated in the Earth interior. They are reflected and refracted in each material layer and are returned to the surface in a given position and time. A set of receivers positioned on the surface (onshore or offshore) at fixed aligned intervals records the waveform and wave arrival times. The collected wave amplitudes and frequencies are related to the reflection location, and the wave traveled distance. Thus, the early received signals correspond to upper subsurface layers, and vice-versa. As it can be depicted in Figure 2, for example, each spatiotemporal series can be interpreted as vertical columns of pixels, where negative amplitude values correspond to lighter pixels whereas positive amplitude values correspond to darker pixels. The inline was made available[6] using SAX with an alphabet of size 7 [12].

6.2 Exploratory Analysis

The algorithm proposed in this work allows for users to set solid range threshold γ and solid block threshold δ constraints. Finding adequate values for these thresholds depends on the characteristics of input dataset/application. Lower values for such constraints can lead to the identification of a large number of non-useful frequent sequences. Conversely, higher values for these thresholds can result in the detection of a small number of frequent sequences that may become too small to be interesting. All code, experimental data, and results are made available[6].

We explored the combination of solid range threshold γ (60%, 70%, 80%, 90%, and 100%) with solid block threshold δ (10%, 20%, 30%, 40%, and 50%). For sake of comparison, consider the identified sequence $<a, a, g, g>$. In this scenario, when γ is under 80%, a single solid range for this sequence is detected covering the entire dataset. Values between 80% and 100% divide the dataset into more fine-grained solid ranges. A consequence of having better solid range division is that the computation required for identifying solid-blocked sequences is reduced. Relaxing γ from 80% to 60% led to an increase 18% in computing time without improving the detection of a solid-blocked sequence. Adopting the extreme value of 100% may, however, limit too much the detection of solid block sequences. Figure 2.a depicts sequence occurrences for $\gamma = 80\%$.

Similarly, examining the solid block threshold δ, when relaxing it to 10%, many larger solid-blocked sequences are identified with many overlapping among

[6] https://github.com/eogasawara/stsm

them. Conversely, when increasing the δ threshold to higher values (30%), it is possible to observe a significant decrease in the number of identified solid-blocked sequences, many of them lose their overlapping. In Figure 2.a, δ is set to 20%.

6.3 Discussions

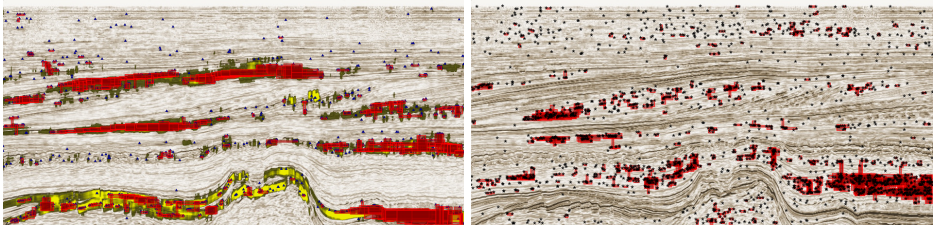
The goal of this paper is to provide a way to discover sequences that are frequent in tight time and space. By applying $STSM$, we managed to identify sequences that would not have been found using traditional spatiotemporal techniques using the same support. We ranked the identified sequences by density. The density of a sequence s was computed by the mean block size of all solid-blocked sequences for s. A representative high-ranked sequence ($<e, e, f>$) is depicted in Figure 2.b.

Despite the simplicity of used ranking and the absence of any significant seismic concept in its logic, it was good enough to prioritize interesting areas. In the seismic domain, it is of particular importance to identify *horizons*, *i.e.*, zones of major nonconformities between geologic boundary materials, usually associated to different lithologies, which produces variations in the collected reflection values [17]. $STSM$ algorithm was able to identify known *horizons*. The majority of high-ranked sequences made available in our site match known seismic *horizons* previously known for this dataset [5]. Additionally, $STSM$ was also able to find potential areas of accumulation of hydrocarbon, known as *bright spots*. *Bright spots* are rare patterns that occur when there is an inversion of the wave phase. Such pattern is important not only because gas can be interesting for exploration, but also since it can be near to rocks containing oil. For example, in Figure 2.a, the discovered patterns, evidenced in red, are plotted on the top of yellow marks obtained from the previously known *bright spots* for this dataset [5]. Figure 2.a shows that $STSM$ algorithm has discovered patterns that cover large parts of known *bright spots*.

Figure 2.b shows a comparison between $GSTSM$ and $STSM$. The sequence $<e, e, f>$ is identified by both algorithms. In $GSTSM$ it has high support, whereas in $STSM$ it is mid-term ranked results obtained the chosen configuration. It can be observed that although many solid-blocked sequences (depicted in red) are covering the majority of known horizons, many isolated occurrences identified by $GSTSM$ (marked as black) correspond to noise. Considering Figure 2, it is possible to conclude two important differences between $GSTSM$ and $STSM$. Not only $STSM$ can identify sequences that $GSTSM$ is not capable of identifying without lowering too much the frequency threshold, but it also can filter-up sequence occurrences regarding space and time. Such approach allows for researchers to focus on tight sequences.

7 Conclusion

This paper presented a challenging problem with high potential impact: spatiotemporal sequential mining that focuses on discovering frequent tight patterns, along with their constraints of frequency in both space and time. Our



(a) (b)

Fig. 2. Identified solid-blocked sequence $<a, a, g, g>$ for *inline* 401, alphabet size 7, solid range threshold $\gamma = 80\%$ and solid block threshold $\delta = 20\%$. Its density was 206. Solid-blocked sequences are marked in red. The results follow the yellow pattern produced using the previously known *bright spots* for this dataset [5]. (a) Comparison of quality between $GSTSM$ and $STSM$ for sequence $<e, e, d, d>$ in *inline* 401 using alphabet size 7, with support of 80% for $GSTSM$ and with solid range threshold (γ) of 80% and solid block threshold (δ) of 20% for $STSM$. Identified occurrences are marked as red when identified by $STSM$ and as black in $GSTSM$. Although occurrences from $STSM$ correspond to seismic horizons, many occurrences from $GSTSM$ correspond to noise. (b)

formal background provides us with the necessary conditions of such discovery. We proposed $STSM$, a novel algorithm that performs an efficient extraction of these patterns and their spatiotemporal constraints. To the best of our knowledge, this is the first study on this topic. Our experiments, using a real-world seismic dataset, highlighted significant insights according to the feedback of specialists in the domain. $STSM$ allows for extracting patterns that follow areas of major nonconformities between geologic boundary materials of the subsurface, such as *horizons* and *bright spots*. The existence of these discovered patterns was confirmed comparing them with a previously known survey on this dataset, while their meaning was assessed by domain experts. Owing to its novelty and the characteristics of the extracted patterns, $STSM$ is opening new research for tight spatial-time sequences discovery.

Although we have evaluated $STSM$ algorithm using seismic data, it has been conceived to be sufficiently generic for different spatiotemporal datasets (such as IoT domain). The plot of detected solid-blocked sequences according to space and time, as shown in this paper, can aid specialists to analyze identified sequences. Additionally, in scenarios in which a large number of frequent constrained sequences are produced, we can explore specialized ranking functions to prioritize the most interesting ones.

Acknowledges

The authors would like to thank CAPES, CNPq, and FAPERJ for partially funding this paper.

References

1. Alatrista-Salas, H., Bringay, S., Flouvat, F., Selmaoui-Folcher, N., Teisseire, M.: Spatio-sequential patterns mining: Beyond the boundaries. Intelligent Data Analysis 20(2), 293–316 (2016)
2. Aydin, B., Angryk, R.: Spatiotemporal event sequence mining from evolving regions. In: Proceedings - International Conference on Pattern Recognition. pp. 4172–4177 (2017)
3. Batu, B., Temizel, T., Duzgun, H.: A Non-Parametric Algorithm for Discovering Triggering Patterns of Spatio-Temporal Event Types. IEEE Transactions on Knowledge and Data Engineering 29(12), 2629–2642 (2017)
4. Chen, Y.L., Hu, Y.H.: Constraint-based sequential pattern mining: The consideration of recency and compactness. Decision Support Systems 42(2), 1203–1215 (2006)
5. dgbes: Seismic Interpretation Software & Services. Tech. rep., https://dgbes.com/ (2018)
6. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 330–339 (2007)
7. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)
8. Huang, Y., Zhang, L., Zhang, P.: A framework for mining sequential patterns from spatio-temporal event data sets. IEEE Transactions on Knowledge and Data Engineering 20(4), 433–448 (2008)
9. Julea, A., Méger, N., Bolon, P., Rigotti, C., Doin, M.P., Lasserre, C., Trouve, E., Lazarescu, V.: Unsupervised spatiotemporal mining of satellite image time series using grouped frequent sequential patterns. IEEE Transactions on Geoscience and Remote Sensing 49(4), 1417–1430 (2011)
10. Li, K., Fu, Y.: Prediction of human activity by discovering temporal sequence patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(8), 1644–1657 (2014)
11. Li, Y., Bailey, J., Kulik, L., Pei, J.: Mining probabilistic frequent spatio-temporal sequential patterns with gap constraints from uncertain databases. In: Proceedings - IEEE International Conference on Data Mining, ICDM. pp. 448–457 (2013)
12. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03. pp. 2–11 (2003)
13. Mooney, C., Roddick, J.: Sequential pattern mining - Approaches and algorithms. ACM Computing Surveys 45(2) (2013)
14. Sunitha, G., Rama Mohan Reddy, A.: Mining frequent patterns from spatiotemporal data sets: A survey. Journal of Theoretical and Applied Information Technology 68(2), 265–274 (2014)
15. Tsai, C.Y., Shieh, Y.C.: A change detection method for sequential patterns. Decision Support Systems 46(2), 501–511 (2009)
16. Tsoukatos, I., Gunopulos, D.: Efficient mining of spatiotemporal patterns. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2121, 425–442 (2001)
17. Zhou, H.W.: Practical Seismic Data Analysis. Cambridge University Press, New York, 1 edn. (Mar 2014)