CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

SEQUENCE MINING IN SPATIAL-TIME SERIES

Riccardo Campisano

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Produção e Sistemas, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadores:
Eduardo Soares Ogasawara
Florent Masseglia

Rio de Janeiro,
Julho 2017

Sequence Mining in Spatial-Time Series

Dissertação de Mestrado em Engenharia de Produção e Sistemas, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Riccardo Campisano

Aprovada por:

Presidente, Prof. Eduardo Soares Ogasawara, D.Sc. (orientador)

Prof. Florent Masseglia, PhD (INRIA & LIRMM) (co-orientador)

Prof. Esther Pacitti, D.Sc. (INRIA & LIRMM)

Prof. Fabio Andre Machado Porto, D.Sc. (LNCC)

Prof. Diego Moreira de Araujo Carvalho, D.Sc.

Rio de Janeiro,

Julho, 2017

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C197 Campisano, Riccardo
Sequence mining in spatial-time series / Riccardo Campisano.— 2017.
52f. + apêndice : il. (algumas color.) , grafs. ; enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2017.
Bibliografia : f. 48-52
Orientadores : Eduardo Soares Ogasawara
Florent Masseglia

1. Engenharia de produção. 2. Engenharia de sistemas. 3. Mineração de dados (Computação). 4. Análise de séries temporais. I. Ogasawara, Eduardo Soares (Orient.). II. Masseglia, Florent (Orient.). III. Título.

CDD 519.232

A mia moglie, la persona che ha dato inizio, incoraggiato e pagato assieme a me il prezzo di questa odissea, e con la quale voglio continuare a dividere gioie e dolori di molti altri progetti.

RESUMO

Sequence Mining in Spatial-Time Series

Riccardo Campisano

Orientadores:
Eduardo Soares Ogasawara
Florent Masseglia

Resumo da Dissertação submetida ao Programa de Pós-graduação em Engenharia de Produção e Sistemas do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do título de mestre.

O problema da descoberta de padrões sequenciais em grandes conjuntos de dados envolve uma ampla gama de aplicações científicas e industriais. Em um número crescente de aplicações, os dados são coletados como sequências espaço-temporais onde cada item na sequência é associado a um tempo e a uma posição espacial. Isso leva a um novo e fascinante desafio para este domínio: encontrar, com o mesmo processo: i) sequências frequentes restritas no espaço e no tempo que podem não ser frequentes em todo o conjunto de dados e ii) o intervalo de tempo e espaço onde essas sequências são frequentes. A descoberta de tais padrões junto com suas restrições pode levar a extrair conhecimento importante que pode permanecer escondido usando os métodos tradicionais, pois seu suporte é extremamente baixo no inteiro conjunto de dados. Apresentamos o novo algoritmo *Spatio-Temporal Sequence Miner* ($STSM$) para esse fim. O $STSM$ é baseado em nosso novo princípio de mineração de padrões sequenciais a fim de detectar intervalos espaciais onde as sequências são frequentes. Em seguida, ele compõe todas as sequências detectadas dentro de cada intervalo para descobrir aqueles restritos no espaço e no tempo em que essas sequências são frequentes. Apesar de nossa solução ser genérica, avaliamos o $STSM$ em um caso de uso sísmico e ilustramos sua capacidade de detectar sequências frequentes restritas no espaço e no tempo. Conseguimos identificar 1.500 sequências restritas com alto limite de suporte, as quais não teriam sido encontradas usando as técnicas atuais. Além disso, as sequências identificadas pelo $STSM$ correspondem na sismica a áreas candidatas a *horizons* e a *bright-spots* que são de grande interesse para os especialistas nesse domínio. Tanto quanto é do nosso conhecimento, esta é a primeira solução para enfrentar o problema da identificação de sequências frequentes restritas no espaço e no tempo.

Palavras-chave:
Spatial-Time Series; Sequential Patterns; Data Mining.

Rio de Janeiro,

Julho, 2017

ABSTRACT

Sequence Mining in Spatial-Time Series

Riccardo Campisano

Advisors:
Eduardo Soares Ogasawara
Florent Masseglia

Abstract of dissertation submitted to Programa de Pós-graduação em Engenharia de Produção e Sistemas - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

The problem of discovering sequential patterns in large datasets affects a wide range of scientific and industrial applications. In a growing number of applications, data are collected as spatio-temporal sequences that associate to each item in the sequence a time and a spatial position. This leads to an appealing new challenge for this domain: find, with 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 important 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 for such purpose. $STSM$ is based on our novel sequential pattern mining principle in order to detect spatial ranges where sequences are frequent. Next, it composes all detected sequences inside each range to discover the ones constrained in space and time where these sequences are frequent. Even though our solution is generic, we evaluate $STSM$ on a seismic use case and illustrate its ability to detect frequent sequences constrained in space and time. We managed to identify 1,500 constrained sequences under high support threshold, which would not have been found using current techniques. Moreover, the identified sequences from $STSM$ correspond to candidate areas for seismic *horizons* and *bright spots* that are of high value for domain experts. To the best of our knowledge, this is the first solution to tackle the problem of identifying frequent sequences constrained in space and time.

Key-words:
Spatial-Time Series; Sequential Patterns; Data Mining.

Rio de Janeiro,

July, 2017

List of Figures

2.1 Agrawal and Srikant 1994 algorithm behavior. 18
2.2 Example of time series representing exchange rates. 21
2.3 Example of seismic spatial-time series data. 22
2.4 Time series conversion to PAA representation. 23
2.5 Time series conversion to SAX representation. 23

4.1 Seismic dataset survey location. 35
4.2 Onshore seismic survey representation. 36
4.3 Visual representation of used seismic dataset. 37
4.4 Representation of the data discretization. 38
4.5 SAX alphabet size results comparison for *inline* 401. 39
4.6 Threshold parameters comparison: *inline* 401 - SAX 10 - sequence $<a, a, a>$. . . . 41
4.7 Ranked identified sequences: *inline* 401 - SAX 10 - γ 80% - δ 20%. 42
4.8 Potential *bright spots* identification: *inline* 401 - SAX 10 - γ 80% - δ 20%. 43

6.1 Inlines versus SAXs result comparison for sequence $<a, a, a>$. 54
6.2 Inlines versus SAXs result comparison for sequences of size two. 55
6.3 Inlines versus SAXs result comparison for sequences of size four. 56

List of Tables

2.1 Example of database used in Agrawal and Srikant 1995a. 19
2.2 Example of Agrawal and Srikant 1995a converted *customer-sequence* dataset. . . . 19

3.1 Frequent sequences on D where $\gamma = \frac{1}{2}$. 26

6.1 SAX alphabet symbol ranges by size. 53

List of acronyms

CSV *Comma Separated Value* 37, 38

GSP *Generalized Sequential Pattern* 19, 45, 46

PAA *Piecewise Aggregate Approximation* 22, 23

SAX *Symbolic Aggregate approXimation* 23, 37, 38, 39, 40

$STSM$ *Spatio-Temporal Sequence Miner* 13, 14, 24, 29, 30, 35, 37, 40, 42, 43, 46, 47

TSS *TimeStamped Sequence* 24, 25, 26, 27, 29, 33, 37, 42, 47

Contents

1 Introduction **12**

2 Background **15**

2.1 Frequent Patterns 15

2.1.1 Association Rules 16

2.1.2 Apriori Algorithm 16

2.1.3 Sequential pattern mining 17

2.1.4 Tight patterns 20

2.2 Time Series and Spatial-Time Series 20

2.2.1 General Concepts 20

2.2.2 Indexing 21

3 Proposal **24**

3.1 Definitions 24

3.1.1 Basic Concepts 24

3.1.2 Problem Statement 26

3.1.3 Spatial-TSS Definitions 26

3.2 General Principle and Algorithm 29

3.2.1 General Principle 29

3.2.2 Generation of Candidates 30

3.2.3 Selection of Solid Ranged Sequences 30

3.2.4 Selection of Solid Blocked Sequences 31

3.2.5 Merging Kernels 32

4 Experimental Evaluation **35**

4.1 Input Dataset 35

4.2 Discretization setup 37

4.3 STSM Parameters Setup 39

4.4 Discussions 40

5 Related works **44**

6 Conclusion **46**

References **48**

Appendix **53**

Chapter 1 Introduction

Frequent patterns mining algorithms are widely used to obtain new exploratory knowledge from data. Commonly, their results can be used as input for other mining techniques like clustering and classification. Earlier frequent patterns techniques focus on finding association rules on transactional databases, where each transaction is assumed to be independent from the others, to find frequent pattern over the time [Agrawal et al., 1993a]. Later, this concept evolved to sequential pattern mining [Agrawal and Srikant, 1995a] taking advantage of the fact that some successive transactions belong to the same customer. Such concept was further generalized to handle databases composed of generic type of data-sequences [Srikant and Agrawal, 1996a].

Meanwhile, many important event records of different application areas are characterized as time series. In such model, data is composed of chronological ordered observations of continuous (non discrete) values. The search for frequent patterns in such data is a difficult task due the fact that comparison between continuous values can not be accomplished using an exact match like it is realized in traditional databases of discrete values [Fu, 2011]. Other difficulties for time series analysis are the unceasing increment of observations over the time, as well as the space needed to store the large volume of generated data and the time consumed to analyze it [Fu, 2011]. Financial trends, medical and climatic data, industry processes, remote sensors, statistical information along with others are examples of areas that generate large volume of time series data.

Moreover, some important phenomena are indeed spatial-time series. This leads to an additional challenge not just because the volume of data is amplified by the introduction of the spatial dimension, but mainly due the complexity of the analysis of spatial-time relationship associated to the collected observations [Han et al., 2007]. Examples of spatial-time datasets are the ones obtained from seismic surveys. A set of receivers, placed in specific fixed positions of the surface (onshore or offshore), captures the waves generated producing artificial shots in the ground. Such waves propagate in the subsurface and are reflected and refracted by the different materials encountered. In this way, each receiver composes a time series of observations, where each observation is related to a specific travel time and all observations in the same time series are related to the spatial position of the receiver. Exploring relationship between space and time constraint can be important to give new clues about the studied phenomena.

In this work, we investigate a new problem related to spatio-temporal pattern identification. We are interested in finding sequences that are frequent on a constrained space and time (space-time block), and may not be frequent in the entire dataset. The main challenge is to discover these blocks and the frequent sequences they contain. Solving this problem has a valuable impact for 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, by means of existing methods, because they concern a niche (*i.e.*, they have an extremely low support, that hinders their discovery). Such a niche could be, for instance, that "people in Manhattan who jog at 7am 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 meanwhile, the accumulation of such niches, particularly in massive data, makes their discovery a difficult and valuable target for data analytics, where the challenge is to extract both the pattern (*e.g.*, jog, lunch, buildings displays), its occurrence time (*e.g.*, from 7am to lunch time), and the location where it occurs (*e.g.*, Time Square). This also concerns use-cases related to monitoring or observation data. Consider the seismic use-case of this work (detailed in Chapter 4). We are provided with geolocalized sequences that represent the texture of several different lithologies in subsurface to a given depth. Here, the space dimension may be mapped to the physical location of the sequence (it is geolocalized), and each depth interval may be mapped to a virtual time range. In this domain, finding a frequent pattern of textures in the entire set is of little interest (it would lead to spread patterns and would not help focusing on a specific area). However, some patterns occurring in a constrained depth interval and a specific space location may lead to highly valuable lithology change indicator such as *horizons* (which stand for kind of layers) or *bright spots* (potential areas of hydrocarbon accumulation) [Zhou, 2014a].

This work introduces the problem of spatio-temporal sequence identification in constrained blocks of space and time 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.

We evaluated the $STSM$ algorithm in a seismic use case based on an open seismic survey [OpendTect website]. By applying constrained space-time sequence mining techniques, we managed to identify 1,500 constrained sequences under high support threshold, which would not have

been found using current techniques. Moreover, the algorithm was able to detect candidate areas for seismic *horizons* and *bright spots*, known as important characteristics for the domain experts.

This work is organized as follows. Chapter 2 provides the theoretical basis about frequent pattern mining, from Association Rules to Sequential Pattern mining, as well as about time series, spatial-time series and their indexation techniques. Chapter 3 presents the problem definition and the $STSM$ algorithm, describing each part of the algorithm and details about the implementation. Chapter 4 presents the proposed experimental evaluation using a real-world seismic use-case, describes the used input spatial-time dataset, the parameters used in the discretization and in the fine tuning of the $STSM$ algorithm, concluding the Chapter presenting the experiment results their discussions. Chapter 5 presents related works in this area of application, evidencing the differences with the solution proposed in this work. Finally, Chapter 6 presents the conclusions with some considerations and proposed future directions. Furthermore, additional results using different algorithm's configurations are presented in Appendix.

Chapter 2 Background

Sequential pattern mining is a technique to discover frequent patterns in a dataset of sequences [Mabroukeh and Ezeife, 2010]. In this context, a sequence is a list of transactions ordered by time. A transaction, in turn, is a set of some kind of items, without a defined order between them. For example, a department store keeps track of the purchases of each customer. This can generate a dataset containing a time-ordered list of customer purchases, each of them is a set of products that a single customer bought in a single purchase [Mabroukeh and Ezeife, 2010].

One can be interested to know that there is a pattern of customer that first buy a specific product and then another one (among others products that can be bought before, during or after that purchase). Thus, finds frequent patterns means to find frequent repetitions of time-ordered set of itemsets, where itemset is a non-empty and unordered set of items (*e.g.* products bought in a single purchase). It can be thought as using association rules discovery in a temporal dataset: while association rules are used to find patterns without the use of transactions, sequential pattern mining discover sequential patterns, *i.e.*, patterns between transactions, where the order of sequences of itemsets is very important [Mabroukeh and Ezeife, 2010].

This chapter is organized as follows. Section 2.1 provides the fundamentals of Frequent Pattern mining, including Association Rules, the *Apriori Principle* and Sequential Patterns Mining. Then, Section 2.2 provides the general concepts about time series, spatial-time series and the indexing techniques useful to reduce the dimensionality and to discretize time series data to improve the efficiency of further analysis.

2.1 Frequent Patterns

Frequent patterns are defined as “itemsets, subsequences, or substructures that appear in a dataset with frequency no less than a user-specified threshold” [Han et al., 2007]. Example of frequent itemset is “a set of items, such as milk and bread, that appear frequently together in a transaction dataset”. Example of frequent sequential pattern is a frequent subsequence “such as buying first a PC, then a digital camera, and then a memory card”. Example of frequent structural pattern is a substructure that appears frequently in a graph database and “can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with item-

sets or subsequences" [Han et al., 2007].

2.1.1 Association Rules

Agrawal et al. first introduced the discover of frequent pattern and association rules in transactional databases. Their algorithm can generate all significant association rules between the items in a database [Agrawal et al., 1993a]. Let $I = \{i_1, i_2, ..., i_n\}$ be the set of all the items that, for example, a department store can sell. An association rule take the form $A \Rightarrow b$, where A is a set of items $\mid A \subset I$, b is a single item $\mid b \in I$ and $b \notin A$. A significant association rule is defined by two kind of constraints: (i) syntactic and (ii) support. A syntactic constraint is a constraint that may define that an item must be in the antecedent or in the consequent part of the association rule. A support constraint may define the minimum number of transactions that support an association rule. In their work, the dataset T is composed of a set of transactions $\{t_1, t_2, ..., t_n\}$ and each transaction is a binary vector of same size of I, where $t[k] = 1$ if t "bought" the item i_k, otherwise $t[k] = 0$.

2.1.2 Apriori Algorithm

Considering that there are a lot of distinct items in a typical database. The evaluation of the support of their combinations means the generation of a very large number of combinations in a way that is very difficult to evaluate the support of each combination in a efficient way. The mechanics of efficiently computing association rules was firstly described by Agrawal and Srikant [1994], based on the *Apriori Principle* which is described as follows: given a frequent itemset of k items (called *large k-itemset*), any subset of that itemset is also frequent. So, an efficient way to skip generation of not frequent candidate combinations is to consider only the $(k-1)$-itemsets that are frequents and use only their combinations to generate the k-itemsets candidates that are filtered to keep only the ones that match the minimum support count [Han et al., 2007].

The *Apriori* algorithm is presented in Algorithm 1 and it is composed of two activities called *join* step and *prune* step. The algorithm starts with previously detected frequent itemset as inputs at line 1, treated as 1-itemsets. Then, in the *join* step, the candidate generation function $apriori{-}gen$ at line 3 generates all combinations of k-itemsets between all $(k-1)$-itemsets that contain identical items except the last one. Next, in the *prune* step, the support count of each frequent itemsets candidates is calculated (testing them against the transactions of the original database) between the lines 4 and 9, and the candidates are filtered at line 10 keeping only the ones that reach the minimum support value. The algorithm iterates between the *join* and the *prune* steps until no new frequent itemsets can be found.

Assuming that the data is stored in a relational database, it is possible to express the *Apriori* algorithm *join* step using a declarative query as depicted in Algorithm 2 [Agrawal and Srikant,

Algorithm 1 Agrawal and Srikant 1994 algorithm pseudo-code.

```
1: L_1 ← {large 1−itemsets};
2: for (k = 2; L_{k−1} ≠ ∅; k++) do
3:     C_k = apriori−gen(L_{k−1}); // New candidates
4:     for all transaction t ∈ D do
5:         C_t = subset(C_k, t); // Cantidates contained
6:         for all candidates c ∈ C_t do
7:             c.count++;
8:         end for
9:     end for
10:    L_k = {c ∈ C_k | c.count ⩾ minsup}
11: end for
12: Answer = ⋃_k L_k;
```

1994]. For example, the *join* of $\{abc\}\{abd\}\{abe\}\{acd\}$ produces $\{abcd\}$ (joining $\{abc\}$ with $\{abd\}$), $\{abce\}$ (joining $\{abc\}$ with $\{abe\}$) and $\{abde\}$ (joining $\{abd\}$ with $\{abe\}$). Note that $\{acd\}$ can not be joined with other itemset because it is the only itemset that starts with ac characters and that itemsets like $\{abdc\}$ are not generated since items ordering does not matter. The complete *Apriori* algorithm behavior is elucidated using an example in Figure 2.1.

Algorithm 2 Agrawal and Srikant 1994 *join* step algorithm takes L_{k-1} as argument and returns the set of all $(k-1)$-itemsets.

```
insert into Ck
    select
        p.item_1, p.item_2, ..., p.item_{k−1}, q.item_{k−1}
    from
        L_{k−1} p, L_{k−1} q
    where
        p.item_1 = q.item_1, ..., p.item_{k−2} = q.item_{k−2},
        p.item_{k−1} < q.item_{k−1}
```

2.1.3 Sequential pattern mining

Frequent pattern mining techniques was subsequently expanded to support sequential pattern mining. While in their previous work, the support is defined as the number of transactions that contains the itemset, in Agrawal and Srikant [1995a] the support is redefined as the fraction of customers that bought the items of the itemset in a single transaction of the customer. In their work, the algorithm is not only interested in frequent itemset in all transactions, but the aim became to find frequent sequence of items between customer purchase data. An example of sequence pattern to discover is customers that typically rent “Star Wars: A New Hope”, then “Empire Strikes Back”, and then “Return of the Jedi” [Agrawal and Srikant, 1995a].

A transactional database usually have records that associate customer identification with date-

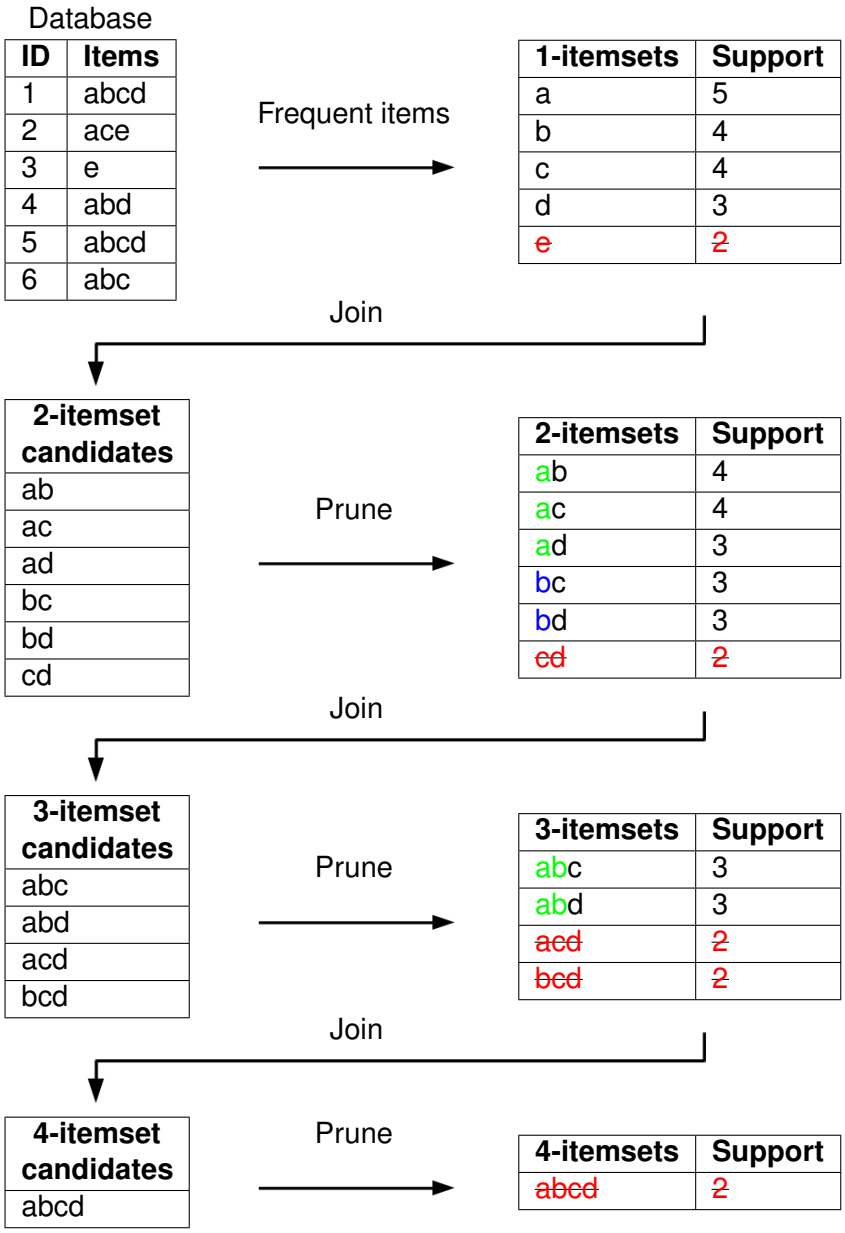


Figure 2.1: Agrawal and Srikant 1994 algorithm behavior. First the frequent items are detected, then there is a repetition of *join* and *prune* steps, alternatively. There, the k-itemsets candidates are generated from previously $(k-1)$-itemsets and then the candidates with a support count lower than the minimum support are discarded.

time. All the transaction of the same customer can be considered as a sequence, where each sequence is composed of a list of itemsets and each itemset is composed of a set of unique items that was bought in the same transaction. So, the first step of the algorithm is the transformation of the original database in a list of *customer-sequences*. An example of the transformation of such database is displayed in Tables 2.1 and 2.2 that show the original database sorted by the customer_id and the converted *customer-sequence* dataset, respectively. The subsequent steps of the algorithm behavior remains similar to Agrawal and Srikant [1994].

The problem of discovery frequent sequential pattern is defined in Agrawal and Srikant [1995a]

Table 2.1: Example of database used in Agrawal and Srikant 1995a, sorted by customer_id.

Customer Id	Transaction Id	Items
1	1	c
1	2	i
2	1	a,b
2	2	c
2	3	d,f,g
3	1	c,e,g
4	1	c
4	2	d,g
4	3	i
5	1	i

Table 2.2: Example of converted *customer-sequence* dataset used in Agrawal and Srikant 1995a.

Customer Id	Customer Sequence
1	$<$ (c)(i) $>$
2	$<$ (a b)(c)(d f g) $>$
3	$<$ (c e g) $>$
4	$<$ (c)(d g)(i) $>$
5	$<$ (i) $>$

as finding the maximal sequences among all sequences that have a certain user-specified minimum support. For each frequent sequence discovered, all subsequences of such sequence are also frequent. In the case of a large frequent sequence, there is an exponential number of smaller frequent sub-sequences. Several works use the concept of *closed sequence pattern* and *maximal sequence pattern* to reach the same results quality generating a minor set of results. A pattern B is a *closed sequence pattern* only if it has not a super pattern $A \mid B \subset A$ where A has the same support as B. A pattern B is a *maximal sequence pattern* only if it has not a super pattern $A \mid B \subset A$ where A is frequent [Han et al., 2007].

Generalized Sequential Pattern (GSP) algorithm [Srikant and Agrawal, 1996a] is a generalized version of the work of Agrawal and Srikant [1995a] with the aim to handle user-defined taxonomies. This work introduces optimization features such as *min-gap* and *max-gap* time constraints to restrict the time gap between sets of transactions that contains consecutive elements of the sequence. Additionally, a *sliding time window* parameter was added to relax the constraint in which all items in an element of a sequential pattern must come from the same transaction [Srikant and Agrawal, 1996a]. This allows for all contiguous transactions between *sliding time window* interval to be considered as a unique transaction. Moreover, the concept of *customer-sequence* was generalized to handle every dataset defined as a list of *data-sequence*, and several performances improvements were evaluated [Srikant and Agrawal, 1996a].

2.1.4 Tight patterns

However, there are interesting patterns that are not frequent in the entire dataset being analyzed. Paschal egg is not a frequent purchased item of a hypermarket chain. However, we can imagine that there is a frequent purchase pattern of such product in a specific date range just before Easter. The discover of others not so obvious patterns may lead to important gain of knowledge for industry or scientific fields of research.

Saleh and Masseglia [2008] proposed an algorithm to detect tight patterns in transactional databases, i.e. patterns that are frequent in a specific periods of subsequent transactions. To achieve this result in a efficient way, their algorithm introduces the notion of *kernel*. Let D be the analyzed dataset and f the user defined frequency threshold, a *kernel* k of an item i is defined as the period that identify a subset of subsequent transactions of D where the first and the last transactions of the period support i and the frequency of i in such period is greater then or equal to the threshold f. The set of all existing *kernels* in the dataset can be easily obtained performing a single and continuous scan over the dataset, obtaining a first set of shorter *kernels*. To obtain the largest periods of transactions where an item is frequent without execute additional dataset scans, the function *mergeKernels* was introduced. It produces all possible couples combining the set of *kernels* obtained from the previous and unique scan of the dataset. For each couple, if the period of the two *kernels* can be joined returning a set of subsequent transactions where the item is still frequent, then the *mergeKernels* function produces a new *kernel* from the join of the two *kernels* periods, add it in the set of resulting *kernels* and removes the two original *kernels* from such set. The function continues looping the merge process until no further merges can be performed, obtaining the largest periods in the dataset where each item is frequent. The same logic can be used to discover the largest time intervals and spatio-temporal blocks where a sequence is frequent, as is presented in Chapter 3.

2.2 Time Series and Spatial-Time Series

2.2.1 General Concepts

Differently from transactional datasets, that contains repetitions of a limited number of possible discrete items, time series datasets contains collections of observations of numerical continuous (non discrete) values in a chronological order [Fu, 2011]. A time series of such floating point values T can be defined as a series of observation $< t_1, t_2, ..., t_n >$, where t_1 is the value of the first observation, t_n is the value of the last observation, $|T| = n$ is the number of observations in the time series and $\forall\ t_i \in\ T, t_i \in \mathbb{R}$. An example of time series data in financial area is shown in Figure 2.2 where in the vertical axis are displayed the Euro-Dollar exchange rate values and in

the horizontal axis is displayed the time related to each observation, in a period between the years 2011 and 2016.



Figure 2.2: Example of time series representing Euro vs Dollar exchange rates between the years 2011 and 2016. The image was obtained from the European Central Bank website.

Some of the difficulties in time series analysis are the unceasing increment of observations over the time, the space needed to store the large volume of data and the time consumed to analyze it [Fu, 2011]. This is especially true for spatial-time series where the mining of frequent patterns is more expansive due the complexity of spatial-time data objects and their relationships [Han et al., 2007]. This data can be considered as space-related time series data or, in turn, as time-related spatial data. A spatial-time series S can be defined as a structure composed of its spatial coordinates, *e.g.* $S.x$ and $S.y$, and a time series of values $S.t =< t_1, t_2, ..., t_n >$. An example of such spatial-time series in seismic area is shown in Figure 2.3. It represents a vertical cut of the subsurface reproducing the waveform and arrival times of waves traveling in the existing materials of the ground. The horizontal axis represents a spatial direction (*e.g.* north-south), the vertical axis represents the travel time (that is related to the depth) and the gray scale represents the waveform. Each spatial position in the horizontal axis is related to the sensor that received the wave signals and produced the time serie for such position.

2.2.2 Indexing

The search of frequent patterns in traditional databases, where the data is composed of a limited number of distinct discrete items, differs substantially from the search of frequent patterns in time series data, which is composed of series of continuous (non discrete) values. In the former case the search is based on exact match between items, while in the latter case the search is mostly

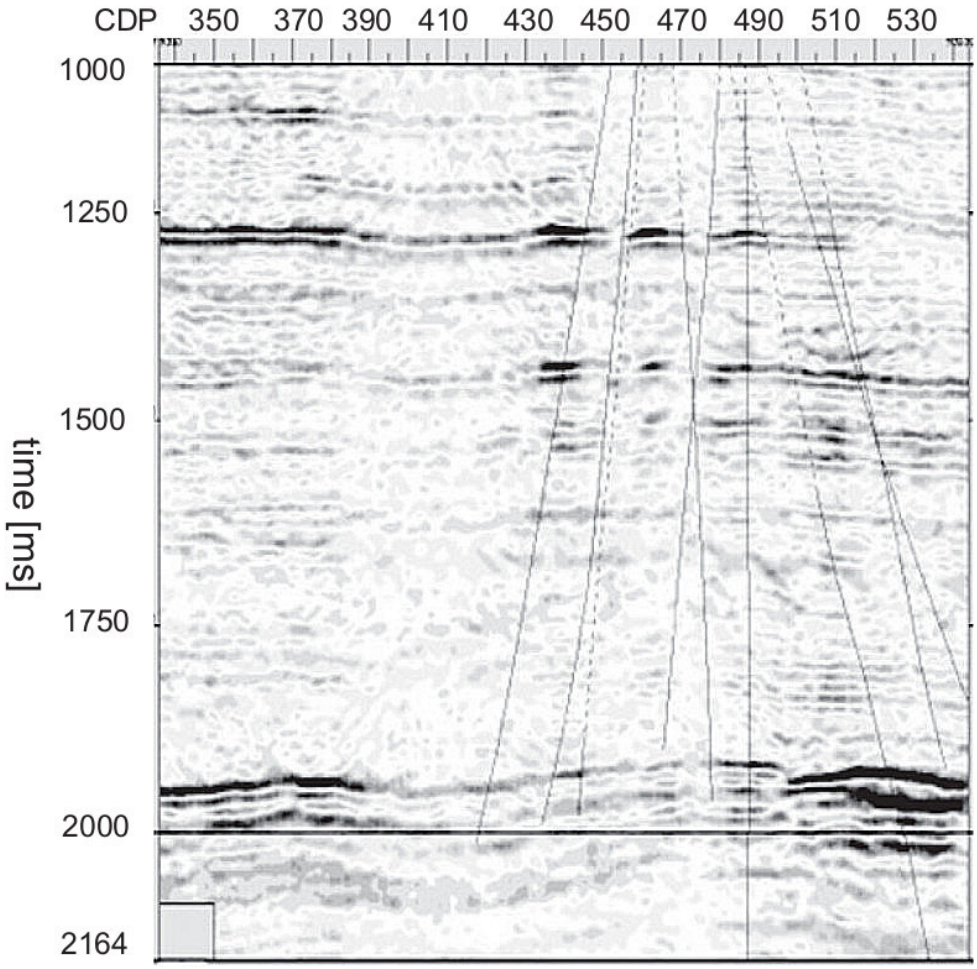


Figure 2.3: Example of seismic spatial-time series data reproducing wave arrival times and waveforms of the materials existing in a vertical cut of the subsurface [Zhou, 2014b].

achieved through approximations due to the difficulty to find exact match among continuous values [Fu, 2011]. To diminish this complexity, indexing techniques can be used to discretize time series dataset as well as to reduce their dimensionality [Lin et al., 2003a].

A simple way to reduce dimensionality is sampling. Let o be the number of observations in a time series t and let r be the number of dimension requested. Defining a factor $f = o/r$, one can consider only one observation every f of the original time series values to perform the dimensionality reduction. On the other hand, this method heavily distorts the original time series shape. The *Piecewise Aggregate Approximation* (PAA) representation [Kang and Yong, 2010] performs a dimensionality reduction and obtains a more representative value of the original time series. PAA divides the dataset in segments of equal size, so the value of the mean of the data that each segment contains can be used to represent the segment. Considering the segment k of a time series $t =< t_1, ..., t_i, ..., t_n > \mid k \subset t$. Let k_1 and k_n be the position number of the start and the end observations in the sequence, respectively. The value $\hat{t_k}$ representing the dimensional reduced segment k of t can be calculated as follows [Fu, 2011], and an example of result can be

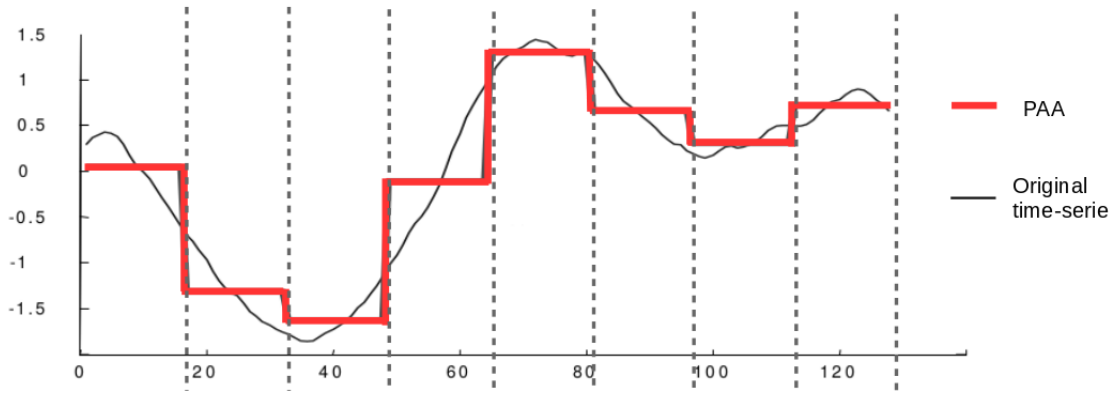


Figure 2.4: Time series conversion to PAA representation. Figure adapted from Lin et al. 2003a.

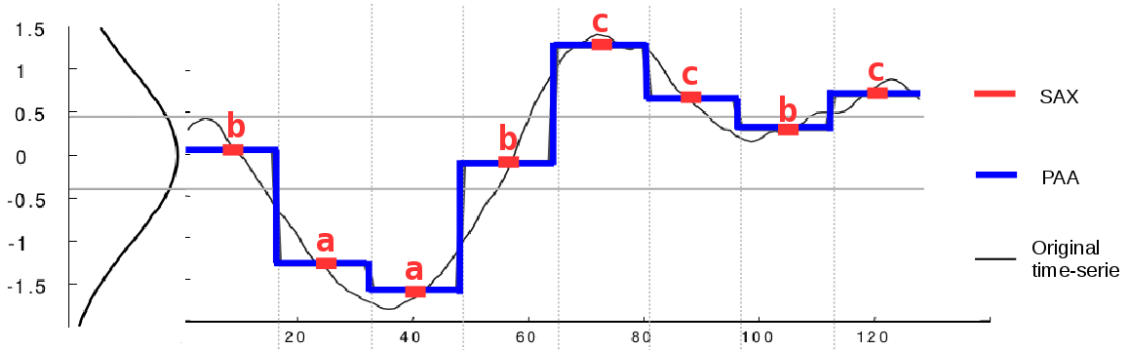


Figure 2.5: Time series conversion to PAA representation (in blue) and then to SAX representation (in red). Figure adapted from Lin et al. 2003a.

shown in Figure 2.4:

$$\hat{t_k} = \frac{1}{k_n - k_1 + 1} \sum_{i=k_1}^{k_n} t_i \tag{2.1}$$

An enhanced way to represent time series in a reduced dimensionality form as well as in a discrete format is *Symbolic Aggregate approXimation* (SAX) [Lin et al., 2003a]. This algorithm normalizes the time series and uses the PAA representation to perform a dimensionality reduction. In addiction, SAX performs a discretization of PAA mean values, converting them to symbols with equiprobable distribution. This is possible considering the Gaussian behavior of normalized time series, in a way that is possible to identify breaking-points that produces equal-size areas under Gaussian curve simply looking in a statistical table [Lin et al., 2003a]. Figure 2.5 depicts a time series conversion to obtain the PAA representation (in blue) and then to obtain the SAX representation (in red) using the Gaussian behavior of time series illustrated from the curve at left of the image.

Chapter 3 Proposal

This chapter is organized in two more sections. Section 3.1 formalizes the problem and presents the definitions needed to describe our algorithm. Section 3.2 presents the $STSM$ algorithm proposed to identify sequential patterns in spatial-time datasets.

3.1 Definitions

The problem of identifying patterns in spatio-temporal datasets is rather new. However, the problem of finding sequential patterns in transaction databases [Agrawal and Srikant, 1995b] can bring light to address the former problem. Indeed, the later problem of association rule mining is based on the extraction of frequent itemsets [Agrawal et al., 1993b] where many algorithms have been proposed so far [Toivonen, 1996; Bayardo, 1998; Pasquier et al., 1999; Han et al., 2000b; Wang et al., 2003; Burdick et al., 2005; Lucchese et al., 2006; Palshikar et al., 2007; Lian et al., 2007].

3.1.1 Basic Concepts

Definition 1 states the characteristics of frequent sequences in spatio-temporal datasets. It is different from the initial or traditional definitions since we now consider that each item in a dataset is associated with a position and a time. In this work, we consider a bi-dimensional space, which is a first step towards n-dimensional spaces. We first give a set of definitions and we illustrate them on a toy example that embeds all the concepts.

Definition 1*:* Let $t =< v_1, v_2, \cdots, v_n >$ be a ***TimeStamped Sequence*** **(**TSS**)**, *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 $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ be a set of positions, over which a linear order $<_P$ is defined, where $p_i <_P p_j$ means that p_i is positioned before p_j (i.e. space is constrained to a single spatial dimension). A **Spatial-**TSS d is a couple (p, t) where $p \in P$ is a position and t is the associated TSS. A **spatial-**TSS **dataset** D is a set of spatial-TSS. For the sake of simplicity, we consider that each position of $d \in D$ is unique.

A spatial-TSS $d = (p, t)$ 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 spatial-TSS in D in which s is included. The **frequency** of a sequence s in D is the fraction of spatial-TSS 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 $freq(s, D) \geq \gamma$.

Definition 2*:* Let $\mathcal{I}$ be the set of all possible sequences s included in D, **the set F of frequent sequences** in D with respect to γ is denoted by $F(D, \gamma) = \{s \in \mathcal{I} : freq(s, D) \geq \gamma\}$.

Definition 3*:* A **spatial range** (or simply **range**) $r = (p_s, p_e)$ is defined by a starting position p_s and an end position p_e. We define the set of all potential ranges over D as PR. The set of spatial-TSS 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 spatial-TSS 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 $sup(s, r)$.

Definition 4*:* 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.

Definition 5*:* 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 $i_s \geq 1 \wedge i_e \leq n$, $i_e - i_s + 1 = k$ 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.

Definition 6*:* 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 $PB(r)$.

Definition 7*:* A **minimum block** b_v for a sequence s is a block in which s is included and there is no other block b_l, $b_l \subset b_v$, such that s is included in b_l. Moreover, a **bounding box** between two blocks b_i, b_j: $bound(b_i, b_j)$ be the minimum block b that contains them.

Definition 8*:* The **occurrences** of a *sequence* s in a block b, $occur(s, b)$ is a set of non-overlapping minimum blocks b_j for s, with $|b_j.r| = 1$ and $b_j \subseteq b$. The **support** of a *sequence* s in a block b, $sup(s, b)$ is the number of occurrences of s in b: $|occurrences(s, b)|$.

Definition 9*:* The **item-support** of a *sequence* s in a block b, $isup(s, b)$, is the number of items in b that corresponds to occurrences of 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 $ifreq(s, b) \geq \delta$.

Definition 10*:* 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: $ifr = ifreq(s, b)$.

Example 1*:* Table 3.1 shows an example of spatio-temporal dataset D. Each position p_i is associated with a spatial-TSS d_i in dataset D. In order to simplify the illustration, we assume that spatial-TSS of D are sorted according to their position (*i.e.* d_1 is located before d_2, etc.). Let us consider a minimum frequency $\gamma = \frac{1}{2}$ given by the user. In this context, the frequent sequences of D are $<a>$, $<b>$, $<c>$, with a frequency of $\frac{6}{10}$, and $<a, c>$, with a frequency of $\frac{1}{2}$. Yet, consider the ranged sequence $sr = (< a, b, c >, [7..10], \frac{3}{4})$ in D from Table 3.1. The sequence of sr (*i.e.* s) is $<a, b, c>$. The range of sr (*i.e.* r) is $[7..10]$ and the frequency of s over r (*i.e.* fr) is $\frac{3}{4}$ (s is included in spatial-TSS d_7, d_9 and d_{10} in range r on D). Finally, consider a block b $([1..3], [1..5])$. Let s be a sequence $<e, x>$. The item-support and item-frequency for s in b are, respectively, 5 and $\frac{5}{15}$. Thus, we obtain a blocked sequence $bs(< e, x >, ([1..3], [1..5]), \frac{5}{15})$.

Table 3.1: Frequent sequences on D where $\gamma = \frac{1}{2}$.

D	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
v_1	**a**	v	e	h	**a**	**b**	p	x	w	z
v_2	**b**	w	e	y	i	k	s	m	v	t
v_3	e	v	x	h	q	l	s	x	**a**	z
v_4	**c**	d	o	**c**	t	r	**a**	n	w	**a**
v_5	x	**a**	**b**	j	z	k	**b**	x	**b**	**b**
v_6	z	**c**	o	y	t	r	**c**	o	**c**	**c**

$F(D, \frac{1}{2})$ = {$<a>$, $<b>$, $<c>$, and $<a, c>$}

3.1.2 Problem Statement

Considering a spatial-TSS dataset D, the problem we address is to find sequences that are frequent on a spatial range in D and constrained into a particular 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 the core of this work. Here again, these concepts will be illustrated on the same toy example.

3.1.3 Spatial-TSS Definitions

Given γ, a user's minimum threshold, we introduce the characteristics of solid ranged sequence in Definition 11.

Definition 11*:* Let sr be a ranged sequence with 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) $\forall r_2 \in PR$ such that $r \subseteq r_2$ 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 $r_2 \subset r$, $sup(s, r_2) < sup(s, r)$

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

The first condition of Definition 11 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, on 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 will not 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 spatial-TSS in r, so if a smaller range exists where s is frequent, the support will be lower anyway (*i.e.* relevant spatial-TSS supporting s would have been dropped from the range and should be kept). An illustration of this definition is given in Example 2.

Example 2*:* We can observe from Table 3.1 that the solid ranged sequences of size 1 are $<a>$, $<b>$, $<c>$, $<e>$, $<x>$. For $<a>$, $<b>$, $<c>$ the range for corresponds to the entire dataset with a frequency of $\frac{6}{10}$ and for $<e>$, $<x>$ the range is $[1..3]$ with a frequency of $\frac{2}{3}$. Then, we have four solid ranged sequences of size 2: $<a, c>$, with a frequency of $\frac{5}{10}$ and a range that corresponds to the entire dataset; $<e, x>$ on the range $[1..3]$ with a frequency of $\frac{2}{3}$; $<a, b>$ and $<b, c>$, on the range $[7..10]$ with a frequency of $\frac{3}{4}$. Finally, there is one solid ranged sequence with size 3: $sr = (<a, b, c>, [7..10], \frac{3}{4})$ corresponding to the sequence $<a, b, c>$ which occurs during the range $[7..10]$ with a frequency of $\frac{3}{4}$. It is worth-mentioning that $<a, b, c>$ in not frequent in the entire dataset D. Thus, these punctual behaviors of a phenomenon constrained in space arise from the definition of solid ranged sequences.

The sequence $<a, b, c>$ associated with the range $[7..10]$ in sr is optimal (as stated in Definition 11) since:

- This sequence is frequent over this range.
- No longer range allows this sequence to have the minimum threshold (condition 2 in Definition 11 is respected for all ranges larger than $[7..10]$).
- No shorter range allows this sequence to have the minimum threshold without decreasing the support. Therefore, sr respects the third condition of Definition 11.

Given γ and δ, user's minimum thresholds, respectively, for frequency and item-frequency, we introduce the characteristics of solid blocked sequence in Definition 12.

Definition 12*:* 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) $\exists\ sr$ with range r such that $b.r \subseteq r$ and sr is a **solid blocked sequence**.

2) $ifr \geq \delta$

3) $\forall b_2\ (b_2.i \in PI, b_2.r \in PR)$ such that $b \subseteq b_2$, we have either a) or b) or both:

   a) $ifreq(s, b_2) < \delta$

   b) $isup(s, b_2) = isup(s, b)$

4) $\forall b_2\ (b_2.i \in PI, b_2.r \in PR)$ such that $b_2 \subset b$, $isup(s, b_2) < isup(s, b)$

5) $sup(s, b) > 1$

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

The first condition ensures that sb is inside a solid blocked sequence sr. In other words, candidate blocks for solid blocked sequence are identified from computed solid blocked sequence. 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 will not 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 will be lower anyway. Finally, the fifth condition avoids trivial solid blocked sequences that contains only a single occurrence of s in b. An illustration of this definition is given in Example 3.

Example 3*:* From Table 3.1 we can extract k-solid ranged sequences with $\gamma = \frac{1}{2}$: SR_1= $<a>$, $<b>$, $<c>$, $<e>$, $<x>$; SR_2= $<a, c>$, $<e, x>$, $<a, b>$, $<b, c>$; SR_3= $<a, b, c>$. Considering a δ of $\frac{1}{2}$, it is possible to observe the following solid blocked sequences according to Definition 12: Four SB_1: ($<a>$, $([9..10], [3..4]), \frac{1}{2}$), ($<b>$, $([7..10], [5..5]), \frac{3}{4}$), ($<c>$, $([7..10], [6..6]), \frac{3}{4}$), ($<e>$, $([3..3], [1..2]), 1$); Three SB_2: $(<a, b>, ([7..10], [3..5]), \frac{7}{12})$, $(<a, c>, ([7..10], [3..6]), \frac{5}{8})$, $(<b, c>, ([7..10], [5..6]), \frac{3}{4})$; One SB_3: $(<a, b, c>, ([7..10], [3..6]), \frac{5}{8})$.

We can observe that using solid blocked sequences definition, it is now possible to filter some of the patterns extracted from solid ranged sequences. According to example, the sequence of size 3 (*i.e.* $<a, b, c>$) occurs on a very specific range (*i.e.* $[7..10]$) and interval (*i.e.* $[3..6]$). Additionally, sequence (*i.e.* $<a, c>$), although being supported by the entire dataset with frequency

of $\frac{1}{2}$, is only solid blocked sequence in block $(r = [7..10], i = [3..6])$ with item-frequency of $\frac{5}{8}$. Yet, we can observe that although sequences $<e, x>$ is solid ranged frequent, it is not solid blocked frequent, since its item-frequency is $\frac{1}{3}$.

3.2 General Principle and Algorithm

This section is devoted to the presentation of $STSM$ (Spatio-Temporal Sequence Miner), our algorithm designed for the identification of solid blocked sequences in spatio-temporal dataset. The notion of kernels [Saleh and Masseglia, 2008] (spatial-kernels and blocked-kernels) introduced in this section, allows for extracting solid blocked sequences efficiently. First, we give an overview of our main algorithm in Section 3.2.1. Later, each relevant function is described in the following sections.

3.2.1 General Principle

Algorithm 3 aims to generate solid blocked sequences (SB) from size 1 to k. It receives as input a spatial-TSS 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 3.2.2). 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 frequency greater than or equal to γ (detailed in Section 3.2.3). Then, solid blocked sequences SB_k with item-frequency greater than or equal to δ are computed (detailed in Section 3.2.4). Later, 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 and all solid blocked sequences are returned.

Algorithm 3 Spatio-Temporal Sequence Miner ($STSM$)

```
1:  function STSM(D, γ, δ)
2:      C_1 ← generateCandidates(D)
3:      k ← 0
4:      repeat
5:          k ← k + 1
6:          SR_k ← selectSolidRangedSequences(D, C_k, γ)
7:          SB_k ← selectSolidBlockedSequences(D, SR_k, δ)
8:          C_{k+1} ← generateCandidates(D, SR_k)
9:      until C_{k+1} ≠ ∅
10:     return {SB_1, ··· , SB_k}
11: end function
```

3.2.2 Generation of Candidates

Frequent pattern mining algorithms are compromised in 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 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 $xr(s_x, r_x, fr_x)$ be a ranged sequence that is not a solid ranged sequence (*i.e.* $fr_x < \gamma$). 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.* $fr_y < \gamma$).

However, the counting step (or "pruning" in apriori) is not straightforward. It is not enough to evaluate the support of a candidate on a given range. Let us, for instance, consider, in Table 3.1, $c = (<a, b>, [1..10], fr_c)$, a candidate ranged sequence that has been generated thanks to the solid ranged sequences of size 1: $x = (<a>, [1..10], \frac{6}{10})$ and $y = (<b>, [1..10], \frac{6}{10})$. In this case, for $\gamma = \frac{1}{2}$, c is not a solid ranged sequence since $fr_c = \frac{4}{10}$. However, there exists a ranged sequence $r_c \subset c$ that is a solid ranged sequence, $r_c = (<a, b>, [7..10], \frac{3}{4})$.

3.2.3 Selection of Solid Ranged Sequences

In this section, we give the formal properties of a spatial kernel used in the selection of solid ranged sequences described in Algorithm 4.

Definition 13*:* A **spatial 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 spatial kernels for the sequence s over a range r with respect to 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 occurrence of s 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 the loss of the frequency for s). If P is empty then k_e is defined as the last occurrence of s 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 spatial kernels is encapsulated by function $ExtractKernels$ of Algorithm 4. Intuitively, spatial kernels of a sequence are the longest ranges such that:

1. The first and last records support the sequence.
2. 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 4 Identifying Solid Ranged Sequences

```
1: function selectSolidRangedSequences(D, C_k, γ)
2:     SR_k ← ∅
3:     extractKernels(D, C_k, γ)
4:     for (c ∈ C_k) do
5:         mergeKernels(c.s, c.skernels, mRS, freq, γ)
6:         for (r ∈ c.skernels) do
7:             SR_k ← SR_k + (c.s, r, frequency(c.s, r))
8:         end for
9:     end for
10:    return SR_k
11: end function
```

After extracting spatial kernels for all candidate sequences C_k, whenever possible, Algorithm 4 merges them. Based on Lemma 1 (Section 3.2.5), Algorithm $MergeKernels$ makes it possible to find solid ranged sequences of s over r with respect to threshold γ. Given two spatial kernels t and u, Algorithm $MergeKernels$ tries to merge them into v using function $mRS(t, u)$, which simply combines their ranges: t, u. If v has a $freq$ for s greater than or equal to γ, v is included in $skernels$ 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 4, all merged kernels are used to produce solid ranged sequences SR_k.

3.2.4 Selection of Solid Blocked Sequences

In this section, we give the formal properties of a block kernel used in the selection of solid blocked sequences described in Algorithm 5.

Definition 14*:* A **spatio-temporal kernel** is a minimum 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 not less than δ. Let $B(sr, \delta)$ be the set of all possible **spatio-temporal 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 spatio-temporal kernel $b \in B(sr, \delta)$, let o be the set of occurrences of sr in b: $o = occurrences(s, b)$. If $|o| = 1$, then according to this definition, $ifreq(s, o) \geq \delta$. We name this case as trivial spatio-temporal kernel. However, if $|o| > 1$, then there exists a partition of b of size 2: b_i and b_j, such that $ifreq(s, b_i) \geq \delta \wedge ifreq(s, b_j) \geq \delta \wedge o = occurrences(s, b_i) \cup occurrences(s, b_j)$.

Definition 15*:* Let $o = occurrences(s, b)$ of a spatio-temporal kernel b. Whenever $|o| > 1$, $(s, b, ifreq(s, b))$ is a solid blocked sequence.

From the aforementioned definition, Algorithm 5 computes all solid blocked sequences. It starts computing all trivial spatio-temporal kernels. Then, based on Lemma 2 (Section 3.2.5), Algorithm $MergeKernels$ makes it possible to find solid blocked sequences of s over sr with respect to threshold δ. Given two spatio-temporal kernels t and u, Algorithm $MergeKernels$ tries to merge them into v using function $bound(t, u)$, which produces a bounding box for t and u. If v has a $ifreq$ for s greater than or equal to δ, v is included in $stkernel$ 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 5, all merged spatio-temporal kernels are used to produce solid blocked sequences SB_k.

Algorithm 5 Identifying Solid Blocked Sequences

```
1:  function selectSolidBlockedSequences(D, SR_k, δ)
2:      SB_k ← ∅
3:      for (sr ∈ SR_k) do
4:          spatialTimeKernels(sr, δ)
5:          mergeKernels(sr.stkernels, bound, ifreq, δ)
6:          for (b ∈ sr.stkernels) do
7:              SB_k ← SB_k + (s, b, freq(s, b.r), ifreq(s, b))
8:          end for
9:      end for
10:     return SB_k
11: end function
```

3.2.5 Merging Kernels

Let us consider that we are provided with a sequence s and K, the set of kernels of s with respect to a threshold σ. Based on Lemma 1 and on Lemma 2 we respectively show that merging the kernels with Algorithm $MergeKernels$ makes it possible to find both solid ranged sequences of s over r with respect to γ and solid blocked sequences of s over sr with respect to δ. In order to do so, Algorithm $MergeKernels$ receives as input $mfunc$, which is a function that actually merges kernels, and $thresfunc$, which is a function that computes the frequency used to evaluate if a merged kernel respects threshold σ.

Lemma 1*:* Let K be the set of spatial kernels of s on r with respect to γ. Algorithm $MergeKernels$ makes it possible to find all the solid ranged sequences $sr = (s, r, fr)$ on r.

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

1. $freq(s, k) \geq \gamma$. According to Definition 13, s is frequent on each kernel. Furthermore, if k is

Algorithm 6 Algorithm Merge Kernels

```
1: function MergeKernels(s, L, mfunc, thresfunc, σ)
2:     mergeable ← true
3:     while mergeable do
4:         mergeable ← false
5:         for t ∈ L do
6:             for u ∈ L, such that u > t do
7:                 v ← mfunc(t, u)
8:                 if thresfunc(s, v) ≥ σ then
9:                     L ← L + v
10:                    K ← K + t + u
11:                    mergeable ← true
12:                end if
13:            end for
14:        end for
15:        for k ∈ K do
16:            L ← L − k
17:        end for
18:        K ← ∅
19:    end while
20: end function
```

the result of a merging process, then Algorithm $MergeKernels$ checks the frequency of s on the resulting range.

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

   - $sup(s, q-k) > 0$, then s is not frequent on q (otherwise, let us consider k' the kernel to which belongs the occurrence of x 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 on q or not, depending on the size of q).

3. According to Definition 13, s is supported by the first and the last spatial-TSS in k. Then, s will have a 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 with respect to γ, then SR_x is the set of all solid ranged sequences $sr = (s, r, fr)$ on PR with respect to γ.

Lemma 2: Let K be the set of spatio-temporal kernels of s on sr with respect to δ. Algorithm $MergeKernels$ makes it possible to find all the solid blocked sequences $sb = (s, b, ifr)$ on sr.

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

1. $ifreq(s,k) \geq \delta$. According to Definition 14, s is item-frequent on each kernel. Furthermore, if k is the result of a merging process, then Algorithm $MergeKernels$ checks the item-frequency of s on the resulting block.

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

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

3. According to Definition 13, s touches the boundaries of k. Then, s will have a lower support on any sub-block of k.

   Based on the three observations above, let $B_x = \{(s,b,ifr)\ \forall b \in K\}$ be the set of blocked sequences corresponding to all the merged spatio-temporal kernels of s on $PB(sr)$ with respect to δ, then B_x is the set of all solid blocked sequences $sb = (s,b,ifr)$ on $PB(sr)$ with respect to δ.

Chapter 4 Experimental Evaluation

This Chapter presents the experimental evaluation using a seismic real use case. Section 4.1 starts presenting the spatio-temporal dataset used in the experiment. Sections 4.2 and 4.3, respectively, present the parameters used for both data-preprocessing discretization and $STSM$ algorithm. Section 4.4 discusses the behavior of the algorithm and the quality of identified sequences from the seismic perspective.

4.1 Input Dataset

The dataset chosen as input for this work is part of the *Netherlands Offshore F3 Block* reflection seismic survey [OSR, 2016]. *F3* is the name of a block of the Dutch sector of the North Sea, a rectangular area of 24 x 16 kilometers where this 3D seismic dataset was collected. The area is located between United Kingdom, Netherlands and Denmark, as depicted in Figure 4.1.



Figure 4.1: *Netherlands Offshore F3 Block* survey area of the chosen seismic dataset, located between United Kingdom, Netherlands and Denmark.

Seismic surveys are commonly collected by producing artificial shots on 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 in 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 to the wave traveled distance. Thus, the early received signals correspond to upper subsurface layers, and vice-versa. A representation of a seismic survey process is depicted in Figure 4.2

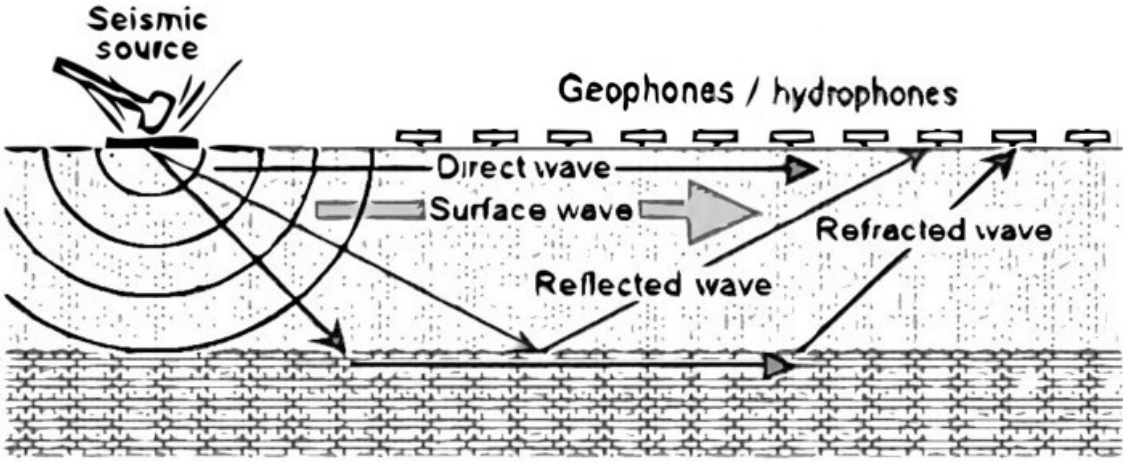
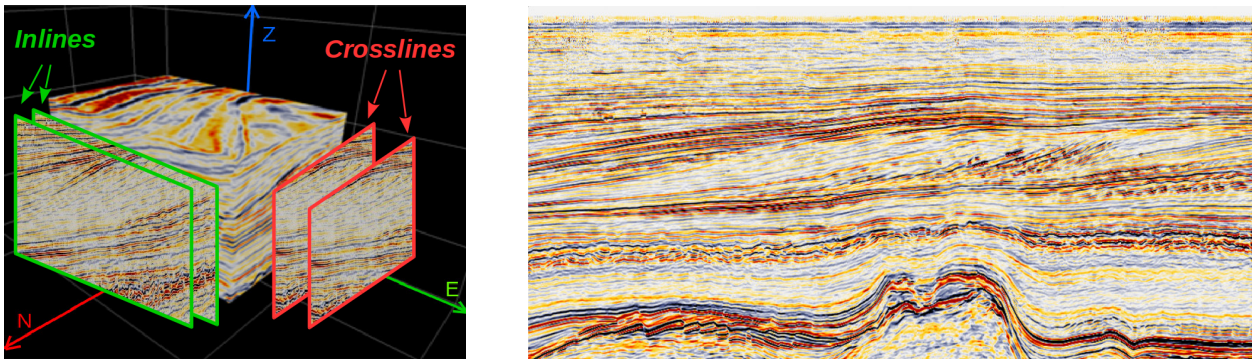


Figure 4.2: Representation of a seismic survey process. An artificial shot is produced on the ground. Waves propagates in the subsurface and are reflected and refracted when encounter different material layers. Receivers positioned in the surface collects returning waveforms and wave arrival times.

The dataset is composed of a range of 100 to 750 *inlines* and 300 to 1250 *crosslines* that can be considered as orthogonal sections over the E, Z plan and the N, Z plan, respectively, of the cube depicted of Figure 4.3.a, that is a partial 3D survey representation generated using the OpendTect software [OpendTect website, 2016]. Each *inline* or *crossline* is composed of several sets of observations made at different positions on the surface, at a distance of 25 meters between each consecutive position. Each observation time, one each four milliseconds, corresponds to a specific depth of the subsurface. In fact, the subsurface depth is measured in milliseconds of wave arrival times time, and this dataset depth range is between 0 to 1848 milliseconds, containing 462 observations. According to OpendTect software, the survey cover a depth range from 0 to about 2000 meters. As it can be depicted in Figure 4.3.b, each spatial-time series can be interpreted as vertical columns of pixels, where the more negative amplitude values correspond to red pixels, amplitudes around zero correspond to light pixels and the more positive amplitudes correspond to black pixels.

The results and graphics presented in this work are focused on *inline* 401. The choice for presenting this particular *inline* is due to the fact that it has a lower superficial layer composed of a homogeneous material. Datasets with large homogeneous superficial layer produces uninteresting patterns that are not useful to analyze. Yet, the algorithm presented a similar behavior in



(a) **(b)**

Figure 4.3: Visual representation of used seismic dataset, obtained from the *OpendTect* software. Image (a) shows a partial 3D representation of the *Netherlands Offshore F3 Block* dataset, while Image (b) shows *Inline* 401 of the same dataset.

different inlines.

Inline 401 dataset, like each other *inline*, is composed of 951 spatial-time series, each of them produced by a receiver located in a specific position on the surface. Each time series contains 462 observations representing wave arrival times and waveforms generated at different depths of the subsurface. The dataset was extracted to a *Comma Separated Value* (CSV) file format, from which the discretization was performed.

4.2 Discretization setup

The floating values that compose such spatial-time series are discretized to produce a spatial-TSS dataset, which is the input for $STSM$ algorithm. SAX algorithm [Lin et al., 2003b] was chosen to perform the discretization. Such algorithm use a symbolic representation to convert k bins of equiprobable distribution into a symbol from an alphabet of size k chosen by the user.

Function $Func.SAX(norm, w, a)$ of the *TSMining* packages in R programming language was used to perform such discretization. As SAX algorithm allows to apply a dimensionality reduction as well as a symbolic discretization of the input dataset, the $Func.SAX$ function accepts the parameters $norm$ that indicates if the data must be normalized or not, w that defines the desired output dimensionality (*i.e.* the *word size*), and a that defines the desired output *alphabet size*, *i.e.* the number of different symbols that the algorithm use to represent the the original dataset in a discrete form.

All the experiments presented in this work use normalization of the dataset (as the original SAX algorithm do [Lin et al., 2003a]) and a *word size* value equal to the SAX input dataset size, due to the fact that we are not interested to the dimensionality reduction feature of the SAX algorithm.

To apply the SAX representation equally to all time series of the input dataset, all the time

series was concatenated together to form a unique, bigger time series. Then, this single time series was converted to the SAX representation and the output was split again to the original multiple time series format. Thus, the dimensionality of the dataset used as input for the SAX algorithm is the product of the number of the time series multiplied the number of observations in each time series.

Figure 4.4 shows an example result of SAX discretization, using 25 as the value of alphabet size, of a sample dataset of *inline* 100. Such *inline* was used in this example due its characteristic and well visible zero pattern in the more superficial layers, as can be visible in the left image of Figure 4.4. It is possible to identify that the gray area of the image corresponds to the *"0"* vertical bands shown in the CSV original data format depict in the top right of the Figure 4.4, which in turns correspond to the *"m"* vertical bands shown in the SAX discretized data that is shown in the bottom right.

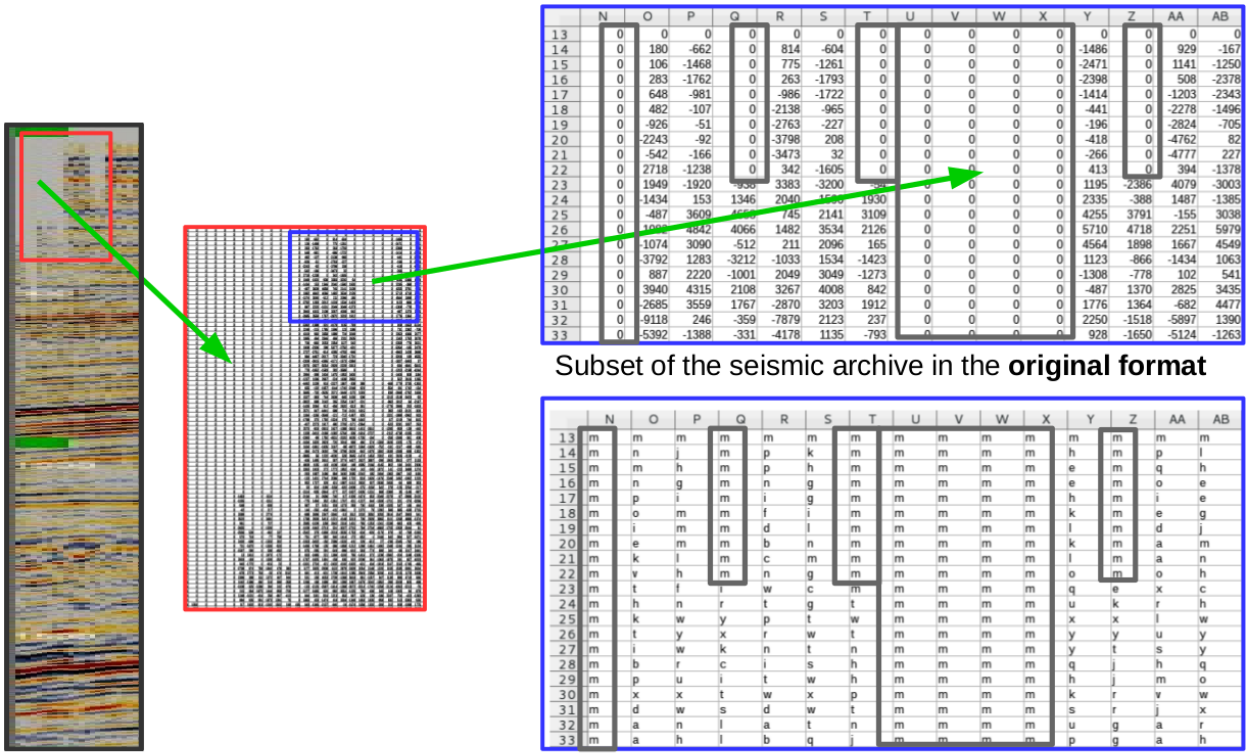


Figure 4.4: Representation of the data discretization. A portion of the original seismic image is shown in the left side, a portion of such image extracted to the CSV format is shown in the middle of the Figure, a fragment of such image is shown expanded in the top right, and the discretized equivalent data is shown in the bottom right.

The usage of a small alphabet size for SAX algorithm can lead to discover too much frequent sequences occurrences that are, in fact, not very similar. Conversely, a large alphabet size may avoid the discovery of many sequences occurrences. Thus, we have explored different alphabet size values of 5, 10, 15, 20, and 25 to find the one that leads to better results. The symbols that the SAX algorithm uses to represent the original dataset floating values vary according to each

alphabet size, so that the symbol ranges $\{a-e\}$, $\{a-j\}$, $\{a-o\}$, $\{a-t\}$ and $\{a-y\}$ are used to represent the discretized data using alphabet sizes of 5, 10, 15, 20, and 25, respectively.

Figures 4.5.a and 4.5.b respectively show the total number of sequences occurrences and the total number of sequences that appear in solid blocks. In both Figures, as we increase the alphabet size, the number of occurrences and the number of identified sequences decrease. Additionally, Figure 4.5.c presents the total number of sequences that appear in solid blocked sequences with at least 5 occurrences. In this case, the number of identified sequences decreases significantly. This is interesting since the number of different sequences to explore reduces as we increase constraints in both time and space.

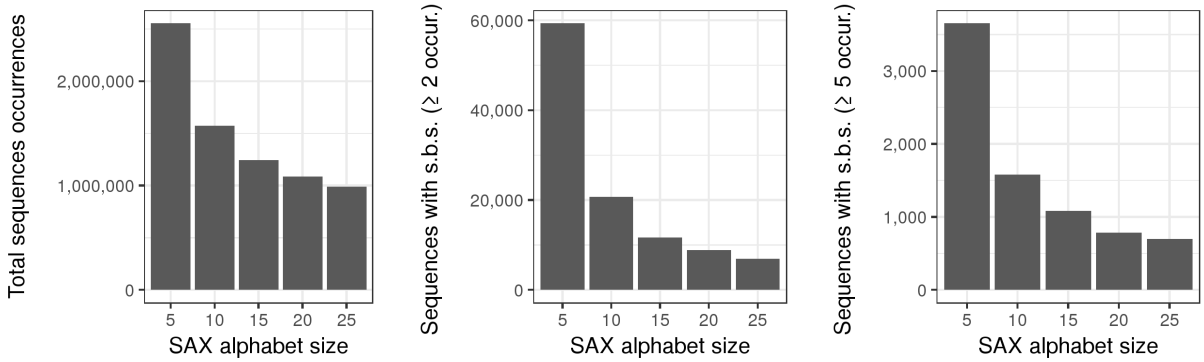


Figure 4.5: Analysis of *inline* 401 showing the total sequences occurrences (a), the number of sequences that appear in solid blocked sequences (b), the number of sequences that appear in solid blocked sequences with at least 5 occurrences (c). In all charts, SAX alphabet sizes varied from 5 to 25, solid range threshold γ was set to 80% and solid block threshold δ was set to 20%.

In order to choose the SAX alphabet size for our experiment, we applied an approach similar to the *elbow method* [Han et al., 2011]. From observations of Figure 4.5, we adopt an alphabet size equals to 10, since an additional increase of this parameter does not reduce significantly the number of sequences, and vice-versa.

4.3 STSM Parameters Setup

The algorithm proposed in this work allows for users to set solid range threshold γ and solid block threshold δ constraints. The former drives the identification of solid ranged sequences, keeping only ranged sequences that satisfy the spatial constraint frequency. The latter drives the identification of solid blocked sequences, keeping only blocks that contain occurrences that are frequent on them (*i.e.*, occurrences that are constrained according to space and time).

Finding adequate values for these thresholds depends on the characteristics of input dataset. Generally, 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 lead to the

detection of a small number of frequent sequences that may become too small to be interesting. For such reasons, in our experiments we empirically 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 the sake of comparison, we have used the sequence $<a, a, a>$ to analyze the behavior of both thresholds. This choice is reasonable since this sequence is produced by $STSM$ algorithm in all different threshold combinations that we had explored. Additionally, "a" is a symbol that is common whatever alphabet size is used, so it is also suitable to compare results among different alphabet sizes.

Figure 4.6 shows an example containing solid-blocked sequences (displayed using red rectangles) for the $<a, a, a>$ sequence obtained from *inline* 401 discretized using a SAX alphabet size of 10. It varies the solid range threshold γ growing from left to right with the values 60%, 80%, and 100% as well as the solid block threshold δ growing from top to bottom with the values 10%, 20%, and 30%.

It is important to observe that the analyzed sequence $<a, a, a>$ is frequent throughout the entire dataset. In this scenario, when γ is under 80%, a single solid range is detected covering the entire dataset, see Figures 4.6 (a,b,d,e,g,h). 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. As it can be observed in Figures 4.6 (e) and (d), relaxing γ from 80% to 60% led to an increase of approximately 1.5 hours in computing time, without improving the detection of solid blocked sequence. Adopting the extreme value of 100% may, however, limit too much the detection of solid block sequences. In this context, we opted for setting the γ parameter to 80%.

Similarly, examining the solid block threshold δ, when relaxing it to 10%, many larger solid blocked sequences are identified with many overlapping among 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 loses their overlapping. When this threshold is close to 20%, we observe a better balance in the number of overlapping blocks. Thus, Figure 4.6.e is obtained using solid range threshold γ of 80% and solid block threshold δ of 20%, which are the thresholds adopted in all following analysis.

4.4 Discussions

Adopting $\gamma = 80\%$ and $\delta = 20\%$ as the parameter settings for the $STSM$ algorithm leads to the identification of a large number of sequences, approximately 20,000, in the seismic dataset, as shown in Figure 4.5.b. To facilitate the analysis, we constrained the results to only include

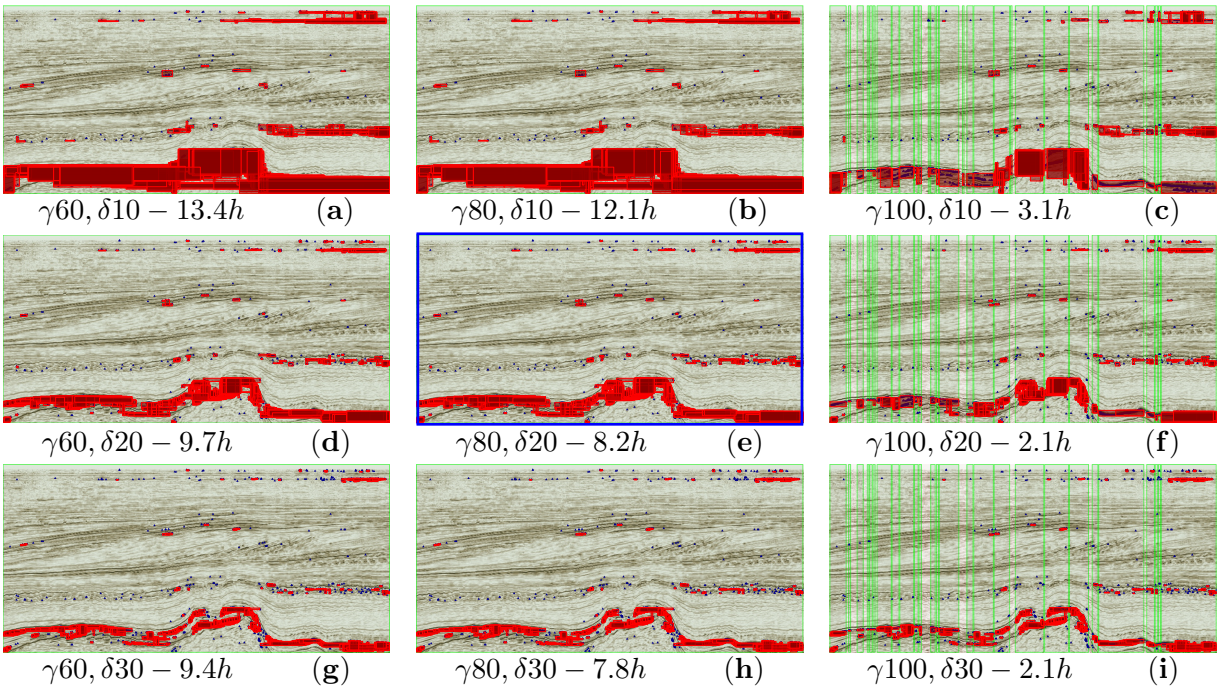


Figure 4.6: Comparison of solid range threshold γ and solid block threshold δ for sequence $<a, a, a>$ discovered using *inline* 401 and SAX alphabet size 10. From left to right, γ grows (60%, 80%, and 100%) whereas from top to bottom, δ grows (10%, 20%, and 30%). The less restrictive configuration is (a) and the more restrictive is (i). Configuration (e), evidenced in blue, seemed the most appropriate configuration and was adopted in all following analysis. Green vertical lines indicate boundaries for solid ranged sequences. For each configuration, the elapsed time in hours is presented.

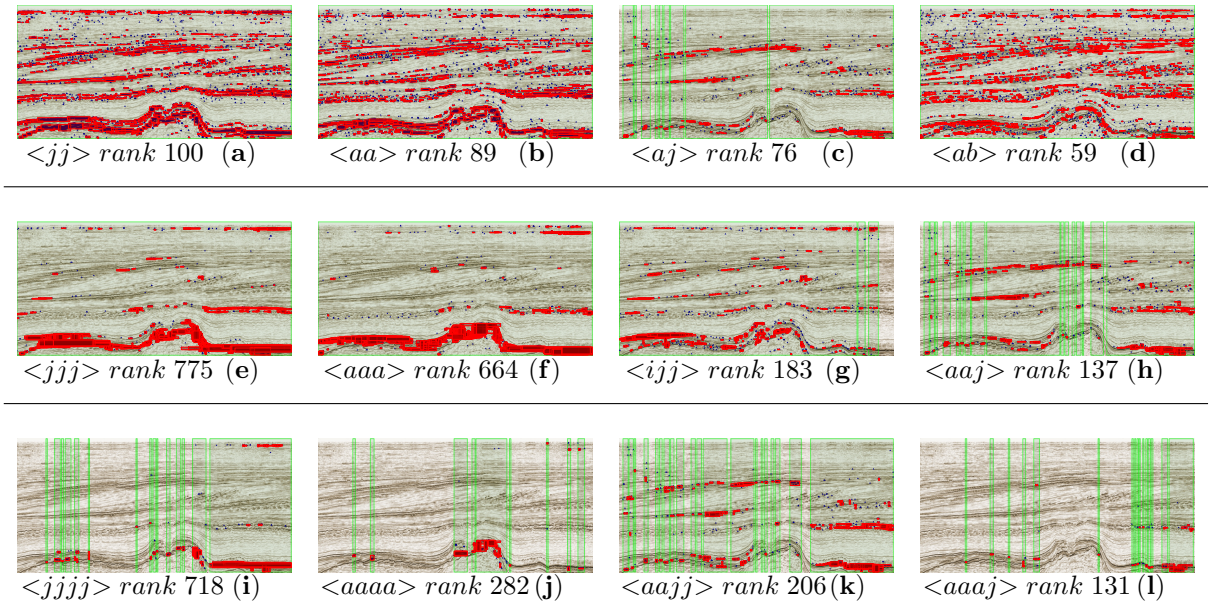
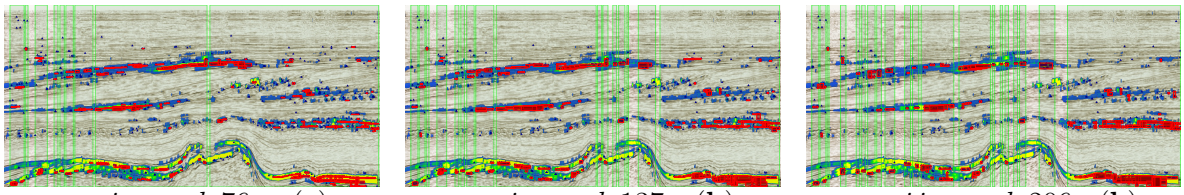


Figure 4.7: Identified sequences ranked according to their density for *inline* 401, alphabet size 10, solid range threshold γ 80% and solid block threshold δ 20%. Images from (a) to (d), (e) to (h), and (i) to (l) respectively present the four best ranked patterns for sequences of size two, three, and four.

sequences appearing in solid blocked sequence with at least five occurrences. This reduces the number of sequences to about 1,500, which corresponds to filter up to 92.5% of frequent sequences. Moreover, in order to ease the evaluation of the results, we ranked the identified sequences by density order. The density of a result is obtained computing the mean block size of its blocks obtained from the sum of all block areas divided by the number of blocks. However, considering the seismic dataset, this simple logic would prioritize some uninteresting patterns that occurs in the superficial layer of sediments. Consequently, to avoid the influence of those uninteresting superficial sequences, the density was computed excluding the first 10% of observations from all spatial-TSS, that corresponds to the first 200 meters of depth in subsurface.

Despite the simplicity of such ranking and the absence of any significant seismic concept in its logic, the obtained rank was good enough to prioritize interesting results, such as *horizons* and *bright spots*. Yet, it was also difficult to obtain interesting results in lower ranks. Such results were visually plotted over the original seismic trace images. Figure 4.7 shows the best ranked results obtained from the chosen configuration, depicted in Figure 4.6.e of *inline* 401. It can be observed in Figure 4.7 that sequences of size two cover the majority of known horizons, but as their rank decrease, more noise in blocks are observed. In addition, as sequences size increase, matching horizons become more precise, however coverage is reduced.

In the seismic domain, it is particularly important to identify *horizons*, *i.e.*, zones of major nonconformities between geologic boundary materials, usually associated to different lithologies, that produces variations in the collected reflection values [Zhou, 2014a]. $STSM$ algorithm was



$<aj>\ rank\ 76 \quad (\mathbf{c})$ $<aaj>\ rank\ 137 \quad (\mathbf{h})$ $<aajj>\ rank\ 206 \quad (\mathbf{k})$

Figure 4.8: Potential *bright spots* identified using the proposed algorithm for *inline* 401, alphabet size 10, γ = 80%, and δ = 20%. The results follow the blue-yellow pattern produced using the previously known *bright spots* for this dataset [OpendTect website].

able to identify interfaces between different lithologies, known as *horizons*, identified as sequences of observations repeated horizontally at similar time (depth). The majority of sequences plotted on Figure 4.7 match known seismic *horizons* previously known for this dataset [OpendTect website].

Furthermore, $STSM$ is able to identify areas of potential 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 due to the fact that it can be near to rocks containing oil.

A wave phase inversion may happen when the wave travels from a more dense layer to a less one or vice-versa. The density of a layer depends on the material that it contains, so that sediments have less density then other crust rocks, as well as crust rocks have less density than mantle rocks. Moreover, rocks are porous and they may contain gas, which has very low density, as well as water or oil that have a higher density then gas. Wave inversion caused from porous rocks containing water or oil in a layer and gas in a nearby layer may characterize a very small but important pattern called *bright spot*.

Examples of potential *bright spots* identifications are the previously presented ranked patterns $<a, j>$, $<a, a, j>$and $<a, a, j, j>$that can be shown, respectively, in Figures 4.8.a, 4.8.b, and 4.8.c. In these images, the discovered patterns, evidenced in red, are plotted on the top of a new blue-yellow marks obtained from the previously known *bright spots* for this dataset [OpendTect website]. Figure 4.8 shows that $STSM$ algorithm has discovered patterns that cover large parts of known *bright spots*.

Chapter 5 Related works

Sequential pattern mining is a broad field of research embracing several approaches. Most of techniques evolved from frequent patterns mining in transactional databases [Agrawal et al., 1993b; Mannila et al., 1994] to sequential pattern mining [Agrawal and Srikant, 1995b; Mannila et al., 1997; Pei et al., 2004]. Over the last decades, some works have been developed for sequential mining of spatio-temporal datasets [Mooney and Roddick, 2013]. We have conducted a systematic map study querying Scopus using the keywords $(\text{“}sequence\ mining'' \vee \text{“}sequential\ pattern'') \wedge (\text{“}space\text{-}time'' \vee \text{“}spatio\text{-}temporal'')$. We identified 61 articles that were related with sequential pattern mining of spatio-temporal datasets. We have classified these works as either moving or immovable objects [Frank, 2003].

We have observed 33 articles that were classified as moving objects. In this context, spatio-temporal datasets refer to objects that vary either their position or shape according to time [Frank, 2003]. Data may refer to objects like urban vehicles movements or mobile phones positions. The analysis of such data can, for example, aim to compute routes frequently used [Cao et al., 2005], trajectory clustering [Pelekis et al., 2009] or trajectory classification [Dodge et al., 2009]. Such branch of analysis clearly differs from the scope of our work.

When it comes to immovable objects, spatio-temporal datasets can be obtained from sensors monitoring specific locations [Frank, 2003]. We found 28 articles that follow such category. Examples of application are pattern detection on satellite imaging [Julea et al., 2011], minimization of traveling distance [Zhou and Meng, 2011], prediction of next point of interest [Zhang and Chow, 2015], and search for wind patterns [Yusof and Zurita-Milla, 2017].

Alatrista-Salas et al. [2015] analyze water quality of a hydrological network of rivers. It uses spatial data, like station proximity or watercourse, to aggregate stations in zones. Then, it uses PrefixSpan [Pei et al., 2004] for previously aggregated data to obtain sequential patterns that corresponds to spatially frequent temporal evolution of the water quality over zones. Later, it applies a clustering technique to group identified patterns according to their distances.

Such work evolved into two different algorithms for mining sequential patterns taking into account spatial neighborhoods [Alatrista-Salas et al., 2016]. Additionally, they evaluated the efficiency of an apriori approach (derived from SPADE [Zaki, 2001]) versus a pattern-growth approach (using pattern-growth strategy [Han et al., 2000a]). They conclude that the former algorithm per-

formance decreased drastically when producing sequences of larger lengths when compared to the latter.

Leong and Chan [2012] developed a variation of GSP algorithm [Srikant and Agrawal, 1996b] over a synthetic dataset of daily events, which were organized as spatio-temporal datasets. They evaluated the support count for each time series location. The authors used transactions to aggregate multiple observations by time, losing the time information inside a same day.

In our work, we do not reduce the number of dataset observations, preserving the original time granularity. Also, we use both spatial and temporal dimensions to discover patterns that are frequent into a maximal delimited space and time. To the best of our knowledge, we do not find similar works that studied frequent sequential patterns constrained in both space and time from spatio-temporal datasets.

Chapter 6 Conclusion

This work presented a challenging problem with high potential impact: spatio-temporal sequential mining that focuses on discovering frequent patterns, along with their constraints of frequency in both space and time. Our formal background provides us with the necessary conditions of such discovery. We proposed $STSM$, a novel algorithmic principle that performs an effective extraction of these patterns and their spatio-temporal constraints, with high efficiency. To the best of our knowledge, this is the first study on this topic. Our experiments, using a real-world seismic dataset, highlighted major insights according with the feedback of specialists in the domain. Actually, $STSM$ allows for extracting patterns that follow areas of major nonconformities between geologic boundary materials of subsurface, such as *horizons* and *bright spots*. The existence of these discovered patterns was confirmed comparing them with previously known survey on this dataset, while their meaning was assessed by a domain expert. Owing to its novelty and the characteristics of the extracted patterns, $STSM$ is opening new research tracks for seismic data analytics.

Although we have evaluated $STSM$ algorithm using seismic data, it has been conceived to be sufficiently generic for different spatio-temporal datasets (such as IoT domain). The plot of detected solid blocked sequences according to space and time, as showed in this work, can aid specialists to analyze identified sequences. Additionally, in scenarios in which a large number of frequent constrained sequences are produced, we can explore new ranking functions to prioritize the most interesting ones. Furthermore, let the user to define his own ranking function could highly improve the quality of discovered sequence patterns according to specific areas of application.

Indeed, a valuable further contribution of future works can be make available $STSM$ algorithm as a ready-to-use framework for the commonly used programming languages for data scientists, like R and Python. Such framework should allow the user for define a custom ranking function based on occurrence of the patterns, time and space constraints, as well as others parameters according with the dimensionality of the used input dataset. In this scenario, the user would have a powerful set of parameters to define complex and business specific ranking functions.

Another important feature that can be added to improve the dynamism of $STSM$ algorithm is the possibility of consider noise between the items of a sequence during the support count in the input dataset. The *min-gap* and *max-gap* thresholds were introduced in the GSP algorithm

[Srikant and Agrawal, 1996a] to allow gaps between sets of transactions that contains consecutive items of the sequence. Using a *max-gap* value of one, for example, will allow the match between the sequence pattern $<a, a, j, j >$and a TimeStamped Sequence (TSS) containing the items a, x, a, x, j, j, where x is a value different then a or j. In other words, it allows the existence of a item that not belongs to the sequence between each two sequential items that belong to the sequence. A similar constraint, *max-stretch*, was successful used in our previous work [Campisano et al., 2016] to achieve a similar behavior, where such constraint demonstrate to allow the detection of larger and more complex sequences.

On the other hand, *min-gap* as well as custom multi-dimensionality ranking functions can considerably increase the computational time consumed by $STSM$ algorithm. The *min-gap* constraint can increase the time needed to perform the scan of the TimeStamped Sequences to evaluate the support of each solid ranged sequence candidate. The multi-dimensionality ranking function can consume additional time to evaluate the rank of each detected solid blocked sequence. Furthermore, actually the slowest part of the $STSM$ algorithm is the merging of small block and the item-frequency calculation of each merged block, due to the high number of detected solid blocks. All such computational problems can be improved using embarrassing parallelism techniques. This is especially true for the support count step, where the data can be partitioned in subset of TimeStamped Sequences, as well as for the merging of solid blocks, where the data can be partitioned according to the detected solid ranges. In the terms of multi-dimensionality ranking function, the use of specialized tree index can improve the performance, improving the time needed to retrieve the parameters related to the sequence block from the input dataset.

References

Agrawal, R., Imieliński, T., and Swami, A. (1993a). Mining association rules between sets of items in large databases. SIGMOD Rec., 22(2):207–216.

Agrawal, R., Imieliński, T., and Swami, A. (1993b). Mining Association Rules Between Sets of Items in Large Databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD '93, pages 207–216, New York, NY, USA. ACM.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Agrawal, R. and Srikant, R. (1995a). Mining sequential patterns. In Proceedings of the Eleventh International Conference on Data Engineering, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society.

Agrawal, R. and Srikant, R. (1995b). Mining Sequential Patterns. In Proceedings of the Eleventh International Conference on Data Engineering, ICDE '95, pages 3–14, Washington, DC, USA. IEEE Computer Society.

Alatrista-Salas, H., Azé, J., Bringay, S., Cernesson, F., Selmaoui-Folcher, N., and Teisseire, M. (2015). A knowledge discovery process for spatiotemporal data: Application to river water quality monitoring. Ecological Informatics, 26:127–139.

Alatrista-Salas, H., Bringay, S., Flouvat, F., Selmaoui-Folcher, N., and Teisseire, M. (2016). Spatio-sequential patterns mining: Beyond the boundaries. Intelligent Data Analysis, 20(2):293–316.

Bayardo, Jr., R. J. (1998). Efficiently Mining Long Patterns from Databases. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, SIGMOD '98, pages 85–93, New York, NY, USA. ACM.

Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., and Yiu, T. (2005). MAFIA: a maximal frequent itemset algorithm. IEEE Transactions on Knowledge and Data Engineering, 17(11):1490–1504.

Campisano, R., Porto, F., Pacitti, E., Masseglia, F., and Ogasawara, E. S. (2016). Spatial sequential pattern mining for seismic data. In SBBD, pages 241–246.

Cao, H., Mamoulis, N., and Cheung, D. W. (2005). Mining frequent spatio-temporal sequential patterns. In Fifth IEEE International Conference on Data Mining (ICDM'05), pages 8 pp.–.

Dodge, S., Weibel, R., and Forootan, E. (2009). Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. Computers, Environment and Urban Systems, 33(6):419–434.

European Central Bank website (2016). Euro exchange rates USD. http://www.ecb.europa.eu/stats/exchange/eurofxref/html/eurofxref-graph-usd.en.html. Accessed April 03, 2016.

Frank, A. U. (2003). Ontology for spatio-temporal databases. In Spatio-Temporal Databases, pages 9–77. Springer.

Fu, T.-c. (2011). A review on time series data mining. Eng. Appl. Artif. Intell., 24(1):164–181.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: Current status and future directions. Data Min. Knowl. Discov., 15(1):55–86.

Han, J., Kamber, M., and Pei, J. (2011). Data Mining: Concepts and Techniques. Third edition edition.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000a). FreeSpan: frequent pattern-projected sequential pattern mining. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 355–359. ACM.

Han, J., Pei, J., and Yin, Y. (2000b). Mining Frequent Patterns Without Candidate Generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, pages 1–12, New York, NY, USA. ACM.

Julea, A., Meger, N., Bolon, P., Rigotti, C., Doin, M.-P., Lasserre, C., Trouvé, E., and Lazarescu, V. (2011). Unsupervised spatiotemporal mining of satellite image time series using grouped frequent sequential patterns. IEEE Transactions on Geoscience and Remote Sensing, 49(4):1417–1430.

Kang, J.-Y. and Yong, H.-S. (2010). Mining spatio-temporal patterns in trajectory data. Journal of Information Processing Systems, 6(4):521–536.

Leong, K. and Chan, S. (2012). STEM: A Novel approach for spatiotemporal sequence mining. Asian Journal of Information Technology, 11(3):94–99.

Lian, W., Cheung, D. W., and Yiu, S. M. (2007). Maintenance of Maximal Frequent Itemsets in Large Databases. In Proceedings of the 2007 ACM Symposium on Applied Computing, SAC '07, pages 388–392, New York, NY, USA. ACM.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003a). 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, pages 2–11, New York, NY, USA. ACM.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003b). 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, pages 2–11, New York, NY, USA. ACM.

Lucchese, C., Orlando, S., and Perego, R. (2006). Fast and memory efficient mining of frequent closed itemsets. IEEE Transactions on Knowledge and Data Engineering, 18(1):21–36.

Mabroukeh, N. R. and Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. ACM Comput. Surv., 43(1):3:1–3:41.

Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of Frequent Episodes in Event Sequences. Data Min. Knowl. Discov., 1(3):259–289.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1994). Efficient Algorithms for Discovering Association Rules. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94, pages 181–192, Seattle, WA. AAAI Press.

Mooney, C. H. and Roddick, J. F. (2013). Sequential Pattern Mining – Approaches and Algorithms. ACM Comput. Surv., 45(2):19:1–19:39.

OpendTect website. Free Open-source Seismic Interpretation Software System.

OpendTect website (2016). Free Open-source Seismic Interpretation Software System. http://www.opendtect.org/. Accessed April 04, 2016.

OSR (2016). Open Seismic Repository - Netherlands Offshore F3 Block. https://opendtect.org/osr/pmwiki.php/Main/NetherlandsOffshoreF3BlockComplete4GB. Accessed April 04, 2016.

Palshikar, G. K., Kale, M. S., and Apte, M. M. (2007). Association rules mining using heavy itemsets. Data & Knowledge Engineering, 61(1):93–113.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In International Conference on Database Theory, pages 398–416. Springer.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Trans. on Knowl. and Data Eng., 16(11):1424–1440.

Pelekis, N., Kopanakis, I., Kotsifakos, E., Frentzos, E., and Theodoridis, Y. (2009). Clustering Trajectories of Moving Objects in an Uncertain World. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM '09, pages 417–427, Washington, DC, USA. IEEE Computer Society.

Saleh, B. and Masseglia, F. (2008). Time aware mining of itemsets. In Temporal Representation and Reasoning, 2008. TIME'08. 15th International Symposium on, pages 93–97. IEEE.

Srikant, R. and Agrawal, R. (1996a). Mining sequential patterns: Generalizations and performance improvements. In Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '96, pages 3–17, London, UK, UK. Springer-Verlag.

Srikant, R. and Agrawal, R. (1996b). Mining Sequential Patterns: Generalizations and Performance Improvements. In Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '96, pages 3–17, London, UK, UK. Springer-Verlag.

Toivonen, H. (1996). Sampling Large Databases for Association Rules. In Proceedings of the 22th International Conference on Very Large Data Bases, VLDB '96, pages 134–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wang, J., Han, J., and Pei, J. (2003). CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, pages 236–245, New York, NY, USA. ACM.

Yusof, N. and Zurita-Milla, R. (2017). Mapping frequent spatio-temporal wind profile patterns using multi-dimensional sequential pattern mining. International Journal of Digital Earth, 10(3):238–256.

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

Zhang, J.-D. and Chow, C.-Y. (2015). Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach. ACM Transactions on Intelligent Systems and Technology (TIST), 7(1):11.

Zhou, C. and Meng, X. (2011). STS: Complex spatio-temporal sequence mining in Flickr. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 6587 LNCS, pages 208–223.

Zhou, H.-W. (2014a). Practical seismic data analysis. Cambridge University Press.

Zhou, H.-W. (2014b). Practical seismic data analysis. Cambridge University Press.

Appendix

Table 6.1: SAX alphabet symbol ranges by size.

SAX alphabet size	5	10	15	20	25
Symbol range	$\{a-e\}$	$\{a-j\}$	$\{a-o\}$	$\{a-t\}$	$\{a-y\}$

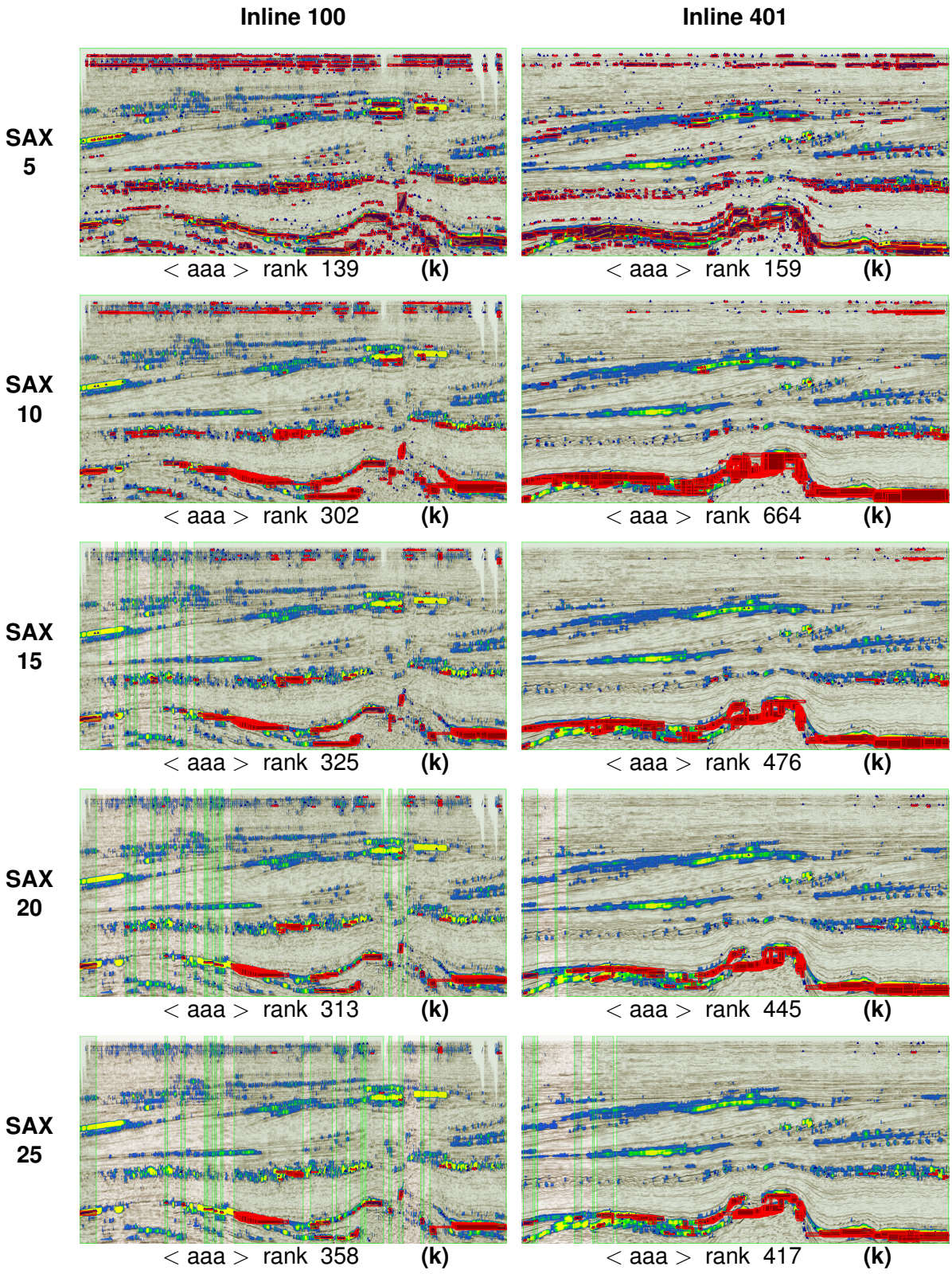


Figure 6.1: Inlines versus SAXs result comparison for sequence $<a, a, a>$ discovered using solid range threshold γ 80% and solid block threshold δ 20%.

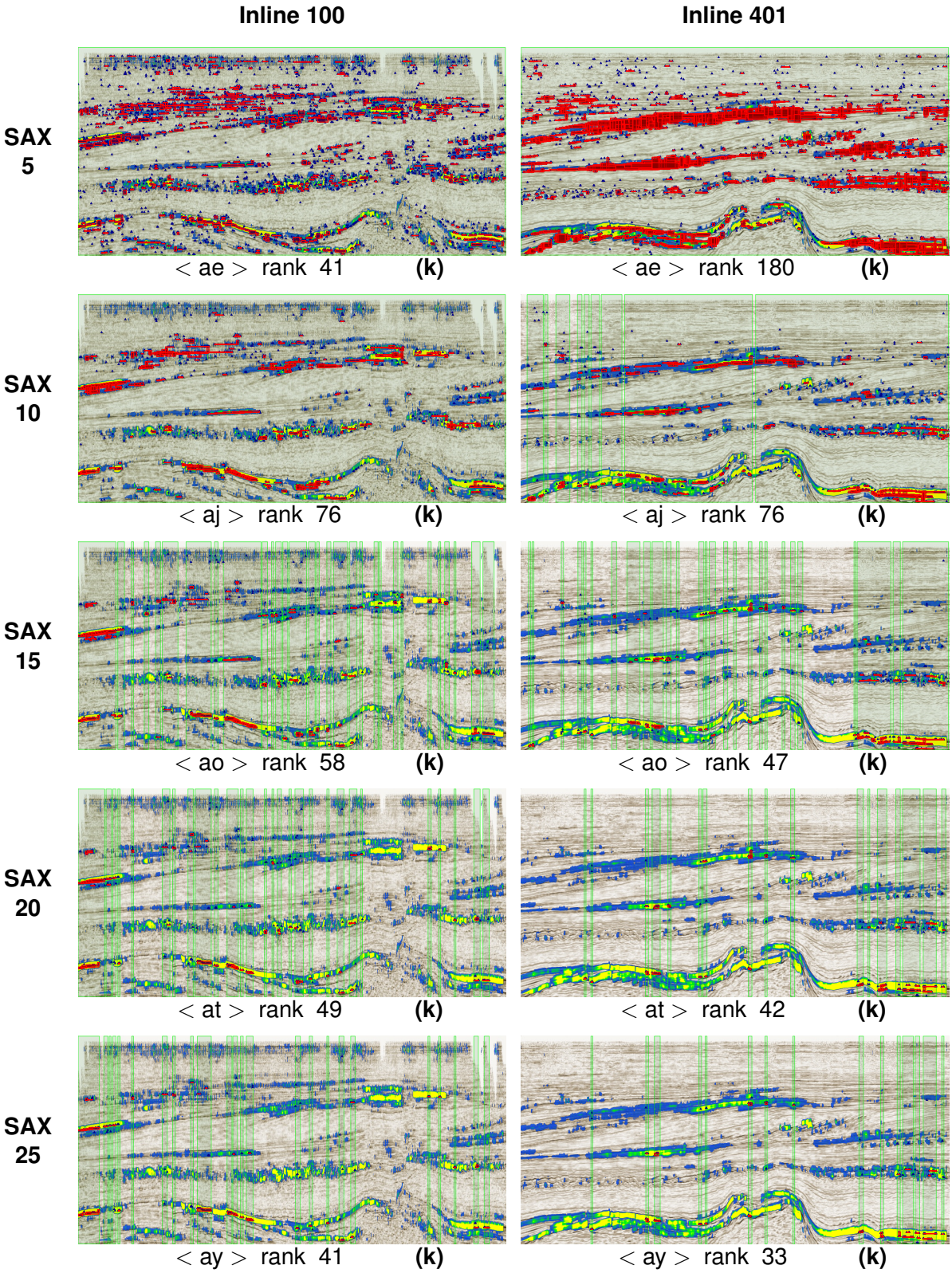


Figure 6.2: Inlines versus SAXs result comparison for sequences of size two, varying the symbols according to the SAX alphabet size related value. For example, the $<a, j>$ sequence of SAX 10 corresponds to the $<a, y>$ sequence of SAX 25. The sequences was discovered using solid range threshold γ 80% and solid block threshold δ 20%.

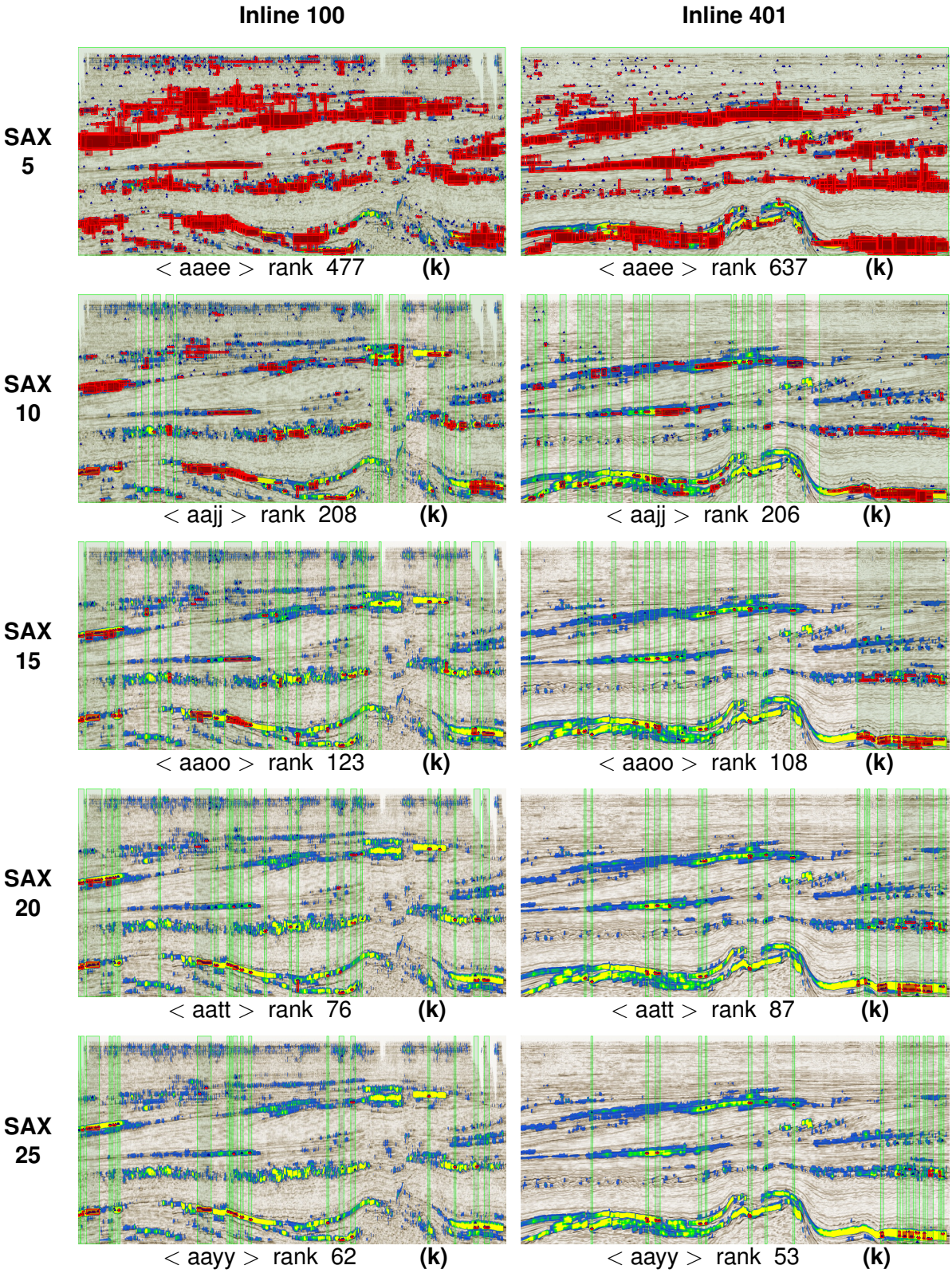


Figure 6.3: Inlines versus SAXs result comparison for sequences of size four, varying the symbols according to the SAX alphabet size related value. For example, the $<a, a, j, j>$ sequence of SAX 10 corresponds to the $<a, a, y, y>$ sequence of SAX 25. The sequences was discovered using solid range threshold γ 80% and solid block threshold δ 20%.