CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

MINERAÇÃO DE SEQUÊNCIAS RESTRITAS NO ESPAÇO E NO TEMPO

Antonio Castro

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientador(a): Rafaelli Coutinho
Coorientador(a): Eduardo Ogasawara

Rio de Janeiro,
Fevereiro 2021

MINERAÇÃO DE SEQUÊNCIAS RESTRITAS NO ESPAÇO E NO TEMPO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Antonio Castro

Banca Examinadora:

Presidente, Professora D.Sc. Rafaelli Coutinho (CEFET/RJ) (Orientador(a))

Professor D.Sc. Eduardo Ogasawara (CEFET/RJ) (Coorientador(a))

Professor D.Sc. Jorge Soares (CEFET/RJ)

Professora D.Sc. Esther Pacitti (INRIA)

Rio de Janeiro,

Fevereiro 2021

RESUMO

Mineração de Sequências Restritas no Espaço e no Tempo

Os padrões espaço-temporais trazem conhecimento sobre o tempo e a posição onde eles estão presentes. Encontrá-los é uma tarefa importante para diferentes domínios. No entanto, nem todos os padrões são frequentes por todo um conjunto de dados, eles podem ocorrer restritos no espaço e no tempo. A mineração desses padrões tem como objetivo descobrir a faixa de tempo e o conjunto de posições espaciais em que as sequências de eventos são frequentes. Este trabalho propõe o algoritmo *Generalized Spatial-Time Sequence Miner* (G-STSM) como uma solução para a descoberta de sequências frequentes que são restritas no espaço e no tempo, trazendo a formalização do problema, definições, provas e algoritmos. Até onde se sabe, após busca na literatura relacionada, o G-STSM é a primeira abordagem capaz de encontrar tais sequências trabalhando com uma dimensão de tempo e três dimensões de espaço. O G-STSM foi comparado com uma abordagem intuitiva que busca sequências de eventos frequentes com suporte muito baixo e agrupa suas ocorrências para encontrar padrões restritos no espaço e no tempo usando algoritmos conhecidos. Foi escolhido um conjunto de dados sísmicos espaço-temporal do mundo real para comparar ambas as abordagens usando métricas de classificação e registro de uso de recursos. Como resultado, o G-STSM apresentou melhor desempenho computacional com qualidade semelhante mostrando-se uma ferramenta de mineração de dados eficiente para encontrar sequências restritas no espaço e no tempo.

Palavras-chave: Séries Espaço-Temporais; Padrões Sequenciais; Mineração de Sequências; Mineração de Dados

ABSTRACT

Mining of Space and Time Constrained Sequences

Spatio-temporal patterns bring knowledge about time and position where they are present. Finding them is an important task for different domains. However, not all patterns are frequent over an entire dataset, they can occur constrained in space and time. Mining these patterns have as objective to discover the time range, and the set of spatial positions in which event sequences are frequent. This work proposes *Generalized Spatial-Time Sequence Miner (G-STSM)* algorithm as a solution for the discovery of frequent sequences that are constrained in space and time, bringing the formalization of the problem, definitions, proofs and algorithms As far as is known, after searching the related literature, G-STSM is the first approach able to find such sequences working with one dimension of time and three dimensions of space. G-STSM has been compared with an intuitive approach that searches for sequences of frequent events with very low support and groups its occurrences to find patterns constrained in space and time using known algorithms. A set of real world space-time seismic dataset was chosen to compare both approaches using classification metrics and resource usage records. As a result, G-STSM presented better computational performance with similar quality and it proved to be an efficient data mining tool for finding tight space-time sequences.

Keywords: Space-Temporal Series; Sequential Patterns; Sequence Mining; Data Mining

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de sequência com marcações de tempo usada na medicina: eletrocardiograma. Fonte: ENEM 2016. 16

Figura 2 – Exemplo de sequência com marcações de tempo usada na economia: taxa de câmbio US$ X R$. Fonte: tradingeconomics.com. 17

Figura 3 – Diagrama UML referente as estruturas de dados utilizadas. Um candidato c possui um conjunto de Ranged Groups $c.rgs$. Ranged Group generaliza Kernel Range-Group, que por sua vez generaliza Solid Range-Group. 36

Figura 4 – Distribuição das posições utilizadas no exemplo. 51

Figura 5 – Processo de mineração utilizado pelo G-STSM. Retângulos com cantos arredondados e fundo cinza representam dados. Retângulos com fundo branco se referem a processamento. 53

Figura 6 – Obtenção de dados sísmicos marinhos. Ondas de som são enviadas ao fundo do mar, a partir de canhões de ar, refletidas e registradas por hidrofones. Fonte: gov.br/anp. 60

Figura 7 – Conjunto de dados T401, marcação dos padrões (linhas coloridas), e divisão em quadrantes (linhas retas verticais e horizontais em preto com numeração em vermelho). Fonte: OpendTect [2020]. 61

Figura 8 – Tempo total de execução para cada abordagem usando diferentes números de quadrantes. 65

Figura 9 – $acurácia$ dados β, σ e tamanho da sequência para ambas as abordagens. Áreas em cinza representam ausência de valor, ou seja, para os parâmetros de entrada não foram encontradas sequências frequentes. 66

Figura 10 – Correlação entre os parâmetros de entrada (γ, β e σ), o uso de recursos, e os resultados. 67

Figura 11 – Mediana de $acurácia$, $precisão$, $sensibilidade$, e f_1 dado o tamanho da sequência para o G-STSM. 68
Figura 12 – Tempo de execução do G-STSM usando diferentes configurações e tamanhos de conjunto de dados. 69
Figura 13 – Uso máximo de memória durante a execução do G-STSM para diferentes tipos de configurações e tamanhos de conjunto de dados. 70

LISTA DE TABELAS

Tabela 1 – Classificação dos trabalhos relacionados apresentando informações do método utilizado e das restrições aplicadas. 30

Tabela 2 – Conjunto de Dados STS. Nove TS, cada uma com três observações. 50

Tabela 3 – Estrutura de dados referente aos candidatos. 51

Tabela 4 – Resultado da execução do passo *1.a* - Criação dos grupos de posições para a primeira marcação de tempo. 52

Tabela 5 – Resultado da execução do processo até o passo *1.b* - Geração dos RGs referentes aos grupos de posições gerados. 54

Tabela 6 – Resultado da execução do passo *1.a* - Criação dos grupos de posições, para a segunda marcação de tempo. 54

Tabela 7 – Resultado da execução do passo *1.b* - Novos RGs criados a partir dos grupos de posições referentes à segunda marcação de tempo para todas as sequências candidatas. 55

Tabela 8 – Resultado da execução do processo até o passo *1.c* - União dos RGs abertos referentes à segunda marcação de tempo. 55

Tabela 9 – Resultado da execução do processo até o passo *1.d* - Validação dos RGs referentes à segunda marcação de tempo. 56

Tabela 10 – Resultado da execução do passo *1.a* - Criação dos grupos de posições para a terceira marcação de tempo. 56

Tabela 11 – Resultado do passo *1.b* - Geração dos RGs referentes aos grupos de posições da terceira marcação de tempo. 56

Tabela 12 – Resultado da conclusão do passo *1.c* referente à terceira marcação de tempo. 57

Tabela 13 – Resuldado da execução do passo *1* - Todos os KRGs ao fim da busca utilizando sequências de tamanho um. 57

Tabela 14 – Resultado do passo *2* - SRGs gerados a partir dos KRGs. 58

Tabela 15 – Resultado do passo *3* - Candidatos com sequência de tamanho dois gerados a partir dos SRGs de sequências de tamanho um. 58

Tabela 16 – Variação de quadrantes. 63

Tabela 17 – Parâmetros usados nos experimentos. 63

Tabela 18 – Média e desvio padrão das métricas qualitativas para ambas abordagens. 66

Tabela 19 – Configuração utilizada no cenário C. 68

LISTA DE ALGORITMOS

Algoritmo 1 – G-STSM 38
Algoritmo 2 – FindKernelRangeGroup 39
Algoritmo 3 – SplitGroups 40
Algoritmo 4 – CreateGroup 41
Algoritmo 5 – MergeOpenKernelRangeGroups 42
Algoritmo 6 – ValidateKernelRangeGroups 43
Algoritmo 7 – ValidateAndClose 44
Algoritmo 8 – MergeKernelRangeGroups 47
Algoritmo 9 – GenerateCandidates 49

LISTA DE ABREVIATURAS E SIGLAS

CRAN	*The Comprehensive R Archive Network*
G-STSM	*Generalized Spatial-Time Sequence Miner*
KRG	*Kernel Range-Group*
RG	*Ranged Group*
SRG	*Solid Range-Group*
STS	Sequência Com Marcação De Tempo E Espaço (do Inglês *Spatial Time-stamped Sequence*)
TS	Sequência Com Marcação De Tempo (do Inglês *Time-stamped Sequence*)

SUMÁRIO

1	**Introdução**	**13**
2	**Referencial Teórico**	**16**
2.1	Fundamentos	16
2.2	Mineração de Sequências	19
3	**Trabalhos Relacionados**	**21**
3.1	Bases de Dados de Trajetória	22
3.2	Bases de Dados de Posições Fixas	24
3.3	Comparação	28
4	**Metodologia**	**32**
4.1	Formalização do Problema	32
4.2	Algoritmo G-STSM	35
4.2.1	Princípio Geral	37
4.2.2	Seleção dos Kernel Range-Groups	39
4.2.3	União dos Kernel Range-Groups	46
4.2.4	Geração dos Candidatos	48
4.3	Exemplo	49
5	**Avaliação Experimental**	**59**
5.1	Conjunto de Dados	59
5.2	Métricas	61
5.3	Configuração Experimental	62
5.4	Análise Comparativa	64
5.5	Análise de Sensibilidade	66
6	**Conclusões**	**71**

Referências **72**

1- Introdução

A popularização de dispositivos digitais com sensores e GPS contribui para o surgimento de extensos conjuntos de dados acerca de diversas áreas de conhecimento com eventos relevantes que acontecem em um determinado momento no tempo e posição no espaço. Contudo, com o crescimento desses conjuntos de dados, torna-se difícil ou mesmo impossível analisá-los de forma não automatizada. Ainda assim a capacidade de analisar esses dados abre oportunidades para extrair padrões espaço-temporais interessantes [Huang et al., 2008]. Ser capaz de obter conhecimento de padrões existentes nesses conjuntos de dados é um diferencial importante. Dada a ocorrência de um evento A poder prever ou quantificar a probabilidade de um evento B, é uma informação de grande utilidade para tomada de decisões.

O uso de mineração de dados, dentro de um processo multidisciplinar como forma de descoberta de estruturas de interesse em grandes conjuntos de dados, possibilita tal análise [Hand, 2007; Fayyad et al., 1996; Alatrista-Salas et al., 2015]. Portanto, os algoritmos de mineração de dados têm sido aplicados para descoberta de padrões em uma grande diversidade de problemas. Estes algoritmos começaram com a busca de regras de associação e evoluíram para mineração de padrões sequenciais [Agrawal et al., 1993; Agrawal and Srikant, 1995]. Dessa forma a mineração de sequências restritas no espaço e no tempo tem se tornado importante para diversos domínios [Alatrista-Salas et al., 2016; Li and Fu, 2014; Huang et al., 2008; Geng and Hamilton, 2006].

No entanto, nem sempre a frequência de ocorrência de alguns padrões é grande por todo o conjunto de dados. Surge, então, a ideia de extrair eventos que sejam frequentes não por todo um conjunto de dados, mas por uma janela de tempo e de espaço.

Considere, por exemplo, que em dias normais de trabalho as ruas do Centro, assim como várias outras na cidade do Rio de Janeiro apresentam congestionamento antes e depois do expediente. Não é diferente para as ruas ao redor do estádio do Maracanã, onde um grande fluxo de veículos também causa congestionamento. De manhã, o fluxo intenso é no sentido centro da cidade e no final do dia, o sentido é o inverso. Um padrão que poderia ser facilmente encontrado para uma ferramenta de mineração de dados

devido ao alto suporte é que nas ruas do Rio de Janeiro, de manhã (no sentido centro da cidade) e ao final do dia (no sentido oposto), antes e depois do expediente, o grande fluxo de veículos causa congestionamento.

Porém, às quartas-feiras à noite, quando acontecem partidas de futebol no estádio do Maracanã, por volta das 22h com o término da partida, carros de transporte de passageiros começam a se aproximar para atender os torcedores que vão deixar o estádio, resultando em novos engarrafamentos nas ruas ao redor do estádio (em ambos os sentidos, indo e vindo do centro da cidade), não só pelo grande número de carros, mas também porque se movem lentamente à espera dos clientes. Esse tipo de padrão dificilmente seria encontrado usando métodos tradicionais devido ao seu suporte muito baixo. O objetivo deste trabalho é justamente encontrar essas sequências de eventos (a proximidade do final da partida de futebol gera congestionamento), o conjunto de posições (ruas próximas ao estádio do Maracanã) e o intervalo de tempo (final da partida de futebol) onde esse padrão é frequente. Saber da existência desse padrão é de grande importância para o planejamento e gestão da cidade.

Uma solução intuitiva para encontrar padrões que são restritos no espaço e no tempo é buscar sequências de eventos frequentes com suporte muito baixo, encontrar suas ocorrências e agrupar essas ocorrências para cada sequência frequente, obtendo grupos restritos no espaço e no tempo onde essas sequências têm suporte alto. Esta solução é inspirada em Alatrista-Salas et al. [2015] e uma possível implementação dela é combinar algoritmos bem conhecidos para mineração de sequência e agrupamento. Contudo, tal solução pode gerar muitos itens frequentes devido ao suporte muito baixo usado para encontrar as sequências, impactando no seu desempenho. Neste trabalho, esta abordagem foi utilizada para fins de comparação com o algoritmo desenvolvido e foi nomeada de SPADE+DBSCAN, pois utiliza os algoritmos SPADE (*Sequencial PAttern Discovery using Equivalence classes*) [Zaki, 2001] e DBSCAN (*Density Based Clustering of Applications with Noise*) [Ester et al., 1996].

Campisano et al. [2018] propuseram uma solução para a descoberta de sequências restritas em uma dimensão do espaço e no tempo. O presente trabalho generaliza este problema considerando o espaço de forma tridimensional e apresenta uma solução eficiente com o algoritmo *Generalized Spatial-Time Sequence Miner* (G-STSM). Desta forma, buscam-se não só as sequências que são padrões, mas também o período de tempo e uma região do espaço (tridimensional) onde tais sequências são frequentes. Portanto,

as principais contribuições deste trabalho podem ser resumidas em: *i)* a formalização da generalização do problema para descoberta de sequências restritas no espaço e no tempo, e *ii)* a solução do problema através da proposta do algoritmo G-STSM.

O algoritmo G-STSM foi comparado com a abordagem SPADE+DBSCAN. Todos os experimentos realizados usam um conjunto de dados sísmico espaço-temporal real. A qualidade das abordagens foi avaliada usando métricas de classificação, além disso o desempenho computacional de ambas as abordagens foi comparado. Os resultados obtidos indicam que o G-STSM teve um desempenho melhor do que o SPADE+DBSCAN, com métricas de classificação semelhantes.

Acredita-se que o G-STSM seja uma ferramenta importante para encontrar sequências restritas no espaço e no tempo. Desta forma, foi conduzida uma análise extensa para avaliar sua sensibilidade com o objetivo de entender a influência dos parâmetros de entrada e diferentes tamanhos de conjuntos de dados na saída e no desempenho do algoritmo proposto.

O restante deste trabalho é organizado como segue: O Capítulo 2 descreve os principais conceitos para a compreensão deste trabalho. O Capítulo 3 apresenta levantamento e revisão de artigos relacionados ao tema. O Capítulo 4 traz detalhes sobre a abordagem utilizada neste trabalho apresentando o algoritmo G-STSM. O Capítulo 5 mostra os resultados da aplicação do G-STSM e sua comparação com a abordagem SPADE+DBSCAN. Por fim, o Capítulo 6 discute os resultados obtidos e aponta trabalhos futuros.

2- Referencial Teórico

Este capítulo tem como objetivo apresentar conceitos importantes para compreensão dos assuntos contidos neste trabalho. A Seção 2.1 introduz fundamentos necessários sobre sequências com marcação de tempo e espaço e a Seção 2.2 descreve o processo de mineração de sequências.

2.1- Fundamentos

Uma sequência com marcação de tempo é uma sequência ordenada de observações obtidas por meio de medições repetidas ao longo do tempo [Han et al., 2011]. O estudo de sequências com marcação de tempo é comum em diversas áreas e aplicações, como por exemplo: *i)* na medicina, para avaliação da atividade elétrica do coração, através de um eletrocardiograma, conforme ilustrado na Figura 1; e *ii)* na economia, para avaliar as relações comerciais e financeiras entre dois países, a taxa de câmbio, conforme ilustrado na Figura 2 [Mooney and Roddick, 2013; Han et al., 2007; Chen and Hu, 2006; Klemettinen et al., 1994].

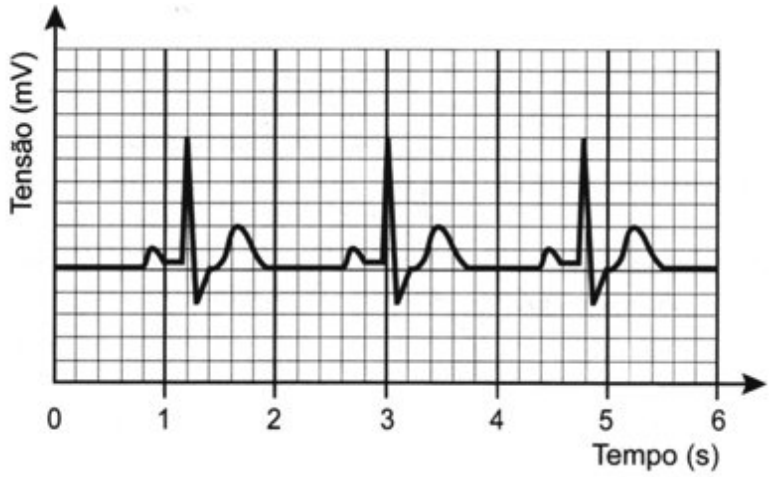


Figura 1 – Exemplo de sequência com marcações de tempo usada na medicina: eletrocardiograma. Fonte: ENEM 2016.



Figura 2 – Exemplo de sequência com marcações de tempo usada na economia: taxa de câmbio US$ X R$. Fonte: tradingeconomics.com.

Seja $t = <v_1, v_2, \ldots, v_n>$ uma **Sequência com Marcação de Tempo (do inglês *Time-stamped Sequence*) (TS)**, onde v_i é um item, $|t| = n$ é o número de itens em t, e v_n é o item mais recente em t [Shumway and Stoffer, 2017]. Uma **subsequência** é uma amostra contínua de uma TS com um comprimento definido. Dessa forma, uma subsequência de uma TS t que começa na marcação de tempo p e de tamanho m é uma sequência ordenada de itens representada por: $sub_{m,p}(t) = <v_p, v_{p+1}, \ldots, v_{p+m-1}>$, onde $|sub_{m,p}(t)| = m$ e $1 \leq p \leq |t| - m$.

Uma **sequência** $s = <w_1, w_2, \ldots, w_k>$ está incluída a partir da marcação de tempo inicial q em uma TS $t = <v_1, v_2, \ldots, v_n>$, se existir uma posição inicial q tal que $w_1 = v_q, w_2 = v_{q+1}, \ldots, w_k = v_{q+k-1}$. Assim, uma sequência s é definida por: $s = <w_1, w_2, \ldots, w_k>$, $\exists q | s = sub_{k,q}(t)$, onde $|s| = k$.

Diversos tipos de eventos envolvem não só dados temporais, como também dados espaciais, como por exemplo *i)* na sismologia, para levantamentos sísmicos e *ii)* na epidemiologia, para registro do número de infectados por uma dada doença em diferentes regiões ao longo do tempo [Alatrista-Salas et al., 2015]. Uma base de dados espaço-temporal é um conjunto estruturado de informações, em que dimensões espaciais e temporais estão inclusas [Alatrista-Salas et al., 2016]. Dados espaço-temporais podem ser indexados por localizações espaciais e marcações de tempo. O espaço pode ser geográfico ou socioeconômico, e as escalas de tempo podem variar de microssegundos

a milênios. As possibilidades de relacionamentos temporais e espaciais são complexas e geram dificuldades em sua análise e busca por padrões [Han et al., 2007].

Uma **posição espacial** (por simplicidade, **posição**) p é definida como um trio ordenado (x, y, z), onde x, y e z indicam valores das coordenadas no sistema Cartesiano. Sejam f e h duas posições, tais que $f = (x_f, y_f, z_f)$ e $h = (x_h, y_h, z_h)$. A distância entre f e h, denotada por $dist(f, h)$, é calculada usando a distância euclideana: $dist(f, h) = \sqrt{(x_h - x_f)^2 + (y_h - y_f)^2 + (z_h - z_f)^2}$.

Seja $P = \{p_1, p_2, ..., p_m\}$ um conjunto de posições, uma **Sequência com Marcação de Tempo e Espaço (do inglês *Spatial Time-stamped Sequence*) (STS)** st é uma dupla (p, t), onde $p \in P$ é uma posição e t é a TS associada. Desta forma, um conjunto de dados de STS D é um conjunto de STS. Diz-se que uma STS $st = (p, t)$ suporta uma sequência s, se s é uma subsequência em t: $sup(s, st) = |Q|, \forall q \in Q | s = sub_{|s|,q}(st.t)$. O **suporte** de uma sequência s em D é o número de marcações de tempo em D em que s está incluído, denotado por: $sup(s, D) = |Q|, \forall q \in Q, \exists st_i \in D | s = sub_{|s|,q}(st_i.t)$, onde Q é o conjunto de marcações de tempo da sequência s em D.

A frequência de uma sequência s em uma STS st é a fração de $st.t$ que apresenta suporte s: $freq(s, st) = \frac{sup(s,st)}{|st.t|}$. Saleh and Masseglia [2008] definem a frequência de um conjunto de itens aplicada sobre uma base de dados como sendo o número de transações que apresentam a ocorrência dos itens dividido pelo tamanho total do conjunto de dados. Desta forma, a **frequência** de uma sequência s em D é a fração de tempo em D que suporta s, representada por: $freq(s, D) = \frac{sup(s,D)}{|st.t|}$, $st \in D$, assumindo que $|st.t|$ é o mesmo em todas as STS. Dado um valor mínimo definido pelo usuário $\gamma \in]0, 1]$, uma sequência é dita frequente, se $freq(s, D) \geq \gamma$.

Um **período de tempo** (por simplicidade, **período**) $r = (r_s, r_e)$ é definido por uma marcação de tempo inicial r_s e uma marcação de tempo final r_e. O tamanho do período r é dado por: $|r| = r_e - r_s + 1$. PR é o conjunto de todas os possíveis períodos de tempo sobre o conjunto de dados D.

2.2- Mineração de Sequências

A mineração de dados é um processo de descoberta de padrões significativos em um conjunto de dados. A área de conhecimento de mineração de sequências é uma especialização da mineração de dados, focada em encontrar sequências ou séries de eventos em bases de dados, os quais ocorrem formando algum tipo de padrão, um conjunto de atributos que aparecem persistentemente em meio ao conjunto de dados [Aydin and Angryk, 2016]. Para efetuar tal tarefa são utilizados conceitos de diferentes áreas de conhecimento, como a estatística, aprendizado de máquina, reconhecimento de padrões, inteligência artificial, dentre outras [Hand, 2007; Witten et al., 2016; Roiger, 2017; Klemettinen et al., 1994; Aydin and Angryk, 2016].

O conceito de mineração de sequências, primeiramente abordado por Agrawal and Srikant [1995], evoluiu e apresenta novas definições. Tsai and Shieh [2009] descrevem mineração de sequências como a técnica que explora padrões frequentes que ocorrem relacionados ao tempo extraídos de uma base de dados. O objetivo da mineração de sequências é o de ser capaz de observar um conjunto de subsequências que são frequentes em um conjunto de dados. Isso significa que sua frequência excede um valor mínimo definido pelo usuário.

Um dos mais conhecidos algoritmos de busca de padrões frequentes é o Apriori. A ideia chave por trás dos algoritmos Apriori é que, como o próprio nome diz, ele utiliza conhecimentos prévios para realizar a busca por padrões frequentes. Sequências candidatas de tamanho $k+1$ são baseadas em combinações de sequências frequentes já descobertas de tamanho k. Esta fase da mineração de sequências é conhecida como geração de candidatos, a qual leva em conta o conceito antimonotônico, o qual frequentemente é citado, no âmbito de mineração de sequências, como: para que uma sequência seja frequente suas subsequências tem de ser frequentes. Exemplificando, na busca de padrões frequentes tal conhecimento é usado da seguinte forma: a existência das sequências de tamanho dois AB e BC permite a geração de um candidato de tamanho três ABC [Mooney and Roddick, 2013].

O algoritmo SPADE é um dos algoritmos baseados na técnica Apriori. Ele utiliza bases de dados em um formato vertical baseada em identificadores e usa técnicas de busca baseadas em rede (busca em largura e busca em profundidade). SPADE aplica

propriedades combinatórias para decompor o espaço de busca em sub-redes que podem ser processadas independentemente na memória principal, permitindo assim que o banco de dados seja verificado até três vezes, o que minimiza E/S e custos computacionais. Este algoritmo permite também a adição de restrições as buscas de sequências [Zaki, 2001].

cSPADE é uma extensão do SPADE que adiciona restrições nas buscas de sequências frequentes, as seguintes foram adicionadas: comprimento ou largura nas sequências, intervalo mínimo ou máximo em elementos de sequência consecutivos, janela de tempo total de validade da sequência, restrições de item e enumerar sequências preditivas de uma determinada classe entre um conjunto de valores de classe Zaki [2000].

Um problema relacionado a abordagem Apriori é o grande número de candidatos gerados. O paradigma de crescimento de padrão frequente (*pattern growth*) remove a necessidade de geração de candidatos, adotando uma abordagem de divisão e conquista que usa projeções do conjunto de dados. Em lugar de efetuar buscas por todo o conjunto de dados com todo o conjunto de candidatos, este paradigma divide o conjunto de dados e também as sequências a serem verificadas, o que pode resultar em melhor desempenho em grandes conjuntos de dados [Han et al., 2000; Mooney and Roddick, 2013].

A mineração de sequências relacionadas a espaço e tempo é a busca por conhecimentos relacionados aos fenômenos que envolvem tanto componentes espaciais como temporais, tentando encontrar todas as sequências de eventos significantes, úteis, interessantes e não triviais [Aydin and Angryk, 2016; Sunitha and Rama Mohan Reddy, 2014; Alatrista-Salas et al., 2015; Huang et al., 2008].

Um padrão espaço-temporal trata de uma sequência de eventos as quais são restritas a uma região e a um período de tempo. É importante observar que estas sequências restritas no espaço e no tempo podem apresentar baixo suporte, se considerarmos todo o conjunto de dados, mas se consideradas dentro de um período de tempo e de espaço alcançam valores mais altos [Huang et al., 2008].

Na busca por padrões frequentes, alguns trabalhos utilizam não apenas a mineração de dados, mas também técnicas de agrupamento. Tal abordagem busca agrupar espacialmente ocorrências de eventos próximas utilizando o grupo como uma única região e, dessa forma, restringindo padrões espacialmente. Esta forma de trabalho, assim como outras, podem ser vistas no Capítulo 3, que vem a seguir.

3- Trabalhos Relacionados

Este trabalho de pesquisa foi realizado a partir do desenvolvimento de um mapa sistemático da literatura relacionada ao assunto aqui pesquisado sobre mineração de sequências restritas no espaço e no tempo. Pretende-se com essa revisão sistemática buscar mais informações acerca do assunto, identificar lacunas e comparar o presente trabalho de pesquisa com outros no contexto onde se insere.

Como ponto de partida foi conduzida uma busca por palavras-chave contidas nos campos: título, resumo e palavras-chave de documentos na língua inglesa na base de dados *Scopus*. A seguinte *string* de busca foi utilizada: TITLE-ABS-KEY((“sequence mining” OR “sequential pattern”) AND (“space-time” OR “spatiotemporal”)) AND (LIMIT-TO(LANGUAGE, “English”)). Como resultado da busca, foram obtidas oitenta e três referências a documentos, entre artigos e conferências.

Alguns dos documentos não estavam relacionados ao assunto aqui discutido e dois foram classificados como *survey* sobre do tema de mineração de padrões frequentes: Sunitha and Rama Mohan Reddy [2014] descreveram um *survey* sobre mineração de padrões em bases de dados espaço-temporais. Ele apresenta as principais técnicas para descoberta de três tipos de padrões espaço temporais, de acordo com a ordenação em relação ao tempo: sequenciais (totalmente ordenados), co-ocorrência (desordenados) e em cascata (parcialmente ordenados). Os autores pontuam como trabalhos futuros o uso de medidas de avaliação e técnicas de validação de padrões de real interesse com o intuito de reduzir o número de padrões insignificantes gerados e, assim, a quantidade de memória utilizada na busca por tais padrões. Sukanya and Ranjit Jeba Thangaiah [2019] realizaram uma revisão dos trabalhos relacionados a mineração de padrões frequentes, observando principalmente suas diferentes aplicações, além de abordar possíveis direções futuras.

Os outros sessenta e três artigos foram classificados de acordo com os conjuntos de dados utilizados em suas pesquisas e serão discutidos nas seções que se seguem. Nem todos os artigos são discutidos, foram escolhidos para leitura completa apenas os que, após rápida leitura do título e do resumo, mais significativos e/ou apresentaram relação direta ao trabalho aqui descrito. A Seção 3.1 aborda nove dos quarenta trabalhos

que utilizam conjuntos de dados de trajetórias, os quais buscam padrões referentes ao movimento. No contexto de mineração de sequências, a trajetória de um objeto é uma sequência de locais com marcação de tempo. Portanto, uma base de dados de trajetória contém registros de posições no espaço com sua respectiva marcação de tempo [Giannotti et al., 2007].

A Seção 3.2 aborda dezessete dos vinte e três trabalhos que utilizam conjuntos de dados de posições fixas, os quais buscam padrões referentes às sequências de eventos. Em tais conjuntos de dados, para diferentes posições no espaço, registram-se os eventos que se dão em um dado período de tempo. Por fim, a Seção 3.3 traz uma comparação entre as abordagens encontradas na literatura relacionadas ao presente trabalho.

3.1- Bases de Dados de Trajetória

Os trabalhos referentes aos conjuntos de dados de trajetória descrevem uma coleção de eventos do mesmo objeto, ou fenômeno natural, movendo-se em diferentes marcações de tempo e espaço. A coleção de eventos pode estar relacionada aos movimentos de grupos [Feuerhake and Sester, 2013], às atividades humanas de deslocamento [Li and Fu, 2014; Chen et al., 2014; Lee et al., 2016; Xu and Kwan, 2020], às regiões baseadas em polígonos que se movem [Aydin and Angryk, 2016; Aydin et al., 2020], às postagens em uma rede social [Huang et al., 2016], aos fenômenos geográficos [He et al., 2020], ou mesmo às corridas de táxi [Yang and Gidófalvi, 2018; Ibrahim and Shafiq, 2019; Cheng et al., 2020].

Feuerhake and Sester [2013] buscaram padrões de movimento em grupo, como por exemplo, padrões de formações de ataque ou defesa em uma partida de futebol. A abordagem consiste em dois passos. Primeiro, a clusterização é aplicada sobre diferentes marcações de tempo nos dados, permitindo que elementos, que possuem suas posições descritas uns em relação aos outros, sejam descobertos. Neste passo, as possíveis rotações, translações e mudanças de escala são consideradas. Como segundo passo, eles buscaram no conjunto de dados por sequências dos grupos.

Chen et al. [2014] utilizaram mineração de padrões sequenciais com o intuito de extrair sequências de lugares frequentemente visitados e usá-los para modelar um perfil

de mobilidade. Isso possibilita, então, comparar diferentes perfis e calcular a similaridade entre eles. A construção do perfil de mobilidade segue quatro passos: *i*) marcar pontos de parada de cada trajetória; *ii*) aplicar um algoritmo de agrupamento nos pontos para gerar rotas de interesse; *iii*) transformar as trajetórias GPS em trajetórias com rotas de interesse; e *iv*) minerar padrões de trajetória frequentes.

Li and Fu [2014] e Lee et al. [2016] buscaram padrões frequentes para predizer atividades humanas de deslocamento. No intuito de predizer próximos passos de atividades em andamento, eles compararam padrões encontrados nas sequências modeladas com as em atividades andamento.

Aydin and Angryk [2016] propuseram dois algoritmos para mineração de sequências espaço-temporais de regiões em movimento. Este trabalho foi estendido em Aydin et al. [2020], no qual um novo algoritmo foi introduzido. O intuito é minerar sequências de eventos espaço-temporais em uma base de dados de trajetórias de regiões em movimento, sem informação de valores limites definidos pelo usuário. Tal algoritmo repete aleatoriamente o processo de mineração em um subconjunto aleatório de instâncias estimando um índice de participação das sequências de eventos. Ambos os trabalhos restringem tempo e espaço de forma que o movimento entre duas instâncias só é considerado se predicados de continuidade temporal e proximidade espacial forem atendidos.

Huang et al. [2016] buscaram padrões de trajetórias frequentes a partir de dados de mensagens com marcações geoespaciais de uma rede social. Primeiro, um algoritmo de clusterização é aplicado para agrupar em regiões os locais onde o usuário posta suas mensagens. Para encontrar as trajetórias, verifica-se novamente as mensagens (restritas no tempo de um dia) e as regiões onde ocorreram, cada mudança de região se traduz como uma trajetória.

Ibrahim and Shafiq [2019], com o intuito de descobrir padrões de trajetórias frequentes, agruparam pontos de origem e destino de viagens de táxi, identificando distritos (da cidade do Porto, em Portugal) aos quais pertencem. A partir dessas informações, os pontos e horários de maior concentração de corridas foram encontrados. Outra informação obtida foi a de padrões frequentes usando o algoritmo SPADE para mineração de sequências frequentes.

He et al. [2020] analisaram as características de fenômenos geográficos complexos propondo uma nova estrutura espaço-temporal hierárquica complexa para representar rotas espaço-temporais. Uma abordagem de mineração de padrões espaço-temporal

complexa baseada em eventos é proposta neste artigo. Eles aplicaram mineração de sequências para buscar padrões na propagação dos fenômenos desenvolvendo uma nova técnica baseada no algoritmo SPADE.

Embora os trabalhos apresentados na presente seção sejam relevantes, eles buscam padrões diferentes dos explorados neste trabalho. No presente trabalho, não são buscados padrões de trajetórias, mas sim padrões de sequências de eventos, onde todos os eventos de uma mesma sequência ocorrem em uma mesma posição. A seção que se segue aborda trabalhos que utilizam bases de dados de posições fixas.

3.2- Bases de Dados de Posições Fixas

A presente seção aborda trabalhos que utilizam bases de dados de posição fixa, ou seja, dados obtidos do ambiente através do uso de sensores fixos [Batu et al., 2017]. Dessa forma, bases de dados de posição fixa podem consistir do comportamento dos clientes em suas compras [Chen et al., 2020; Koseoglu et al., 2020], de dados hidrológicos [Alatrista-Salas et al., 2015, 2016], de imagens de satélite [Julea et al., 2008, 2011] ou de padrões de crime [Chen et al., 2017].

Tsoukatos and Gunopulos [2001] usaram a abordagem da teoria de rede para decompor o espaço de pesquisa original. Eles efetuaram uma busca em profundidade, que encontra apenas as sequências espaço-temporais de tamanho máximo. Desta forma, a proposta não faz uso de grande quantidade de memória, uma vez que não precisa de todas as sequências de tamanho k para gerar as de tamanho $k + 1$. Este algoritmo não visa reduzir o número de varreduras do conjunto de dados. Ele efetua buscas para averiguar a frequência de um conjunto de sequências e as agrupa em diferentes granularidades no espaço utilizando uma abordagem de pré-processamento para unir sub regiões.

Julea et al. [2008] e Julea et al. [2011] tem, nos seus trabalhos, o intuito de descobrir padrões de sequência através do uso de séries de imagens de satélite. O primeiro trabalho apresenta duas técnicas distintas, utilizando para ambas o algoritmo SPADE na tarefa de mineração de sequências. O primeiro trabalho se baseia em duas técnicas distintas, utilizando para ambas o algoritmo SPADE na tarefa de mineração

de sequências. O segundo trabalho aplica o uso de aprendizado de máquina não supervisionado. Assim, ele apresenta as diferentes técnicas para descrição das imagens obtidas por satélite, e introduz um novo tipo de padrão de mineração de dados dedicado à extração de *pixels* que dividem um padrão temporal e que observem, na média, uma conectividade espacial mínima. A técnica pode ser utilizada em vários tipos de imagens com diferentes resoluções.

Leong and Chan [2012] propuseram um algoritmo que busca por sequências de eventos frequentes por todo um conjunto de dados espaço-temporal, o qual pode ser dividido em cinco fases: *i*) ordenação do conjunto de dados, *ii*) busca por eventos que apresentem suporte maior que um mínimo estabelecido, *iii*) remoção de transações que não apresentem ao menos um item com suporte maior ou igual ao pré definido, *iv*) geração de padrões sequenciais e, por fim, *v*) uma fase que mantém apenas padrões máximos, eliminando padrões que estejam contidos dentro de outros.

Flamand et al. [2014] utilizaram mineração de dados meteorológicos e epidemiológicos para avaliar potenciais causas de surtos de dengue na Guiana Francesa. Este trabalho relacionou os padrões temporais encontrados através do uso do algoritmo PrefixSpan [Pei et al., 2004] com o contexto epidemiológico ou espacial. Os padrões encontrados mostram associações entre condições meteorológicas e a evolução da incidência de dengue.

Gurram and Rama Mohan Reddy [2014] definiram uma estrutura de dados em forma de grafo para representar dados espaço-temporais com o intuito de reduzir o número de buscas na base de dados. Os autores também propuseram um algoritmo para minerar padrões sequenciais, considerando uma dimensão de tempo e uma de espaço. Para isso, eventos são posicionados em uma representação cartesiana, dividindo-a em retângulos, chamados de “células”, as que tiverem uma densidade maior que um valor pré-definido são transformadas em um grafo sequencial em memória. Por fim, define-se uma medida de significância para descobrir padrões úteis a qual leva em conta a densidade das “células”. Um grande problema desta abordagem é que a divisão em retângulos pode remover eventos importantes apenas porque caíram em uma “célula” pouco densa, assim as ligações com estes eventos são perdidas mesmo para células densas próximas a este.

Alatrista-Salas et al. [2015] aplicaram um processo de descoberta de padrões sobre um conjunto de dados hidrológicos com o objetivo de analisar a qualidade da água. O processo consiste em quatro etapas: *i)* agrupar os dados de acordo com a distância;

ii) extrair padrões sequenciais levando em consideração o aspecto temporal; *iii)* filtrar as sequências retendo apenas os que estão de acordo com uma medida de interesse temporal; e *iv)* gerar soluções agrupadas de acordo com a técnica S^2MP [Saneifar et al., 2008] e o algoritmo de clusterização $k - medoids$.

Chen et al. [2015] buscaram por padrões espaço-temporais colocando os dados de entrada (identificação do evento, marcação de tempo, posição e tipo de evento) em uma árvore R. Para montar a representação em árvore R, um nó é utilizado como pivô (elemento inicial da sequência) e os elementos que ocorreram antes do primeiro evento e os que ocorrem depois da marcação de tempo do primeiro mais a restrição de tempo (definido como parâmetro) são removidos. Os elementos que sobram geram uma sequência que se inicia com o pivô. Depois, a restrição espacial para cada sequência, que deve ocorrer dentro de um raio (definido como parâmetro), é verificada. Para cada sequência descoberta, verifica-se seu número de ocorrências, mantendo as que forem frequentes.

Sunitha and Rama Mohan Reddy [2016] propuseram a inclusão de priorização em forma de pesos no processo de mineração utilizando duas medidas de interesse "peso da sequência" e "índice de significância". Os pesos são associados por especialistas no domínio de acordo com a importância dos tipos de evento, regiões geográficas e intervalos de tempo. Como resultado relevante, os autores apontaram a redução do número de padrões descobertos, reduzindo o número de padrões insignificantes.

Alatrista-Salas et al. [2016] definiram duas medidas utilizadas para redução do número de padrões encontrados *absolute support* e *spatiotemporal participation index*. Eles propuseram também dois algoritmos para a mineração de padrões espaço temporais, um baseado em busca em largura com uma estratégia *a priori*, derivado do algoritmo SPADE e o outro baseado em busca em profundidade com uma estratégia *pattern-growth*.

Xue et al. [2016], com o intuito de descobrir padrões frequentes, desenvolveram um método eficaz reduzindo o número de buscas sobre o conjunto de dados e aumentando o desempenho do processo de mineração. O algoritmo proposto é baseado nos conceitos de assimetria de informação, não aditividade (a informação diminui após cada nível em processos de vários níveis) e antimonotonicidade.

Yusof and Zurita-Milla [2017] utilizaram um processo em três etapas com o intuito de encontrar padrões de perfis eólicos: *i)* mineração de padrões (utilizando o algoritmo Linear time Closed Itemset Miner Sequence - LCMSeq), *ii)* detecção de padrões simi-

lares no espaço e no tempo, e *iii)* avaliação da conformidade dos padrões para gerar perfis eólicos. Esta abordagem busca padrões em séries individuais e, depois, encontra interseções dos padrões no espaço e no tempo.

Chen et al. [2017] abordaram o problema de descoberta de padrões frequentes espaço-temporais encadeados sobre bases de dados sem informação de identidade, ou seja, não diz quais ações foram realizadas por quais sujeitos. Os autores propuseram uma abordagem que busca por padrões em um período de tempo e espaço definidas pelo usuário utilizando um modelo baseado em grafos chamado TKSTP.

Batu et al. [2017] propuseram um algoritmo que não depende da entrada de parâmetros definidos pelo usuário para buscar padrões. Tal algoritmo usa um procedimento estocástico e um modelo de Hawkes [Hawkes, 1971] para definir relações entre os tipos de eventos. Os autores aplicaram tal algoritmo sobre dois conjuntos de dados um sintético e um real com dados de acidentes de tráfego.

Zhang et al. [2018] buscaram por padrões em séries temporais medidas sobre diferentes elementos mas que em conjunto descrevem o estado de um sistema. Para isso, eles utilizaram uma abordagem de três estágios: extração de características, descoberta de estados frequentes e síntese de padrões.

Wan et al. [2019] definiram como séries temporais correlacionadas as que são registradas simultaneamente para monitorar e refletir um sistema. Eles utilizaram Redes Neurais Recorrentes (*Recurrent Neural Network*) baseadas no modelo *Long Short Term Memory (LSTM)* para prever futuros valores de séries temporais correlacionadas de uma maneira coletiva.

Chen et al. [2020] investigaram padrões de consumo sobre uma base de dados de registro de compras, usando duas perspectivas. A primeira foi a perspectiva espacial e temporal de maneira agregada, buscando responder a questão de onde e quando os eventos de compra ocorrem mais (conhecidos como *hot spots*). Para isso, foi realizado um agrupamento espaço-temporal utilizando um método denominado ST-DBSCAN [Birant and Kut, 2007] (baseado no DBSCAN), obtendo grupos dos locais e intervalos de tempo onde as compras mais ocorrem. A segunda foi a perspectiva da trajetória de maneira individual para responder a pergunta de qual a sequência de localização das compras realizadas pelos clientes. Para isso, foi utilizada uma técnica híbrida de mineração de padrões sequenciais semânticos combinando o algoritmo PrefixSpan e a análise semântica de *Point of Interest* (POI) [Han et al., 2001]. No entanto, este trabalho considera apenas

sequências dentro de um período de tempo de uma semana.

Koseoglu et al. [2020] propuseram uma abordagem de análise visual, que incorpora a extração de padrão espaço-temporal utilizando um algoritmo de mineração de padrão sequencial estendido e um mecanismo de orientação de descoberta de padrão operando em consulta geográfica. O trabalho aborda passos da concepção da ferramenta de análise visual centrada no usuário utilizada para explicar tendências comportamentais dos grupos de clientes que mudam em relação ao tempo, localização e tipo de cliente.

O presente trabalho faz parte desta classe, também utilizando base de dados de posições fixas. Os pontos que o diferenciam dos outros trabalhos são abordados na seção que se segue.

3.3- Comparação

Na mineração em bases de dados com marcações de espaço e tempo, cada trabalho utiliza métodos diferentes. Alguns utilizam apenas mineração de dados na busca por padrões frequentes, levando em conta somente o tempo [Li and Fu, 2014; Lee et al., 2016; He et al., 2020; Tsoukatos and Gunopulos, 2001; Julea et al., 2008; Leong and Chan, 2012; Xue et al., 2016; Zhang et al., 2018], ou introduzindo alguma restrição prévia para também levar em conta o espaço [Aydin and Angryk, 2016; Aydin et al., 2020; Julea et al., 2011; Flamand et al., 2014; Gurram and Rama Mohan Reddy, 2014; Chen et al., 2015; Sunitha and Rama Mohan Reddy, 2016; Chen et al., 2017; Batu et al., 2017; Koseoglu et al., 2020]. Outros utilizam a mineração de dados para a busca de padrões frequentes no tempo e também agrupamento com o intuito de agrupar posições espaciais [Feuerhake and Sester, 2013; Chen et al., 2014; Huang et al., 2016; Yusof and Zurita-Milla, 2017; Ibrahim and Shafiq, 2019; Alatrista-Salas et al., 2015, 2016; Chen et al., 2020]. O presente trabalho efetua uma abordagem diferente. A mineração de dados é realizada, buscando sequências frequentes no tempo, mas tais sequências frequentes devem ocorrer em grupos espaciais de forma que tais grupos possuam um certo número mínimo de posições espaciais (definido pelo usuário), e cada posição espacial deve estar a uma distância máxima (também definida pelo usuário) em relação a uma outra posição do mesmo grupo.

Apesar deste trabalho fazer parte da classe de posições fixas, cada trabalho lida com restrições de maneiras diferentes. Alguns usam suporte global (um valor de suporte que é válido para todo o conjunto de dados) [Tsoukatos and Gunopulos, 2001; Leong and Chan, 2012; Xue et al., 2016; Yusof and Zurita-Milla, 2017; Batu et al., 2017; Zhang et al., 2018]. Outros consideram suporte local, aplicando restrições pré-definidas de tempo, espaço ou ambos. Julea et al. [2011] limitaram o espaço em número de pontos de uma imagem. Flamand et al. [2014] restringiram o espaço em territórios geográficos e o tempo em intervalos. Gurram and Rama Mohan Reddy [2014] mapearam os eventos do conjunto de dados em células em forma de uma grade de acordo com o tempo e espaço, essas células têm seu tamanho definido pelo usuário limitando o tempo e o espaço. Alatrista-Salas et al. [2015] pré-processaram os dados construindo zonas homogêneas de objetos espaciais. Chen et al. [2015] limitaram o espaço em raios e o tempo em período de tempos. Sunitha and Rama Mohan Reddy [2016] restringiram o espaço em sub-regiões e o tempo em intervalos. Alatrista-Salas et al. [2016] limitaram o espaço em zonas. Chen et al. [2017] pré-definiram um período de tempo e um raio para o espaço. Chen et al. [2020] consideraram o tempo dentro de uma semana para evitar grandes intervalos entre eventos. Koseoglu et al. [2020] permitiram o usuário limitar tanto o tempo quanto o espaço.

O presente trabalho lida com tais restrições de uma maneira diferente, pois não considera inicialmente limitações de tempo ou espaço. Ele encontra as sequências frequentes, a região no espaço e o período de tempo em que são frequentes. Desta forma, o algoritmo a ser apresentado neste trabalho é capaz de encontrar diferentes tamanhos de sequências, intervalos de tempo e regiões do espaço onde uma sequência é frequente.

A Tabela 1 mostra a classificação dos trabalhos relacionados de acordo com a base de dados (trajetória ou posição fixa), o método utilizado (mineração de sequências ou agrupamento com mineração de sequências), e as restrições aplicadas sobre o método, que podem ser suporte global ou suporte local com espaço fixo e /ou tempo fixo.

Como pode ser observado, o único trabalho com abordagem semelhante encontrado na literatura é o proposto por Campisano et al. [2018], o qual busca por sequências frequentes apenas em intervalos de tempo e regiões no espaço, começando a buscar sequências primeiro no espaço e depois no tempo. Dessa forma, ele considera o espaço de forma linear. O presente trabalho estende-o com uma nova perspectiva sobre o

Tabela 1 – Classificação dos trabalhos relacionados apresentando informações do método utilizado e das restrições aplicadas.

Classe	Trabalho	Método		Restrições		
		Min.	Agrup. + Min.	Sup. Global	Espaço Fixo	Tempo Fixo
Trajetória	Feuerhake and Sester [2013]		X	X		
	Chen et al. [2014]		X		X	X
	Li and Fu [2014]	X		X		
	Aydin and Angryk [2016]	X			X	X
	Huang et al. [2016]		X		X	X
	Lee et al. [2016]	X		X		
	Ibrahim and Shafiq [2019]		X	X		
	Aydin et al. [2020]	X			X	X
	He et al. [2020]	X		X		
Pos. Fixa	Tsoukatos and Gunopulos [2001]	X		X		
	Julea et al. [2008]	X		X		
	Julea et al. [2011]	X			X	
	Leong and Chan [2012]	X		X		
	Flamand et al. [2014]	X			X	X
	Gurram and Rama Mohan Reddy [2014]	X			X	X
	Alatrista-Salas et al. [2015]		X		X	
	Chen et al. [2015]	X			X	X
	Sunitha and Rama Mohan Reddy [2016]	X			X	X
	Alatrista-Salas et al. [2016]		X		X	
	Xue et al. [2016]	X		X		
	Yusof and Zurita-Milla [2017]		X	X		
	Chen et al. [2017]	X			X	X
	Batu et al. [2017]	X		X		
	Campisano et al. [2018]	X				
	Zhang et al. [2018]	X		X		
	Chen et al. [2020]		X			X
	Koseoglu et al. [2020]	X			X	X

problema, passando a buscar as sequências primeiro no tempo e depois no espaço. Usando tal abordagem foi possível generalizá-lo considerando o espaço de maneira tridimensional.

Até onde alcançaram as buscas aqui apresentadas, a abordagem proposta neste trabalho é a primeira capaz de encontrar sequências restritas no espaço e no tempo que funcionam com uma dimensão de tempo e três dimensões de espaço. Assim, para efeito de comparação, também desenvolvemos a abordagem SPADE+DBSCAN baseada em Alatrista-Salas et al. [2015]. Tal trabalho agrupou dados de acordo com a distância e extraiu padrões sequenciais para levar em consideração o aspecto temporal.

SPADE+DBSCAN combina um algoritmo de mineração de sequências com um método de agrupamento capaz de encontrar grupos que contenham sequências frequentes em três etapas: *i)* encontrar todas as sequências frequentes, usando o algoritmo SPADE; *ii)* para cada sequência frequente, encontrar todas as suas ocorrências no conjunto de dados; e *iii)* agrupar as ocorrências de cada sequência frequente, utilizando o

algoritmo de agrupamento DBSCAN baseado na noção de densidade. Assim, temos sequências frequentes e seus grupos que são restritos no espaço e no tempo.

A abordagem SPADE+DBSCAN servirá de *baseline* para comparação com o algoritmo G-STSM. Esta comparação será apresentada no Capítulo 5, e neste mesmo capítulo, na Seção 5.3, mais detalhes acerca da abordagem SPADE+DBSCAN são apresentados.

4- Metodologia

Este capítulo apresenta a solução proposta para o problema de Mineração de Sequências Restritas no Espaço e no Tempo, na forma do algoritmo G-STSM. Considerando um conjunto de dados STS D, o problema abordado neste trabalho é encontrar sequências em D que são frequentes em posições espaciais e período de tempo restritos. O objetivo é descobrir sequências frequentes, o período de tempo e o conjunto de posições em que essas sequências são frequentes.

A Seção 4.1 apresenta a formalização do problema que este trabalho pretende resolver, trazendo conceitos fundamentais para este trabalho. Na Seção 4.2 os algoritmos desenvolvidos são apresentados. Por fim, na Seção 4.3 é apresentado um exemplo utilizando um conjunto de dados sintético.

4.1- Formalização do Problema

Um **grupo de posições espaciais** (por simplicidade **grupo**) g é definido por um conjunto de posições onde seus elementos devem estar a uma distancia máxima σ de ao menos um outro elemento do grupo, ou seja: $g|\forall p \in g, \exists q \in g|dist(p,q) \leq \sigma$. P é o conjunto de todas as posições. PG é o conjunto de todos os possíveis grupos de posições sobre o conjunto de dados D. O conjunto de STS de um grupo g é definido por: $sts(g) = SG|\forall st \in SG, st.p \in g$.

Um ***Ranged Group*** **(RG)** rg é um trio (s, r, g), onde s é uma sequência, r é um período de tempo e g é um grupo. As ocorrências de uma sequência s em um RG rg, definido por $occur(s, r, g)$, referem-se ao número de todas as ocorrências de s no intervalo r em $sts(g)$. O suporte de uma sequência s em um RG rg, denotado por $sup(s, r, g)$, é o número de marcações de tempo que s começa no intervalo r em $sts(g)$, ou seja: $sup(s, r, g) = |Q|, \forall q \in Q, \exists st \in sts(g)|s = sub_{|s|,q}(st.t), r_s \leq q \leq r_e, |s| \leq r_e$. A frequência de uma sequência s em um RG rg, $freq(s, r, g)$, é a divisão do suporte do RG $sup(s, r, g)$ pelo tamanho de r: $freq(s, r, g) = \frac{sup(s,r,g)}{|r|}$.

Dados os limites mínimos definidos pelo usuário para frequência γ e para tamanho do grupo β, as características de um ***Kernel Range-Group*** **(KRG)** e de um ***Solid Range-Group*** **(SRG)** são apresentadas na Definição 1 e na Definição 2, respectivamente.

Definição 1 *Seja rg um RG com sequência s, período de tempo r, e grupo g. Então, rg é chamado de KRG, se e somente se, as seguintes condições forem verdadeiras:*
1) $freq(s, r, g) \geq \gamma$
2) $|g| \geq \beta$
3) $\forall r' \in PR | r' \subset r$ *e* $r'.r_s = r.r_s$*, ambas as condições se aplicam:*
a) $sup(s, r', g) < sup(s, r, g)$
b) $freq(s, r', g) \geq \gamma$
4) $\forall g' \in PG | g \subseteq g', occur(s, r, g') = occur(s, r, g)$
5) $\forall g' \in PG | g' \subset g, occur(s, r, g') < occur(s, r, g)$

A primeira condição certifica que a frequência de uma sequência s em um período de tempo r sobre as STS de um grupo g (*i.e.* $sts(g)$) pertencente ao RG rg é maior que uma frequência mínima γ definida pelo usuário. A segunda condição certifica que o grupo g deve respeitar o tamanho mínimo β definido pelo usuário.

A terceira condição certifica que para todo período de tempo r' pertencente ao conjunto de todos os possíveis intervalos de tempo PR, contido no período de tempo r e começando na mesma marcação de tempo, as seguintes condições devem ser verdadeiras: (a) o suporte em r' será menor e (b) a frequência será maior que a frequência mínima definida pelo usuário γ. Em outras palavras, diminuir o período de tempo mantém uma frequência maior que o mínimo, mas diminui o suporte. Pode-se dizer que a condição garante que o tamanho de r é mínimo entre os intervalos que começam na mesma marcação de tempo.

A quarta condição certifica que para todo o grupo g' pertencente ao conjunto dos possíveis grupos PG, tal que g está contido ou é igual a g', a ocorrência da sequência s no período de tempo r será igual em ambos os grupos. Em outras palavras, aumentar o grupo mantém o mesmo número de ocorrências (*i.e.*, não vale a pena aumentar o grupo de g para g', visto que o aumento não vai contribuir para o número de ocorrências de s). Pode-se dizer que a condição garante que o tamanho de g é máximo.

Por fim, a quinta condição certifica que para todo grupo g' pertencente ao conjunto

de todos possíveis grupos PG, tal que g' está contido em g, a ocorrência da sequência s no período de tempo r diminuirá no grupo g'. Em outras palavras, diminuir o grupo reduz o número de ocorrências (*i.e.*, não vale a pena diminuir o grupo de g para g', pois isso também reduz o número de ocorrências de s). Pode-se dizer que a condição garante que o tamanho de g é mínimo.

Definição 2 *Seja rg um RG com uma sequência s, período de tempo r, e grupo g. Então, rg é chamado de SRG se e somente se as condições que se seguem se aplicarem:*

1) $freq(s, r, g) \geq \gamma$

2) $|g| \geq \beta$

3) $\forall r' \in PR | r \subseteq r'$*, é possível ter a) ou b) ou ambas:*

a) $sup(s, r', g) = sup(s, r, g)$

b) $freq(s, r', g) < \gamma$

4) $\forall r' \in PR | r' \subset r, sup(s, r', g) < sup(s, r, g)$

5) $\forall g' \in PG | g \subseteq g', occur(s, r, g') = occur(s, r, g)$

6) $\forall g' \in PG | g' \subset g, occur(s, r, g') < occur(s, r, g)$

A primeira condição certifica que a frequência de uma sequência s em um período de tempo r sobre as STS de um grupo g (*i.e.* $sts(g)$) pertencente ao RG rg é maior que uma frequência mínima γ definida pelo usuário . A segunda condição certifica que o grupo deve respeitar o tamanho mínimo β definido pelo usuário .

A terceira condição certifica que para todo período de tempo r' pertencente ao conjunto de todos os possíveis períodos de tempo PR, que contém o período de tempo r, as afirmações (a) ou (b), ou ambas devem ser verdadeiras: (a) o suporte no período de tempo r' será igual ao suporte no período de tempo r; (b) a frequência em r' será menor que a frequência definida pelo usuário. Assim, não adianta aumentar o período de tempo, pois mantemos o suporte, e podemos acabar reduzindo a frequência a um valor menor que a mínima definida pelo usuário. Pode-se dizer que a condição garante que o período de tempo r é máximo.

A quarta condição certifica que para todo período de tempo r' pertencente ao conjunto de todos os períodos de tempo PR, tal que r' está contido em r, o suporte para o período de tempo r' será menor que para r. De fato, s apresenta suporte na primeira e na última marcação de tempo no período de tempo r, então se um período de tempo menor

existe onde s é frequente, o suporte será menor. Em outras palavras, diminuir o período de tempo diminui o suporte. Pode-se dizer que a condição garante que o tamanho do período de tempo r é mínimo.

A quinta condição certifica que para todo grupo g' que pertence ao conjunto de todos possíveis grupos PG, tal que g está contido em g', a ocorrência da sequência s será igual para o grupo g e para o grupo g' no mesmo período de tempo r. Ou seja, aumentar o grupo mantém o número de ocorrências da sequência. Pode-se dizer que a condição garante que o tamanho do grupo g é máximo.

Finalmente, a sexta condição certifica que para todo grupo g' que pertence ao conjunto de todos possíveis grupos PG, tal que g' está contido em g, a ocorrência da sequência s será menor que para o grupo g no mesmo período de tempo r. Em outras palavras, diminuir o grupo diminui o número de ocorrências da sequência. Pode-se dizer a condição garante que o tamanho do grupo g é mínimo.

Seja k o tamanho da sequência s, então srg é um SRG de tamanho k. $\mathbf{SRG_k}$ é o conjunto de todos SRG de tamanho k.

Como foi dito no início deste capítulo, o problema que se pretende resolver é o de Mineração de Sequências Restritas no Espaço e no Tempo, ou seja, encontrar o período de tempo e o conjunto de posições (*i. e., SRG*) onde as sequências são frequentes. Portanto, usando as definições descritas nesta seção, o objetivo deste trabalho é encontrar todos os SRGs que respeitem a Definição 2.

A Seção 4.2, que se segue, apresenta o algoritmo que utiliza os conhecimentos aqui abordados e é proposto por este trabalho como solução para o processo de Mineração de Sequências Restritas no Espaço e no Tempo.

4.2- Algoritmo G-STSM

Esta seção apresenta os algoritmos que fazem parte de cada passo no processo de Mineração de Sequências Restritas no Espaço e no Tempo na forma do algoritmo proposto, o G-STSM. Tal algoritmo é projetado para a identificação de sequências frequentes em conjuntos de dados espaço-temporais a partir do conceito de SRG. As noções de grupo (RG, KRG e SRG) introduzidas na seção anterior permitem a extração de

sequências restritas no espaço e no tempo de forma eficiente.

O G-STSM é baseado no princípio de geração de candidatos. Nosso objetivo é começar a encontrar SRG para sequências de tamanho um e explorar o suporte e o número de ocorrências de SRG para sequências maiores com um número limitado de varreduras no conjunto de dados. Para isso, precisamos encontrar o período de tempo e o conjunto de posições (*i. e.*, o SRG) em que uma sequência candidata é frequente em apenas uma varredura.

Seja $c \in C_k$ um candidato de tamanho k no conjunto de candidatos C_k. Na estrutura de dados definida, c está associado a uma sequência $c.seq$, ao período de tempo $c.range$, ao conjunto de posições espaciais $c.pos$, e ao conjunto de RGs $c.rgs$ da sequência $c.seq$ sobre o período de tempo $c.range$ que ocorre nas STS cujo conjunto de posições são $c.pos$. Um RG rg está associado a uma sequência $rg.s$, um período de tempo $rg.r$ e um grupo de posições espaciais $rg.g$. Além disso, $rg.freq$ é a frequência da sequência $rg.s$ em rg e $rg.occ$ é um conjunto de ocorrências de $rg.s$ em rg. Por fim, um valor lógico ($rg.closed$) permite saber o estado do RG. Um valor *verdadeiro* significa que, em algum momento, durante a varredura do conjunto de dados ao tentar estender seu período de tempo, sua frequência se tornou menor que a frequência mínima definida pelo usuário γ.

Um KRG krg possui todas as informações de um RG, onde sua frequência é maior ou igual a γ, o tamanho de seu grupo é maior ou igual a β e nele suas posições estão a uma distância máxima σ de pelo menos uma outra posição do mesmo grupo. Da mesma forma, um SRG está associado às mesmas informações de um RG. A Figura 3 detalha, através do uso de um diagrama UML, a estrutura de dados referente aos candidatos e seu relacionamento com as outras estruturas de dados utilizadas neste trabalho, os RGs, KRG e SRG.

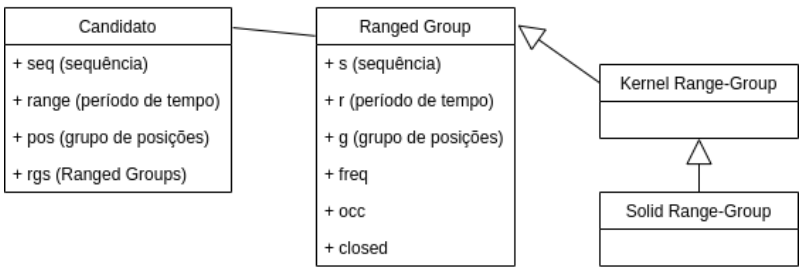


Figura 3 – Diagrama UML referente as estruturas de dados utilizadas. Um candidato c possui um conjunto de Ranged Groups $c.rgs$. Ranged Group generaliza Kernel Range-Group, que por sua vez generaliza Solid Range-Group.

Na Subseção 4.2.1, a seguir, apresentamos o algoritmo G-STSM além dos detalhes de seu funcionamento.

4.2.1 Princípio Geral

O Algoritmo 1 é o ponto de entrada do processo proposto neste trabalho. Ele recebe como entrada um conjunto de dados STS D, um conjunto de itens I, um conjunto de posições espaciais P referentes às STS e os limites definidos pelo usuário: a frequência mínima γ, o tamanho mínimo de um grupo β e a distancia máxima de ao menos um outro elemento do grupo σ. O algoritmo possui três funções principais para: (*i*) encontrar os KRGs através do uso da função $FindKernelRangeGroup$, detalhada na Subseção 4.2.2, (*ii*) unir KRGs para identificar SRGs através do uso da função $MergeKernelRangeGroups$, detalhada na Subseção 4.2.3 e (*iii*) gerar candidatos através do uso da função $GenerateCandidates$, detalhada na Subseção 4.2.4.

O algoritmo inicia sua execução gerando sequências candidatas de tamanho um. Elas são construídas a partir de todos os itens distintos de I apresentados no conjunto de dados STS D, considerando todo o seu período de tempo e todas as posições espaciais P (linhas 3 a 5). Em seguida, uma repetição (linhas 6 a 25) calcula, para cada rodada k, onde k é equivalente ao tamanho da sequência buscada, todos os SRG_k com uma frequência maior ou igual a γ, tamanho do grupo maior ou igual a β, e as posições do grupo a uma distância máxima σ de pelo menos uma outra posição do mesmo grupo.

A busca sobre o conjunto de dados STS D é realizada a partir de uma "janela deslizante" de comprimento k (linha 9) igual ao tamanho da sequência para encontrar os KRGs de cada candidato $c \in C_k$ na função $FindKernelRangeGroup$ (linhas 10 a 14). O motivo do uso de uma janela deslizante de tamanho k, tamanho igual ao da sequência, é que toda a sequência buscada possa ser verificada. Exemplificando, se buscamos uma sequência de tamanho três, digamos "ACE", então são necessárias três marcações de tempo para verificar se a sequência completa está presente, dessa forma, a janela deslizante possui sempre o tamanho da sequência para a qual buscamos os KRGs.

Após a execução do algoritmo $FindKernelRangeGroup$ alguns KRGs podem acabar com seu atributo $closed$ com valor *falso* e sem a devida validação. Este trabalho é

feito através da chamada da função $ValidateAndClose$ para cada candidato (linhas 15 a 17).

A seguir, para cada candidato (linhas 18 a 23), quando possível, os KRGs encontrados são mesclados na função $MergeKernelRangeGroups$ (linha 19) e, a partir dos KRGs mesclados, o conjunto de SRGs SRG_k é gerado (linhas 20 a 22). Finalmente, as sequências candidatas de tamanho $k+1$ são geradas a partir da combinação de SRG_k na função $GenerateCandidates$ (linha 24). A repetição é interrompida quando não obtemos sequências candidatas de tamanho $k+1$ (linha 25). O algoritmo fornece como saída todos os SRGs encontrados.

Algoritmo 1 – G-STSM

```
   Input: D, I, P, γ, β, σ
 1 begin
 2     k ← 0
 3     foreach (i ∈ I) do
 4         C_1 ← C_1 + (i, [D_s..D_e], [P_1..P_n], ∅)
 5     end
 6     repeat
 7         k++
 8         SRG_k ← ∅
 9         SW ← slidingWindows([D_s..D_e], k)
10         foreach (d ∈ SW) do
11             foreach (c ∈ C_k / d.timestamp ∈ c.range) do
12                 FindKernelRangeGroup(c, d, γ, β, σ)
13             end
14         end
15         foreach (c ∈ C_k) do
16             c.rgs ← ValidateAndClose(c.rgs, β)
17         end
18         foreach (c ∈ C_k) do
19             c.rgs ← MergeKernelRangeGroups(c.rgs, γ)
20             foreach (rg ∈ c.rgs) do
21                 SRG_k ← SRG_k + (rg)
22             end
23         end
24         C_{k+1} ← GenerateCandidates(SRG_k)
25     until (C_{k+1} ≠ ∅);
26 end
   Output: SRG
```

As subseções que se seguem detalham os algoritmos das funções chamadas a partir do algoritmo principal do G-STSM. A Subseção 4.2.2 explica em detalhes a função $FindKernelRangeGroup$ e suas funções auxiliares, além de apresentar a função

$ValidateAndClose$. A Subseção 4.2.3 apresenta a função $MergeKernelRangeGroups$. Por fim, a Subseção 4.2.4 detalha a função $GenerateCandidates$.

4.2.2 Seleção dos Kernel Range-Groups

A função $FindKernelRangeGroup$ é a primeira a ser chamada pelo algoritmo principal, ela é apresentada no Algoritmo 2. Seu objetivo é encontrar as informações sobre os KRGs de uma sequência candidata. Ela recebe como entrada a sequência candidata c, a janela deslisante atual d de tamanho k (igual ao tamanho da sequência candidata buscada) na qual será verificada a existência dos KRGs, e os limites definidos pelo usuário γ, β e σ.

Se a sequência candidata estiver contida na janela deslisante recebida como entrada (linhas 2 a 5), novos RGs são criados com o uso da função $SplitGroups$ (linha 3), tendo um período de tempo que inicia e termina na marcação de tempo do início da janela deslisante atual d, e um grupo que contém as posições onde a sequência ocorre em d. Assim, RGs são gerados com o estado aberto (*i.e.*, $closed = falso$). Em seguida (linha 4), os RGs abertos (recém gerados e preexistentes) são unidos respeitando a frequência mínima γ e o tamanho mínimo do grupo β usando a função $MergeOpenKernelRangeGroups$. Como resultado, os RGs da sequência candidata de entrada são atualizados. Vale ressaltar que se um RG rg tem seu atributo $rg.closed$ com valor *falso*, isso significa que ele ainda pode ser seu período de tempo estendido.

Algoritmo 2 – FindKernelRangeGroup

```
   Input: c, d, γ, β, σ
1  begin
2  |  if (c.seq ⊆ d) then
3  |  |  c.rgs ← c.rgs + SplitGroups(c, d, σ)
4  |  |  c.rgs ← MergeOpenKernelRangeGroups(c.rgs, d.timestamp, γ, β, σ)
5  |  end
6  end
   Output: Atualiza o(s) RG(s) de um candidato c
```

A função $SplitGroups$ é apresentada no Algoritmo 3. Seu objetivo é criar novos RGs para grupos de posições da transação atual d que contenham a sequência candidata

procurada. Ela recebe como entrada um candidato c, a transação atual d e a frequência mínima definida pelo usuário σ. O algoritmo encontra todas as posições onde a sequência do candidato $c.seq$ ocorre na transação atual d (linha 3). Caso exista alguma ocorrência, inicia-se uma repetição (linhas 4 a 14) na qual RGs são criados com as posições onde a sequência ocorre com uso da função $CreateGroup$ (linha 5).

Para cada RG, as seguintes informações são definidas: *i)* o início (r_s) e o fim (r_e) do período de tempo, *ii)* o grupo de posições espaciais (g), *iii)* as ocorrências (occ) e a frequência ($freq$) da sequência, e, por fim, *iv)* o seu estado ($closed$). É importante destacar que novos RGs iniciam com o estado aberto ($closed = falso$) indicando que estes podem ser estendidos. Como saída, o algoritmo fornece um conjunto de novos RGs relativos aos conjuntos de posições onde a sequência candidata $c.seq$ é encontrada em d.

Algoritmo 3 – SplitGroups

```
    Input: c, d, σ
 1  begin
 2      Z ← ∅
 3      pos ← positions(c.seq, d)
 4      while (pos ≠ nil) do
 5          new_group ← CreateGroup(pos, σ)
 6          pos ← pos − new_group
 7          new_rg.r_s ← d.timestamp
 8          new_rg.r_e ← d.timestamp
 9          new_rg.g ← new_group
10          new_rg.occ ← d.timestamp
11          new_rg.freq ← 1
12          new_rg.closed ← False
13          Z ← Z + new_rg
14      end
15  end
    Output: Z - conjunto de novos RGs relativos ao candidato c encontrados em
            d.
```

A função $CreateGroup$ é apresentada no Algoritmo 4. Seu objetivo é gerar um grupo com o conjunto de posições fornecidas. Ela recebe como entrada um conjunto de posições pos e a distancia máxima de ao menos um outro elemento do mesmo grupo σ. O algoritmo começa obtendo uma posição do conjunto fornecido (linha 2). Esta posição passa a ser a primeira a fazer parte de um novo grupo (linha 3) e é removida do conjunto de posições fornecido (linha 4). A partir daí, inicia-se a busca por novas posições que possam fazer parte do novo grupo (linhas 5 a 16), respeitando o limite σ (linha 9) fornecida pelo usuário. Como saída, este algoritmo fornece um novo grupo, que é subconjunto do

grupo de posições pos fornecido.

Algoritmo 4 – CreateGroup

```
   Input: pos, σ
 1 begin
 2     n ← next(pos)
 3     new_group ← n
 4     pos ← pos − n
 5     p ← next(new_group)
 6     while (p ≠ nil) do
 7         q ← next(pos)
 8         while (q ≠ nil) do
 9             if (dist(p, q) ≤ σ) then
10                 new_group ← new_group + q
11                 pos ← pos − q
12             end
13             q ← next(pos)
14         end
15         p ← next(new_group)
16     end
17 end
   Output: new_group - um subconjunto de pos que forma um grupo.
```

A função $MergeOpenKernelRangeGroups$ é apresentada no Algoritmo 5. Seu objetivo é tentar estender o período dos RGs cujos atributos $closed$ tenha valor *falso* gerando novos com período de tempo maior e, possivelmente, um grupo maior. Ela recebe como entrada um conjunto de RGs RG de um candidato, a marcação de tempo atual $timestamp$, e os limites definidos pelo usuário γ, β e σ.

A função tenta estender os períodos através da união de cada elemento do conjunto de RGs com um outro elemento do mesmo conjunto, ambos com atributos $closed$ com valor *falso* (linhas 6 a 21). Caso seus grupos de posições possuam interseção ou uma posição de um grupo tenha uma distância menor ou igual a σ de uma posição qualquer do outro grupo (linhas 8 e 9) e a frequência do novo grupo gerado seja maior que γ (linhas 14 e 16). Se todas as condições forem verdadeiras, então o novo RG gerado é adicionado ao conjunto RG (linha 17) e os elementos unidos são inseridos a uma lista para remoção (linha 19).

Se houverem elementos inseridos na lista de remoção RGs estes são removidos do conjunto RG (linhas 23 a 25). Por fim, os RGs passam por uma validação através do uso da função $ValidateKernelRangeGroups$ (linha 26). Como saída, o algoritmo fornece o conjunto RG atualizado com as uniões que foram possíveis.

Algoritmo 5 – MergeOpenKernelRangeGroups

Input: $RG, timestamp, \gamma, \beta, \sigma$
begin
  $mergeable \leftarrow True$
  while $(mergeable)$ **do**
    $mergeable \leftarrow False$
    $toRemove \leftarrow \emptyset$
    foreach $(q, r \in RG | q.closed = False \wedge r.closed = False \wedge r \neq q)$ **do**
      $pos \leftarrow q.group \cup r.group$
      $group \leftarrow CreateGroup(pos, \sigma)$
      if $(pos - group = \emptyset)$ **then**
        $new_rg.r_s \leftarrow min(q.r_s, r.r_s)$
        $new_rg.r_e \leftarrow max(q.r_e, r.r_e)$
        $new_rg.g \leftarrow group$
        $new_rg.occ \leftarrow q.occ \cup r.occ$
        $new_rg.freq \leftarrow \frac{suport(new_rg.occ)}{|[new_rg.r_s..new_rg.r_e]|}$
        $new_rg.closed \leftarrow False$
        if $(new_krg.freq \geq \gamma)$ **then**
          $RG \leftarrow RG + new_rg$
          $mergeable \leftarrow True$
          $toRemove \leftarrow toRemove + q + r$
        end
      end
    end
    foreach $(k \in toRemove)$ **do**
      $RG \leftarrow RG - k$
    end
    $RG \leftarrow ValidateKernelRangeGroups(RG, timestamp, \gamma, \beta)$
  end
end
Output: RG - conjunto de RGs

O algoritmo G-STSM busca sempre estender os RGs, contudo deve respeitar os limites definidos pelo usuário. Se para algum RG, a extensão do período de tempo resultar em uma frequência menor que γ, o atributo do RG $krg.closed$ é atualizado para *verdadeiro*. A partir deste momento ele não é mais capaz de ter seu período estendido e passa a atender todas as condições da Definição 1, em outras palavras ele se torna um KRG.

A função $ValidateKernelRangeGroups$ é apresentada no Algoritmo 6. Seu objetivo é verificar se os limites definidos pelo usuário foram observados nos RGs, verificando se eles ainda podem ser estendidos mantendo-se a frequência maior ou igual a mínima γ e se o tamanho mínimo do grupo β ocorre. Ela recebe como entrada um conjunto

de RGs RG de uma sequência candidata, a marcação de tempo referente à transação atual da janela deslizante $timestamp$, os limites definidos pelo usuário γ e β. Para cada elemento do conjunto RG com atributo $closed$ com valor *falso* ($krg.closed = falso$) (linha 3 a 11), calcula-se a frequência estendendo o período de tempo do RG até a marcação de tempo atual (linha 4). Depois, verifica-se se a frequência é menor que o valor mínimo definido pelo usuário (linhas 5 a 9). Se isso ocorrer, altera-se o estado ($rg.closed$) para *verdadeiro* (linha 6), e verifica-se se o tamanho do grupo ficou abaixo do mínimo definido pelo usuário, marcando tal elemento para remoção caso isso ocorra (linhas 7 a 9). Por fim, todos os elementos marcados são removidos do conjunto RG (linhas 12 a 14). Como saída, a função retorna o conjunto RG atualizado.

Algoritmo 6 – ValidateKernelRangeGroups

```
   Input: RG, timestamp, γ, β
 1 begin
 2 |  toRemove ← ∅
 3 |  foreach (rg ∈ RG|rg.closed = False) do
 4 |  |  freq ← support(rg.occ) / |[rg.r_s..timestamp]|
 5 |  |  if (freq < γ) then
 6 |  |  |  rg.closed ← True
 7 |  |  |  if (length(rg.g) < β) then
 8 |  |  |  |  toRemove ← toRemove + rg
 9 |  |  |  end
10 |  |  end
11 |  end
12 |  foreach (k ∈ toRemove) do
13 |  |  RG ← RG − k
14 |  end
15 end
   Output: RG
```

É importante observar que a função $ValidateKernelRangeGroups$ é executada sem o conhecimento dos dados das próximas marcações de tempo do conjunto de dados. Isso significa que um RG que tenha seu atributo $closed$ alterado para o valor *verdadeiro* em uma dada marcação de tempo talvez pudesse manter uma frequência maior que o mínimo definido pelo usuário com a utilização dos próximos registros. A solução para tal problema é dada na Subseção 4.2.3, onde é apresentada a função $MergeKernelRangeGroups$, na qual RGs com estado fechado são unidos para formar um novo.

A função $ValidateAndClose$ é apresentada no Algoritmo 7. Ela é apresentada aqui por conta de sua semelhança com a função $ValidateKernelRangeGroups$. A

diferença entre as duas se deve principalmente ao contexto onde são executadas. Como a $ValidateKernelRangeGroups$ é executada durante a varredura do conjunto de dados, no escopo da busca por KRG ela sempre verifica se é possível estender o período de tempo do RG, já a função $ValidateAndClose$ por ser executada após a leitura de todo o conjunto de dados não precisa verificar a frequência dos RGs eles sempre apresentam frequência maior ou igual γ. Dessa forma, a única verificação necessária no contexto da função $ValidateAndClose$ se refere ao tamanho do grupo, que deve ser maior ou igual a β.

A função $ValidateAndClose$ recebe como entrada um conjunto de RGs referentes a um candidato e o valor definido pelo usuário como tamanho mínimo do grupo. Para cada elemento do conjunto de RGs recebido como entrada, que possua o valor do atributo $closed$ com o valor *falso* (linhas 3 a 8), a função altera o valor do atributo $closed$ para *verdadeiro* (linha 4) e verifica se o tamanho do grupo é menor que o permitido (linha 5), se sim, ele é adicionado a um conjunto de elementos que serão removidos (linha 6). Por fim, a função remove do conjunto de entrada todos os elementos marcados para remoção (linhas 9 a 11). Como saída, o algoritmo fornece o conjunto atualizado dos RG recebidos, sendo agora todos validados quanto ao tamanho mínimo do grupo. Dessa forma, um conjunto de KRG.

Algoritmo 7 – ValidateAndClose

```
   Input: RG, β
 1 begin
 2 |  toRemove ← ∅
 3 |  foreach (rg ∈ RG|rg.closed = False) do
 4 |  |  rg.closed ← True
 5 |  |  if (length(rg.g) < β) then
 6 |  |  |  toRemove ← toRemove + rg
 7 |  |  end
 8 |  end
 9 |  foreach (k ∈ toRemove) do
10 |  |  RG ← RG − k
11 |  end
12 end
   Output: RG
```

Lema para $FindKernelRangeGroup$: Sejam c e d um candidato e a transação atual de D, respectivamente. O Algoritmo 2 em conjunto com a função $ValidateAndClose$ torna possível encontrar todos os KRG.

Prova: Seja $rg \in c.rgs$, um KRG da sequência s no período de tempo r e grupo g após a execução da função $ValidateAndClose$; (*i.e.* rg não pode ser descoberto por meio de união com qualquer outro KRG cujo atributo $closed$ tenha valor *falso* em $c.rgs$, então:

1. $freq(s, r, g) \geq \gamma$. De acordo com a definição de KRG (Definição 1), s é frequente em cada KRG. Assim, o Algoritmo 2 gera os RG iniciais em que s é frequente e, se rg é um novo KRG resultante de um processo de união, ele verifica a frequência de s em rg. Suponha que $freq(s, r, g) < \gamma$, o Algoritmo 2 não poderia ter criado rg como um KRG, o que seria uma contradição.

2. $|g| \geq \beta$. De acordo com a Definição 1, g tem um tamanho mínimo. Assim, o Algoritmo 2 ou a função $ValidateAndClose$ validaram rg verificando o limite definido pelo usuário β. Suponha que $|g| < \beta$, o Algoritmo 2 ou a função $ValidateAndClose$ não poderiam ter validado o tamanho de g e rg não seria um KRG, o que seria uma contradição.

3. $\forall a$ tal que $a \subset r$ e $a_s = r_s$, temos os dois casos:

   - s é suportado pela primeira e última marcações de tempo em r. Então, s tem suporte inferior em qualquer sub-faixa de r.
   - $freq(s, a, g) \geq \gamma$, caso contrário, a não teria sido estendido para chegar a r, o que seria uma contradição.

4. $\forall b$ em um grupo tal que $g \subseteq b$, temos por definição que $occur(s, r, b - g) = 0$, então $occur(s, r, b) = occur(s, r, g)$.

5. s ocorre em todas as $sts(g)$, caso contrário, se existir uma TS que não contenha uma ocorrência de s, ela não será incluída em $sts(g)$. Então, o número de ocorrências de s em qualquer subconjunto b de g é menor que em g: $occur(s, r, b) < occur(s, r, g)$.

Com base nas cinco observações acima, $c.rgs$ é formado por todos os KRG descobertos de s em PR e PG levando em conta γ, β e σ. Assim, KRG é o conjunto de todos os KRG $krg = (s, r, g)$ em PR e PG levando em conta γ, β e σ.

Uma prova semelhante é aplicada para a definição de SRG na próxima subseção.

4.2.3 União dos Kernel Range-Groups

A função $MergeKernelRangeGroups$ é a segunda a ser chamada pelo algoritmo principal, ela é apresentada no Algoritmo 8. Seu objetivo é unir KRGs respeitando a frequência mínima definida pelo usuário γ. Ela recebe como entrada um conjunto de KRGs KRG de um candidato e a frequência mínima definida pelo usuário γ.

O algoritmo inicia definindo uma *flag* $mergeable$ como *verdadeira* (linha 2), enquanto ela for verdadeira, tenta-se unir KRGs (linhas 3 a 20). A *flag* é definida como falsa (linha 4) e um conjunto de elementos que serão removidos ($toRemove$) do conjunto KRG, uma vez que foram unidos gerando um novo KRG, é criado (linha 5). Verifica-se, então, os elementos do conjunto KRG (linhas 6 a 16), utilizando dois por vez, para saber se possuem interseção no espaço (linha 7) e se a junção dos elementos mantém uma frequência maior que a mínima definida pelo usuário γ (linha 10).

Caso as duas condições citadas sejam observadas, os elementos são unidos e adicionados ao conjunto KRG (linha 11). A *flag* $mergeable$ é, portanto, atualizada para *verdadeira* (linha 12) e os dois elementos unidos são adicionados ao conjunto $toRemove$. Por fim, todos os elementos que foram unidos do conjunto KRG são removidos (linhas 17 a 19). Como resultado, o algoritmo retorna o conjunto de entrada KRG contendo os KRGs unidos, quando possível.

Lema para $MergeKernelRangeGroups$: Seja KRG o conjunto de KRG de s no período de tempo r e o grupo g em relação aos limites γ, β e σ. O Algoritmo 8 torna possível encontrar todos os SRG SRG.

Prova: Seja $krg \in KRG$, um SRG de s em r e g após a execução do Algoritmo 8 (*i.e.* krg não pode ser mesclado com qualquer outro KRG em KRG), então:

1. $freq(s, r, g) \geq \gamma$. De acordo com a definição de SRG (Definição 2), s é frequente em cada SRG. Assim, o Algoritmo 8 gera KRG em que s é frequente e, se krg é o novo KRG resultante de um processo de mesclagem, ele verifica a frequência de s em krg. Se krg não é o resultado de um processo de mesclagem, a frequência foi verificada anteriormente usando o Algoritmo 2. Suponha que $freq(s, r, g) < \gamma$, o Algoritmo 8 não poderia ter criado krg como um SRG, o que seria uma contradição.

2. $|g| \geq \beta$. De acordo com a Definição 2, g tem um tamanho mínimo. Assim, se

Algoritmo 8 – MergeKernelRangeGroups

Input: KRG, γ
begin
  $mergeable \leftarrow True$
  while $(mergeable)$ **do**
    $mergeable \leftarrow False$
    $toRemove \leftarrow \emptyset$
    foreach $(q, u \in KRG | q \neq u)$ **do**
      if $((q.g \cap u.g) \neq \emptyset)$ **then**
        $start \leftarrow min(q.r_s, u.r_s)$
        $end \leftarrow max(q.r_e, u.r_e)$
        if $(\frac{support(q.occ \cup u.occ)}{|[start..end]|} \geq \gamma)$ **then**
          $KRG \leftarrow KRG + q \cup u$
          $mergeable \leftarrow True$
          $toRemove \leftarrow toRemove + q + u$
        end
      end
    end
    foreach $(k \in toRemove)$ **do**
      $KRG \leftarrow KRG - k$
    end
  end
end
Output: KRG

krg é resultado de um processo de mesclagem, então o grupo g é formado pelo Algoritmo 8 com a união entre as posições dos grupos combinados fundidos (que também satisfazem a condição de tamanho mínimo de grupos pela Definição 1). Se krg não é o resultado de um processo de mesclagem, o tamanho do grupo já foi verificado anteriormente usando o Algoritmo 2. Suponha que $|g| < \beta$, o Algoritmo 8 não poderia ter gerado este SRG pelo processo de mesclagem ou o Algoritmo 2 e a função $ValidateAndClose$ não poderiam ter validado krg como um KRG, o que seria uma contradição.

3. $\forall a$ tal que $r \subseteq a$, temos um dos seguintes casos:

   - $sup(s, a - r, g) > 0$, então s não é frequente em a (caso contrário, consideremos r' o período de tempo em que s está presente em a, então r e r' teriam sido mesclados).
   - $sup(s, a - r, g) = 0$, então $sup(s, a, g) = sup(s, r, g)$ (neste caso, s pode permanecer frequente em a ou não, dependendo do tamanho de a).

4. De acordo com a Definição 2, s é suportado pela primeira e última marcações de tempo no período de tempo em r. Então, s tem suporte inferior em qualquer sub-faixa de r.

5. $\forall b$ em um grupo tal que $g \subseteq b$, temos por definição que $occur(s, r, b - g) = 0$, então $occur(s, r, b) = occur(s, r, g)$ (neste caso, s permanece frequente em r).

6. De acordo com a Definição 2, s ocorre em todas as $sts(g)$, caso contrário, se existir uma TS que não contenha uma ocorrência de s, ela não será incluída em $sts(g)$. Então, o número de ocorrências de s em qualquer subconjunto b de g é menor que em g: $occur(s, r, b) < occur(s, r, g)$.

Com base nas seis observações acima, o KRG atualizado contém o conjunto de SRG resultante do processo de união de todos os KRG de s em PR e PG considerando γ, β e σ. Assim, SRG é o conjunto de todos os SRG $srg = (s, r, g)$ em PR e PG relativos a γ, β e σ.

4.2.4 Geração dos Candidatos

A função $GenerateCandidates$ é a terceira a ser chamada pelo algoritmo principal, ela é apresentada no Algoritmo 9. O processo do G-STSM se baseia na geração de candidatos, seu objetivo é gerar candidatos de tamanho de sequência $k + 1$ a partir dos SRGs de tamanho de sequência k. Ela recebe como entrada um conjunto de SRGs SRG_k.

A propriedade antimonotônica diz que uma sequência só pode ser frequente se todas as suas subsequências o forem. O algoritmo verifica se (linha 3):*i)* o final de uma sequência de um elemento do conjunto SRG_k é igual ao início da sequência de um outro elemento $(x.s_2, \ldots, x.s_k) = (y.s_1, \ldots, y.s_{k-1})$; *ii)* a interseção dos SRG ao longo do período de tempo ($[x.r_s..x.r_e] \cap [y.r_s..y.r_e]$); *iii)* e do conjunto de posições espaciais ($x.g \cap y.g$). Se as sequências de tamanho k dos SRGs x e y têm uma subsequência comum, mas seus intervalos de tempo ou grupo espacial não possuem interseções, eles não são considerados para gerar um novo candidato de tamanho $k + 1$.

Se todas as restrições forem respeitadas, define-se um candidato com uma nova

sequência (linha 4), um novo período de tempo (linha 5) e um novo grupo de posições espaciais (linha 6). O período de tempo referente ao candidato é gerado a partir da marcação de tempo de início r_s da primeira sequência e fim da segunda menos um $r_e - 1$, e o conjunto de posições é igual a interseção dos SRGs que geram o candidato. Por fim, o novo candidato é adicionado ao conjunto de sequências candidatas C_{k+1} (linha 7). Como resultado, a função retorna o conjunto de sequências candidatas de tamanho $k + 1$.

Algoritmo 9 – GenerateCandidates

Input: SRG_k
1 **begin**
2 $C_{k+1} \leftarrow \emptyset$
3 **foreach** $(x, y \in SRG_k)$ *tal que:* $(((x.s_2, \ldots, x.s_k) = (y.s_1, \ldots, y.s_{k-1})) \wedge ([x.r_s..x.r_e] \cap [y.r_s..y.r_e] \neq \emptyset) \wedge ((x.g \cap y.g) \neq \emptyset))$ **do**
4 $z \leftarrow (x.s_1, \ldots, x.s_k, y.s_k)$
5 $p \leftarrow [x.r_s \ldots (y.r_e - 1)]$
6 $u \leftarrow (x.g \cap y.g)$
7 $C_{k+1} \leftarrow C_{k+1} + (z, p, u, \emptyset)$
8 **end**
9 **end**
Output: C_{k+1} conjunto de candidatos tendo tamanho $k + 1$

4.3- Exemplo

Esta seção busca facilitar a compreensão deste trabalho fornecendo um exemplo do funcionamento do algoritmo G-STSM. As informações necessárias para o processamento do algoritmo são: um conjunto de dados STS D, o conjunto de todos os itens distintos I apresentados em D, o conjunto de posições espaciais P referentes às STS do conjunto de dados, além de limites definidos pelo usuário para a frequência mínima γ, o tamanho mínimo de um grupo β e a distancia máxima de ao menos um outro elemento do mesmo grupo σ.

A Tabela 2 apresenta o conjunto de dados STS D sintético utilizado neste exemplo, no qual é possível observar alguns conceitos já abordados. Tal conjunto de dados apresenta nove STS: $<st_1, st_2, \ldots, st_9>$. Seja $P = \{p_1, p_2, \ldots, p_9\}$ o conjunto de posições referentes ao conjunto de dados STS. Cada STS está associada a uma TS t, representadas pelas colunas da tabela, e a uma posição espacial p, onde $p \in P$. Todas as TS

apresentam três observações, representadas pelas linhas da tabela. Por motivos de simplicidade, algumas células da tabela apresentam um traço (-) representando ausência de valor. As outras células apresentam letras que representam eventos quaisquer. Observando a Tabela 2 é fácil perceber que o conjunto de todos os itens distintos I apresentados em D é: $I = \{A, B, C, D\}$.

Tabela 2 – Conjunto de Dados STS. Nove TS, cada uma com três observações.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
v_1	D	A	C	-	A	C	B	B	B
v_2	D	D	-	A	B	A	A	-	A
v_3	B	B	B	-	-	C	-	-	C

É importante observar que apesar da apresentação em forma tabular dar uma visão inicial das posições alinhadas, as posições de p_1 a p_9 podem estar distribuídas em uma infinidade de formas. São importantes para a formação de grupos de posições espaciais, a distribuição das posições e a distancia máxima de ao menos um outro elemento do grupo (σ) definida pelo usuário.

Para este exemplo será considerada a distribuição da Figura 4, que apresenta dois eixos ortogonais e a distribuição dos pontos em relação a estes eixos. Por motivos de simplicidade, consideram-se todas as posições em um mesmo plano. A unidade de medida não é importante para o algoritmo, mas é necessário que a distribuição das posições e a distancia máxima de ao menos um outro elemento do mesmo grupo fornecida pelo usuário utilizem a mesma unidade de medida.

Os valores dos limites definidos pelo usuário considerados neste exemplo serão: a frequência mínima $\gamma = 60\%$, o tamanho mínimo de um grupo $\beta = 2$ e a distância máxima de ao menos um outro elemento do mesmo grupo $\sigma = 1$. Dessa forma, temos todos os dados de entrada necessários para iniciar o processo sumarizados a seguir:

- conjunto de dados STS D sintético apresentado na Tabela 2;
- conjunto de todos os itens distintos apresentados em D, $I = \{A, B, C, D\}$;
- conjunto de posições $P = \{p_1, p_2, \ldots, p_9\}$ distribuídas conforme a Figura 4;
- frequência mínima $\gamma = 60\%$;
- tamanho mínimo de um grupo $\beta = 2$; e
- distancia máxima de ao menos um outro elemento do mesmo grupo $\sigma = 1$.

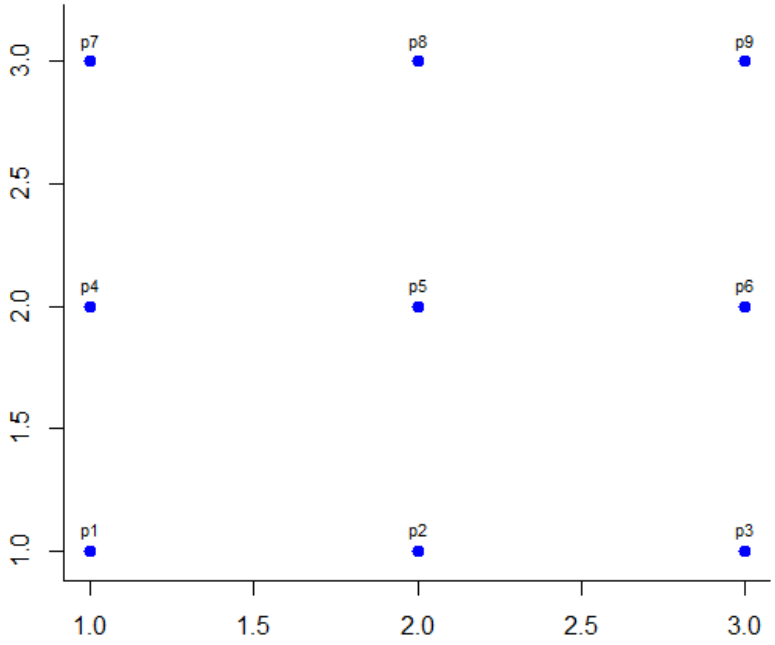


Figura 4 – Distribuição das posições utilizadas no exemplo.

O processo utilizado se baseia na geração de candidatos, dessa forma, como primeiro passo é necessário gerar candidatos com tamanho de sequência um. Cada item de I gera um candidato e informa-se que ele tem a possibilidade de ocorrer por todo o período de tempo existente no conjunto de dados e em qualquer posição espacial. Cada candidato pode ocorrer em diferentes períodos de tempo e em diferentes posições, gerando os RGs. Cada candidato pode ter vários RGs, mas começa com a lista de RGs vazia. Dessa forma, a estrutura de dados gerada inicialmente é como apresentada na Tabela 3.

Tabela 3 – Estrutura de dados referente aos candidatos.

seq	$range$	pos	rgs
A	[1, 3]	$p_1, p_2, ..., p_9$	-
B	[1, 3]	$p_1, p_2, ..., p_9$	-
C	[1, 3]	$p_1, p_2, ..., p_9$	-
D	[1, 3]	$p_1, p_2, ..., p_9$	-

Para cada tamanho de sequência, após a formação dos candidatos de tamanho k inicia-se uma repetição de passos, são eles:

1. Com o intuito de encontrar os KRGs, percorre-se todo o conjunto de marcações de tempo utilizando uma janela deslizante de tamanho igual a da sequência buscada k.

Para cada candidato c no conjunto de marcações de tempo de uma janela deslizante de tamanho k, buscam-se as ocorrências da sequência desejada $c.seq$ executando os seguintes passos:

a. com as posições das STS nas quais a sequência $c.seq$ buscada ocorre, criar os grupos de posições espaciais;

b. gerar os novos RGs a partir dos grupos de posições espaciais;

c. tentar unir os RGs abertos (recém criados e pré existentes);

d. validar os RGs abertos. Caso estender seu período de tempo até a marcação de tempo corrente torne sua frequência menor que o valor mínimo definido pelo usuário ($\gamma = 60\%$), atualizar o estado para fechado e validar o tamanho mínimo do grupo ($\beta = 2$), caso seu grupo seja menor que o tamanho mínimo ele é desconsiderado.

2. mesclar os KRGs para identificar SRGs; e

3. gerar os candidatos.

A Figura 5 permite uma visão mais geral dos passos do processo utilizado pelo G-STSM, assim como a saída fornecida por cada passo do processo.

Os itens do passo *1* devem ser executados individualmente para cada candidato, contudo, vamos executar de maneira simultânea para todos os candidatos, com o intuito de simplificar o exemplo. A primeira rodada se inicia buscando por ocorrências das sequências candidatas de tamanho um para a primeira marcação de tempo. Para cada candidato, o passo *1.a* é executado, verificando as ocorrências da sequência candidata. As posições nas quais a sequência ocorre são registradas e busca-se formar grupos de posições de acordo com a distancia máxima de ao menos um outro elemento do grupo $\sigma = 1$. O resultado deste passo pode ser verificado na Tabela 4.

Tabela 4 – Resultado da execução do passo *1.a* - Criação dos grupos de posições para a primeira marcação de tempo.

sequência	grupos de posições
A	$\{p_2, p_5\}$
B	$\{p_7, p_8, p_9\}$
C	$\{p_3, p_6\}$
D	$\{p_1\}$

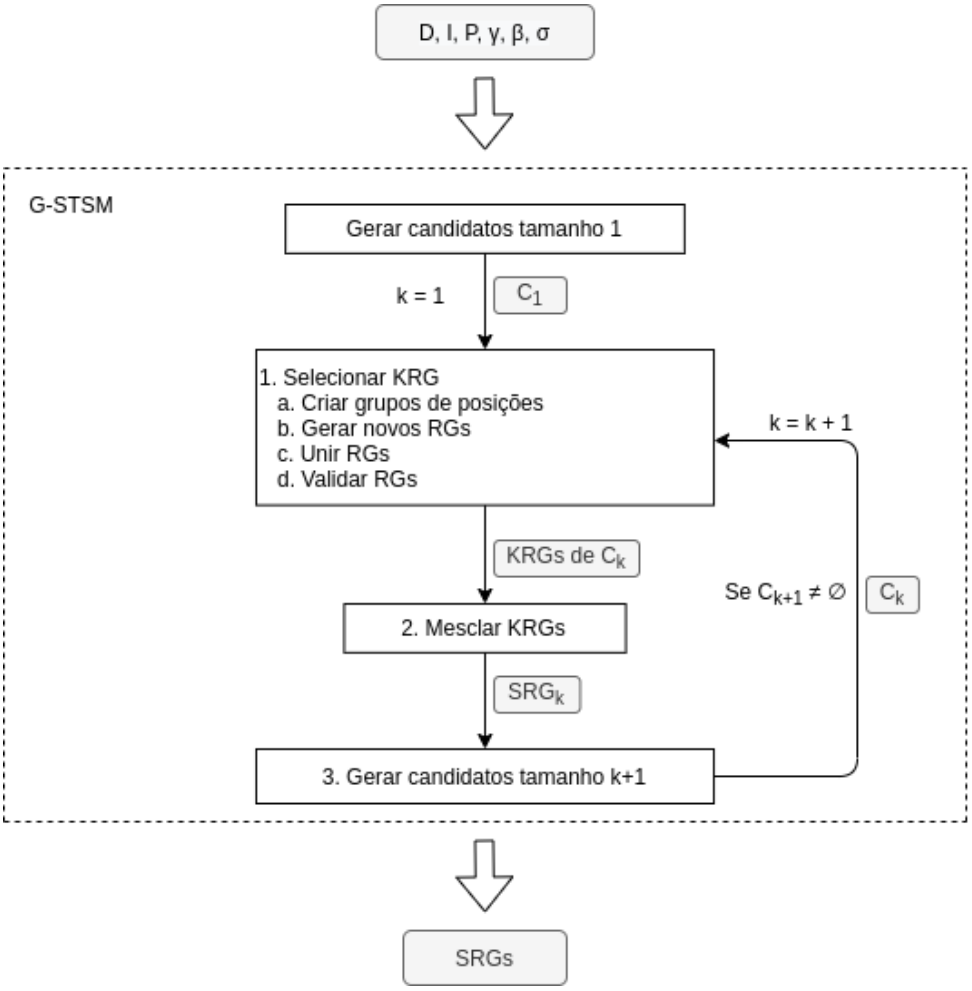


Figura 5 – Processo de mineração utilizado pelo G-STSM. Retângulos com cantos arredondados e fundo cinza representam dados. Retângulos com fundo branco se referem a processamento.

A seguir, o passo *1.b* é executado. Para cada um dos grupos de posições espaciais, criam-se RGs e estes são associados a sequência candidata. O período de tempo dos RGs é inicializado com a marcação de tempo do início da janela deslizante, seu início e fim coincidem e tem o valor um. É também registrado que existe uma ocorrência na marcação de tempo corrente inicializando-se o atributo occ com o valor do início da janela deslizante. A frequência é sempre inicializada com o valor um (100%). O atributo $closed$ é sempre inicializado com o valor *falso*, o que significa que o RG pode ser estendido. O resultado deste passo pode ser verificado na Tabela 5.

O próximo passo a ser executado é o *1.c*, contudo não existem RGs pré existentes para que seja possível efetuar alguma união, apenas os recém criados existem. Assim, não faz sentido pensar em unir apenas os RGs recém criados, a formação de grupos

Tabela 5 – Resultado da execução do processo até o passo *1.b* - Geração dos RGs referentes aos grupos de posições gerados.

s	r	g	occ	$freq$	$closed$
A	[1, 1]	$\{p_2, p_5\}$	1, 1	1	falso
B	[1, 1]	$\{p_7, p_8, p_9\}$	1, 1, 1	1	falso
C	[1, 1]	$\{p_3, p_6\}$	1, 1	1	falso
D	[1, 1]	$\{p_1\}$	1	1	falso

gera o menor número de grupos possível. Com isso, o resultado deste passo é o mesmo do passo anterior, ou seja, os dados da Tabela 5 são mantidos.

O passo *1.d* é, então, executado, verificando se estender o período de tempo dos RGs abertos (no contexto atual todos) até a marcação de tempo corrente (1) torna a frequência menor que o mínimo definido pelo usuário $\gamma = 60\%$. A resposta é não para todos. Na primeira marcação de tempo, o período de todos os RGs é $r = [1, 1]$, ou seja, a marcação de tempo inicial $r_s = 1$ e a marcação de tempo final $r_e = 1$ coincidem e todos os RGs possuem frequência igual a 100%. Assim, os RG mantém seu estado $closed$ com valor *falso* e não são validados quanto ao tamanho mínimo do grupo. O resultado do processo executado até aqui é o mesmo da Tabela 5. É importante observar que apesar de todos os RGs fazerem parte de uma mesma tabela neste exemplo, cada um é atribuído a sua respectiva sequência candidata.

Os passos do item *1* do processo se repetem para a segunda marcação de tempo. Como resultado do passo *1.a*, temos os grupos de posições conforme a Tabela 6. Observe-se que a sequência “A” ocorre em quatro posições (p_4, p_6, p_7, p_9), contudo elas geram dois grupos distintos dado o limite da distância máxima de ao menos um outro elemento do mesmo grupo $\sigma = 1$ e a distribuição das posições, conforme Figura 4.

Tabela 6 – Resultado da execução do passo *1.a* - Criação dos grupos de posições, para a segunda marcação de tempo.

sequência	grupos de posições
A	$\{p_4, p_7\}$, $\{p_6, p_9\}$
B	$\{p_5\}$
D	$\{p_1, p_2\}$

A Tabela 7 mostra o resultado do passo *1.b* para a segunda marcação de tempo. Observe-se que os diferentes grupos de posições referentes a sequência “A” geram diferentes RGs.

Inicia-se, então, o passo *1.c*, onde os RGs que acabaram de ser criados (Tabela

Tabela 7 – Resultado da execução do passo *1.b* - Novos RGs criados a partir dos grupos de posições referentes à segunda marcação de tempo para todas as sequências candidatas.

s	r	g	occ	$freq$	$closed$
A	[2, 2]	$\{p_4, p_7\}$	2, 2	1	falso
A	[2, 2]	$\{p_6, p_9\}$	2, 2	1	falso
B	[2, 2]	$\{p_5\}$	2	1	falso
D	[2, 2]	$\{p_1, p_2\}$	2, 2	1	falso

7) são unidos com os que já existiam (Tabela 5). As condições para que isso ocorra são relacionadas às posições e à frequência. Só é possível mesclar dois RGs se houver interseção entre os grupos de posições ou se um elemento qualquer de um grupo de posições estiver a uma distância menor ou igual a σ de um elemento qualquer do outro grupo de posições. Em outras palavras, uma das condições que tornam possível mesclar dois RGs é que utilizando todo o conjunto de posições de ambos seja possível gerar um único grupo.

Em relação a frequência, só é possível mesclar dois RGs se a frequência do novo RG gerado for maior ou igual a frequência mínima γ. O resultado do passo *1.c* é mostrado na Tabela 8. Os RGs que são resultados da mesclagem de dois outros são facilmente reconhecíveis nesta tabela, eles tem seu período de tempo ampliado e registram suas ocorrências em mais de uma marcação de tempo. Todos os RGs novos são mesclados aos RGs preexistentes referentes as suas respectivas sequências. É importante ressaltar que os três RGs (dois recém criados e um preexistente) referentes a sequência “A” foram mesclados dando origem a um único RG.

Tabela 8 – Resultado da execução do processo até o passo *1.c* - União dos RGs abertos referentes à segunda marcação de tempo.

s	r	g	occ	$freq$	$closed$
A	[1, 2]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	1, 1, 2, 2, 2, 2	1	falso
B	[1, 2]	$\{p_5, p_7, p_8, p_9\}$	1, 1, 1, 2	1	falso
C	[1, 1]	$\{p_3, p_6\}$	1, 1	1	falso
D	[1, 2]	$\{p_1, p_2\}$	1, 2, 2	1	falso

A seguir, a validação dos RGs, passo *1.d*, é executada. Dentre os RGs da Tabela 8, o único cuja frequência ficará abaixo do valor mínimo é o referente à sequência “C”. Caso se estenda seu período de tempo, sua frequência será igual a 50%. Seu estado $closed$ deve ser atualizado para *verdadeiro* e o tamanho de seu grupo deve ser verificado. O tamanho de seu grupo de posições é exatamente igual ao mínimo definido pelo usuário

$\beta = 2$, portanto, ele será mantido. O resultado do passo *1.d* é apresentado na Tabela 9, na qual a diferença em relação à saída do passo anterior (Tabela 8) é a atualização do estado $closed$ do RG referente à sequência “C”.

Tabela 9 – Resultado da execução do processo até o passo *1.d* - Validação dos RGs referentes à segunda marcação de tempo.

s	r	g	occ	$freq$	$closed$
A	[1, 2]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	1, 1, 2, 2, 2, 2	1	falso
B	[1, 2]	$\{p_5, p_7, p_8, p_9\}$	1, 1, 1, 2	1	falso
C	[1, 1]	$\{p_3, p_6\}$	1, 1	1	verdadeiro
D	[1, 2]	$\{p_1, p_2\}$	1, 2, 2	1	falso

Uma nova rodada do passo *1* é iniciada para a terceira marcação de tempo. O passo *1.a* tem como saída os grupos de posições conforme a Tabela 10.

Tabela 10 – Resultado da execução do passo *1.a* - Criação dos grupos de posições para a terceira marcação de tempo.

sequência	grupos de posições
B	$\{p_1, p_2, p_3\}$
C	$\{p_6, p_9\}$

Dessa forma, o passo *1.b* se resume a criar dois RGs referentes aos grupos de posições das sequências encontradas. Tal resultado pode ser observado na Tabela 11.

Tabela 11 – Resultado do passo *1.b* - Geração dos RGs referentes aos grupos de posições da terceira marcação de tempo.

s	r	g	occ	$freq$	$closed$
B	[3, 3]	$\{p_1, p_2, p_3\}$	3, 3, 3	1	falso
C	[3, 3]	$\{p_6, p_9\}$	3, 3	1	falso

O passo *1.c* fará com que o novo RG referente à sequência “B” seja unido com o previamente existente. Seus grupos não possuem interseção, mas a posição p_2 do novo RG e a posição p_5 do preexistente estão a uma distância igual a distância mínima para um outro elemento do grupo definida pelo usuário $\sigma = 1$. Além disso, o novo RG gerado possui frequência acima do mínimo $\gamma = 60\%$. Assim, o resultado da saída do passo *1.c* pode ser visto na Tabela 12.

No passo *1.d*, os RGs são validados. Um RG referente à sequência “C” possui o estado $closed$ com o valor *verdadeiro*, então, ele não passa pela validação, apenas os RGs abertos passam por este processo. Verifica-se, então, se estender o período gera uma frequência menor do que a mínima. Isso não acontece com nenhum RG aberto.

Tabela 12 – Resultado da conclusão do passo *1.c* referente à terceira marcação de tempo.

s	r	g	occ	$freq$	$closed$
A	[1, 2]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	1, 1, 2, 2, 2, 2	1	falso
B	[1, 3]	$\{p_1, p_2, p_3, p_5, p_7, p_8, p_9\}$	1, 1, 1, 2, 3, 3, 3	1	falso
C	[1, 1]	$\{p_3, p_6\}$	1, 1	1	verdadeiro
C	[3, 3]	$\{p_6, p_9\}$	3, 3	1	falso
D	[1, 2]	$\{p_1, p_2\}$	1, 2, 2	1	falso

Para os RGs referentes às sequências “A” e “D”, estender o período torna a frequência aproximadamente 66%. Os RGs referentes as sequências “B” e “C” apresentam frequência de 100%. O resultado do passo *1* se mantém o mesmo mostrado na Tabela 12.

Ao fim do conjunto de dados alguns RGs ainda se encontram com atributo $closed$ com o valor *falso* e ainda não foram validados, assim tal trabalho deve ser feito, fechar cada um dos RGs abertos e validar o tamanho de seus grupos. Como resultado temos a Tabela 13, onde a única diferença é que todos os RGs agora apresentam seu atributo $closed$ com valor *verdadeiro*. É importante observar que os grupos dos RGs que tiveram seu atributo modificado foram validados, como todos possuíam valor maior ou igual ao mínimo, foram mantidos. Dessa forma, no atual contexto temos todos os KRG gerados.

Tabela 13 – Resuldado da execução do passo *1* - Todos os KRGs ao fim da busca utilizando sequências de tamanho um.

s	r	g	occ	$freq$	$closed$
A	[1, 2]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	1, 1, 2, 2, 2, 2	1	verdadeiro
B	[1, 3]	$\{p_1, p_2, p_3, p_5, p_7, p_8, p_9\}$	1, 1, 1, 2, 3, 3, 3	1	verdadeiro
C	[1, 1]	$\{p_3, p_6\}$	1, 1	1	verdadeiro
C	[3, 3]	$\{p_6, p_9\}$	3, 3	1	verdadeiro
D	[1, 2]	$\{p_1, p_2\}$	1, 2, 2	1	verdadeiro

Uma vez tendo percorrido o conjunto de dados para descobrir os KRGs para sequências de tamanho um devemos começar o passo *2* com intuito de mesclar os KRGs para identificar SRGs. A necessidade deste passo pode ser observada na Tabela 13, a qual traz o resultado dos passos anteriores. Os dois KRGs referentes a sequência “C” podem ser mesclados, gerando um novo KRG com um frequência de aproximadamente 66%, superior a mínima definida pelo usuário $\gamma = 60\%$. O resultado pode ser observado na Tabela 14, onde são apresentados os SRGs, os quais respeitam a Definição 2.

Por fim, o passo *3* busca combinar os SRGs de forma a gerar candidatos. Para isso, tenta-se combinar cada SRG com todos os outros, inclusive com o próprio, verificando as possibilidades de combinação no tempo e no espaço. Por exemplo, se deseja-se

Tabela 14 – Resultado do passo *2* - SRGs gerados a partir dos KRGs.

s	r	g	occ	$freq$	$closed$
A	[1, 2]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	1, 1, 2, 2, 2, 2	1	verdadeiro
B	[1, 3]	$\{p_1, p_2, p_3, p_5, p_7, p_8, p_9\}$	1, 1, 1, 2, 3, 3, 3	1	verdadeiro
C	[1, 3]	$\{p_3, p_6, p_9\}$	1, 1, 3, 3	1	verdadeiro
D	[1, 2]	$\{p_1, p_2\}$	1, 2, 2	1	verdadeiro

formar uma sequência “AC” a sequência “A” deve ter um período de tempo que possibilite isso, ou seja, em alguma parte do seu período de tempo ela tem que ocorrer antes da sequência “C”. O candidato gerado recebe como período de tempo o início da primeira até o fim da segunda menos um. Em relação ao espaço, para que seja possível a construção de uma sequência, suas subsequências devem possuir interseção no grupo de posições espaciais. O candidato que é gerado tem tal interseção como grupo de posições. Baseado nos SRGs apresentados na Tabela 14, foi possível gerar os candidatos apresentados na Tabela 15.

Tabela 15 – Resultado do passo *3* - Candidatos com sequência de tamanho dois gerados a partir dos SRGs de sequências de tamanho um.

seq	$range$	pos	rgs
AA	[1, 1]	$\{p_2, p_4, p_5, p_6, p_7, p_9\}$	-
AB	[1, 2]	$\{p_2, p_5, p_7, p_9\}$	-
AC	[1, 2]	$\{p_6, p_9\}$	-
AD	[1, 1]	$\{p_2\}$	-
BA	[1, 1]	$\{p_2, p_5, p_7, p_9\}$	-
BB	[1, 2]	$\{p_1, p_2, p_3, p_5, p_7, p_8, p_9\}$	-
BC	[1, 2]	$\{p_3, p_9\}$	-
BD	[1, 1]	$\{p_1, p_2\}$	-
CA	[1, 1]	$\{p_6, p_9\}$	-
CB	[1, 2]	$\{p_3, p_9\}$	-
CC	[1, 2]	$\{p_3, p_6, p_9\}$	-
DA	[1, 1]	$\{p_2\}$	-
DB	[1, 2]	$\{p_1, p_2\}$	-
DD	[1, 1]	$\{p_1, p_2\}$	-

Assim, uma nova rodada se inicia, agora buscando KRGs de sequências de tamanho dois, repetindo-se os mesmos passos. Apesar de ser um pequeno exemplo, é importante notar que mesmo se isso fizesse parte de um grande conjunto de dados com centenas ou milhares de registros, com eventos de A a Z, os padrões apresentados aqui seriam encontrados da mesma maneira.

5- Avaliação Experimental

Este trabalho foca na identificação de sequências frequentes restritas no espaço e no tempo. Como mencionado antes, uma abordagem simples para esse problema é a SPADE+DBSCAN. Assim, surge a seguinte questão: como é o desempenho do G-STSM em comparação com a abordagem SPADE+DBSCAN? Uma avaliação experimental foi conduzida para responder a esta pergunta e analisar a sensibilidade do algoritmo G-STSM, variando valores de parâmetros de entrada e o tamanho do conjunto de dados.

Na Seção 5.1 os conjuntos de dados utilizados nos experimentos são apresentados. A Seção 5.2 apresenta as métricas utilizadas na comparação entre as abordagens G-STSM e SPADE+DBSCAN. A Seção 5.3 descreve a configuração dos experimentos realizados. Em seguida, a Seção 5.4 apresenta uma análise comparativa das abordagens G-STSM e SPADE+DBSCAN. Finalmente, a Seção 5.5 apresenta uma análise de sensibilidade do G-STSM, fornecendo mais informações sobre o comportamento da abordagem.

5.1- Conjunto de Dados

O conjunto de dados usado nos experimentos foi o conjunto de dados sísmico de sequências espaciais com marcação de tempo denominado *F3 Block*. Tal conjunto de dados foi produzido pelo método de reflexão sísmica em uma região localizada no setor holandês do Mar do Norte. Os dados sísmicos são obtidos enviando ondas sonoras de alta energia para o solo ou fundo do mar, conforme o caso, como pode ser visto na Figura 6. A amplitude das ondas sonoras refletidas é registrada. Quanto mais tarde a onda sonora refletida chega, mais fundo no solo ela foi refletida. Como resultado, este conjunto de dados contém observações relacionadas ao momento em que a onda sonora chega e atributos relacionados à posição do hidrofone que registrou a onda sonora refletida, ou seja, um conjunto de dados de STS. Os resultados apresentados neste trabalho foram focados em dados públicos do *inline* T401, o qual é composto por 951 sequências com

marcação de tempo e espaço com 462 observações.

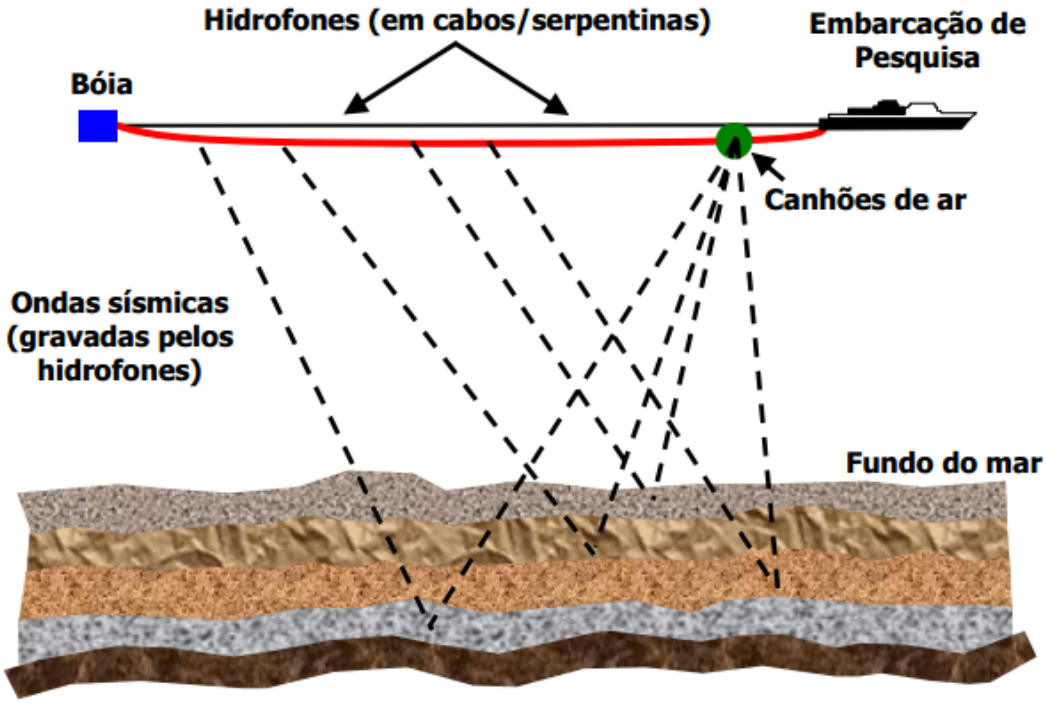


Figura 6 – Obtenção de dados sísmicos marinhos. Ondas de som são enviadas ao fundo do mar, a partir de canhões de ar, refletidas e registradas por hidrofones. Fonte: gov.br/anp.

Os horizontes sísmicos consistem em interfaces de forte reflexão que indicam uma fronteira estratigráfica entre duas regiões diferentes. Neste conjunto de dados, esses horizontes consistem em padrões que foram previamente definidos por especialistas.

O conjunto de dados utilizado foi dividido em dezesseis quadrantes, como pode ser visto na Figura 7. A divisão é mostrada em linhas retas verticais e horizontais em preto. O número de cada quadrante é mostrado em vermelho. As linhas coloridas representam os padrões previamente marcados. Cada quadrante tem aproximadamente 237 sequências com marcação de tempo e de espaço com 115 observações. As duas últimas observações do conjunto de dados T401 são parte dos quadrantes 13, 14, 15 e 16, e as três STS mais à direita fazem parte dos quadrantes 4, 8, 12 e 16. Tanto o conjunto de dados quanto as localizações dos padrões estão disponíveis em (DAL) [2020]. O conjunto de dados é discretizado com um alfabeto de tamanho 25.

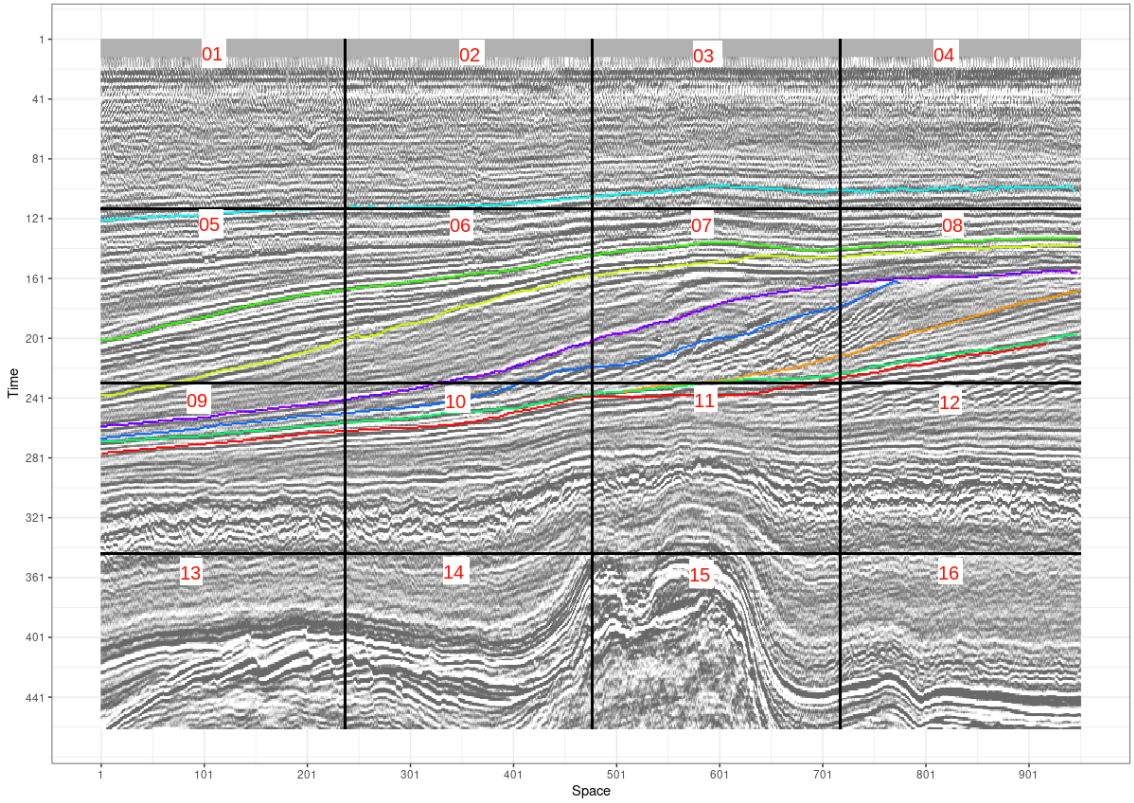


Figura 7 – Conjunto de dados T401, marcação dos padrões (linhas coloridas), e divisão em quadrantes (linhas retas verticais e horizontais em preto com numeração em vermelho). Fonte: OpendTect [2020].

5.2- Métricas

Métricas de classificação foram usadas para avaliar e comparar as abordagens G-STSM e SPADE+DBSCAN. Consideramos duas classes: positiva e negativa. Os pontos positivos correspondem aos padrões previamente marcados e os pontos negativos aos que não o são. P é o número de pontos positivos e N é o número de pontos negativos.

As ocorrências de sequências frequentes que fazem parte de um SRG para a abordagem G-STSM, ou parte de um *cluster* para a abordagem SPADE+DBSCAN, são comparadas com os padrões identificados previamente no conjunto de dados, obtendo-se verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN). TP são pontos que estão em uma sequência frequente e em padrões previamente marcados. FP são pontos que estão em uma sequência frequente e não em padrões previamente marcados. TN são pontos que não fazem parte de uma sequência

frequente e não estão em um padrão previamente marcado. Finalmente, FN são pontos que não fazem parte de uma sequência frequente e estão em um padrão previamente marcado.

A partir dessas medidas, é possível calcular as seguintes métricas: $acurácia$, $precisão$, $sensibilidade$, e f_1, representados pelas equações 1, 2, 3 e 4, respectivamente.

$$acurácia = \frac{TP + TN}{P + N} \tag{1}$$

$$precisão = \frac{TP}{TP + FP} \tag{2}$$

$$sensibilidade = \frac{TP}{TP + FN} \tag{3}$$

$$f_1 = \frac{2 \times precisão \times sensibilidade}{precisão + sensibilidade} \tag{4}$$

A $acurácia$ mede a frequência com que o ponto é classificado corretamente. A $precisão$ relata quantas vezes uma sequência frequente está realmente em um padrão previamente marcado. A $sensibilidade$ significa quantas vezes os padrões previamente marcados são identificados. A f_1 é a média harmônica de $precisão$ e $sensibilidade$.

5.3- Configuração Experimental

Para a análise comparativa (Seção 5.4), os quadrantes de número 5 ao de número 12 foram usados. Este grupo de quadrantes foi especialmente escolhido devido à existência de padrões previamente marcados por especialistas ali presentes, permitindo o uso de métricas de classificação para avaliar ambas as abordagens.

Na análise de sensibilidade (Seção 5.5), o algoritmo G-STSM foi avaliado com um número crescente de quadrantes, começando com apenas um, depois duas vezes maior que a configuração anterior até corresponder a todo o conjunto de dados T401, *i. e.*, com 1, 2, 4, 8 e 16 quadrantes. As variações usadas são mostradas na Tabela 16.

Como parâmetros de entrada, o G-STSM permite que os usuários definam os

Tabela 16 – Variação de quadrantes.

variação	total de quadrantes	conjunto de quadrantes	conjunto de dados
1	1	1	115 x 237
2	2	1+2	115 x 474
3	4	1+2+5+6	230 x 474
4	8	5+6+7+8+9+10+11+12	230 x 951
5	16	T401	462 x 951

seguintes limites: frequência mínima (γ), tamanho mínimo de um grupo (β) e distância máxima de pelo menos uma outra posição do grupo (σ). Os valores de γ podem variar no intervalo $]0, 1]$. Para β, os valores inteiros a partir de 2 podem ser usados, e para σ, os valores inteiros começando de 1.

O algoritmo SPADE recebe um conjunto de dados de entrada em um formato de lista vertical que consiste em uma coleção de sequências de entrada com eventos. Cada sequência de entrada possui um identificador diferente e cada evento em uma sequência de entrada também possui um identificador exclusivo nessa sequência de entrada. O parâmetro que define os valores mínimos que uma sequência deve ocorrer no conjunto de dados para ser considerada frequente é o $support$, que foi configurado dividindo β, usado no G-STSM, pelo número de STS do conjunto de dados usado D, *i.e.*, $support \cong \frac{\beta}{|sts(D)|}$.

O DBSCAN recebe uma matriz com as posições das sequências frequentes e os parâmetros de entrada eps e $minPts$, e gera grupos para as ocorrências de cada sequência frequente. O parâmetro eps é a distância máxima entre dois pontos para que um seja considerado na vizinhança do outro, e o parâmetro $minPts$ é o número de pontos em uma vizinhança para que este seja considerado um ponto central. Pelo menos um ponto deve ter pelo menos $minPts$ com uma distância menor ou igual a eps em cada *cluster*. O eps foi configurado com o mesmo valor que σ usado pelo G-STSM, *i.e.*, $eps \cong \sigma$, e o número de pontos mínimos $minPts$ na vizinhança foi configurado com o mesmo valor de β usado pelo G-STSM, *i.e.*, $minPts \cong \beta$. A configuração dos parâmetros usados nos experimentos pode ser observada na Tabela 17.

Tabela 17 – Parâmetros usados nos experimentos.

cenário	variação	G-STSM			SPADE+DBSCAN		
		γ	β	σ	$support$	eps	$minPts$
A	1, 2, 3, 4, 5	0,6	10	5	$10/\|sts(D)\|$	5	10
B	4	0,6; 0,8; 1.0	[5,10]	[5,10]	[5,10]/951	[5,10]	[5,10]
C	1, 2, 3, 4, 5	0,8	5, 10	5, 10	-	-	-

Os experimentos foram conduzidos em um computador com processador Core i7-7820X com 16 *cores*, 128GB de RAM e sistema operacional Ubuntu 20.04 LTS. Ambas as abordagens, G-STSM e SPADE+DBSCAN, foram executadas em ambiente R (versão 3.6.3) [R Core Team, 2020]. R é um ambiente integrado de *software* disponível para uma grande variedade de plataformas que permite a manipulação de dados, cálculos, programação, etc. Pode ser estendido através de pacotes disponibilizados pela comunidade de usuários em seu repositório CRAN [2019]. *The Comprehensive R Archive Network* (CRAN) é uma rede de servidores FTP e *web* em todo o mundo que armazena versões idênticas e atualizadas de código e documentação para R. A escolha da linguagem R para implementação e teste do algoritmo G-STSM tem como objetivo disponibilizar à comunidade científica no CRAN, uma ferramenta capaz de apoiar outras pesquisas, nas quais a mineração de sequências restritas no espaço e no tempo seja um meio de se chegar aos resultados desejados. Foram utilizadas a implementação $cSPADE$ do algoritmo SPADE disponível no pacote *arulesSequences* [Buchta et al., 2020] e a implementação do algoritmo DBSCAN do pacote *dbscan* [Hahsler et al., 2019].

5.4- Análise Comparativa

A Figura 8 mostra o tempo total de execução das abordagens G-STSM e SPADE+DBSCAN usando todas as variações do tamanho do conjunto de dados (cenário A da Tabela 17). A abordagem G-STSM apresenta o melhor tempo de execução para todas as variações. Em geral, o G-STSM é mais de duas vezes mais rápido, exceto para a variação 4 com oito quadrantes. Observe que para o conjunto de dados completo, dezesseis quadrantes, o G-STSM foi ainda melhor, sendo mais de três vezes mais rápido.

Ambas as abordagens também foram avaliadas usando métricas de classificação para o cenário B da Tabela 17. A partir dos resultados, foi possível obter a frequência mediana para o SPADE+DBSCAN de aproximadamente 0,657. Assim, para ter uma comparação justa, todos os resultados daqui até o fim desta seção usam $\gamma = 0,6$ para a abordagem G-STSM.

A Figura 9 apresenta os resultados de $acurácia$ para diferentes combinações de σ e β, e diferentes tamanhos de sequência (2, 3, 4 e 5). A cor cinza representa

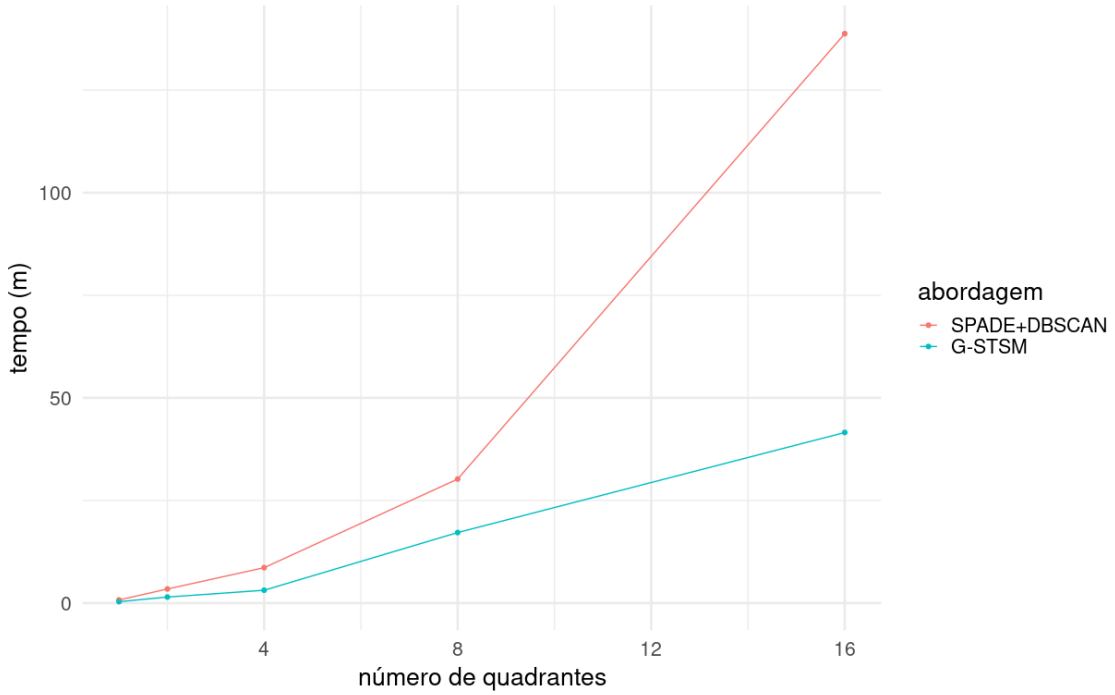


Figura 8 – Tempo total de execução para cada abordagem usando diferentes números de quadrantes.

a ausência de valor, o que significa que para os valores de entrada utilizados (para β e σ) nenhuma sequência desse tamanho foi encontrada. Observe que ambas as abordagens apresentaram altos valores para $acurácia$. Os valores de β mais altos e σ mais baixos fornecem maior acurácia, significando que uma configuração mais "densa" (mais ocorrências com menos distância entre as ocorrências) é melhor.

No entanto, uma configuração mais "densa" é mais restritiva, pois requer um maior número de ocorrências com maior proximidade, resultando em um menor número total de ocorrências. Como consequência direta do menor número de ocorrências devido a uma configuração mais "densa" ou a um tamanho de sequência maior, obtemos um número menor de FP e um número maior de TN. Como a $acurácia$ é diretamente proporcional a TN, se este aumentar, a $acurácia$ também aumenta.

A Tabela 18 apresenta a média de $acurácia$, $precisão$, $sensibilidade$, e f_1 para ambas as abordagens. Em geral, as abordagens obtiveram resultados semelhantes, apresentando diferença apenas a partir da segunda casa decimal e com pequeno desvio padrão.

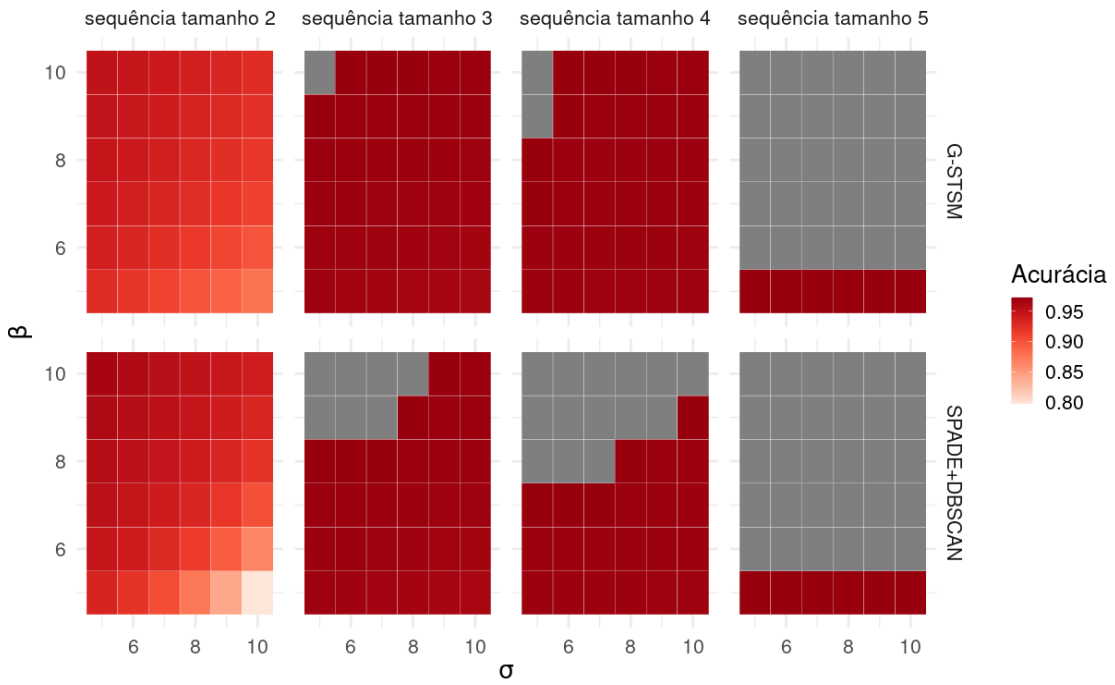


Figura 9 – $acurácia$ dados β, σ e tamanho da sequência para ambas as abordagens. Áreas em cinza representam ausência de valor, ou seja, para os parâmetros de entrada não foram encontradas sequências frequentes.

Tabela 18 – Média e desvio padrão das métricas qualitativas para ambas abordagens.

	G-STSM	**SPADE+DBSCAN**
$acurácia$	$0,96 \pm 0,02$	$0,95 \pm 0,03$
$precisão$	$0,05 \pm 0,04$	$0,06 \pm 0,04$
$sensibilidade$	$0,04 \pm 0,06$	$0,04 \pm 0,06$
f_1	$0,04 \pm 0,04$	$0,04 \pm 0,03$

5.5- Análise de Sensibilidade

Esta seção tem por intuito apresentar um análise do G-STSM a partir da variação dos seus parâmetros de entrada. As Figuras 10 e 11 são usadas para apoiar este estudo. Elas usam o cenário B da Tabela 17. A Figura 10 apresenta a correlação entre os parâmetros de entrada (γ, β e σ) e as informações de saída: o número de SRG, o número de ocorrências, o uso de memória e o tempo de execução do G-STSM.

Na Figura 10, é possível observar que um valor menor de γ (frequência mínima) permite o G-STSM localizar mais ocorrências, desta forma ele usa mais memória devido à quantidade maior de dados e gasta mais tempo para lidar com tais dados. No entanto,

quanto mais baixo for γ, menos SRG teremos, dado que com uma frequência menor aumenta a possibilidade de unir SRG próximos. Com um β menor (tamanho mínimo do grupo), geram-se mais grupos e ocorrências, e gasta-se mais memória e tempo. Para valores maiores de σ (distância máxima de pelo menos uma outra posição do grupo), o número de ocorrências e SRG também são maiores, o uso de memória é menor e a execução é mais demorada.

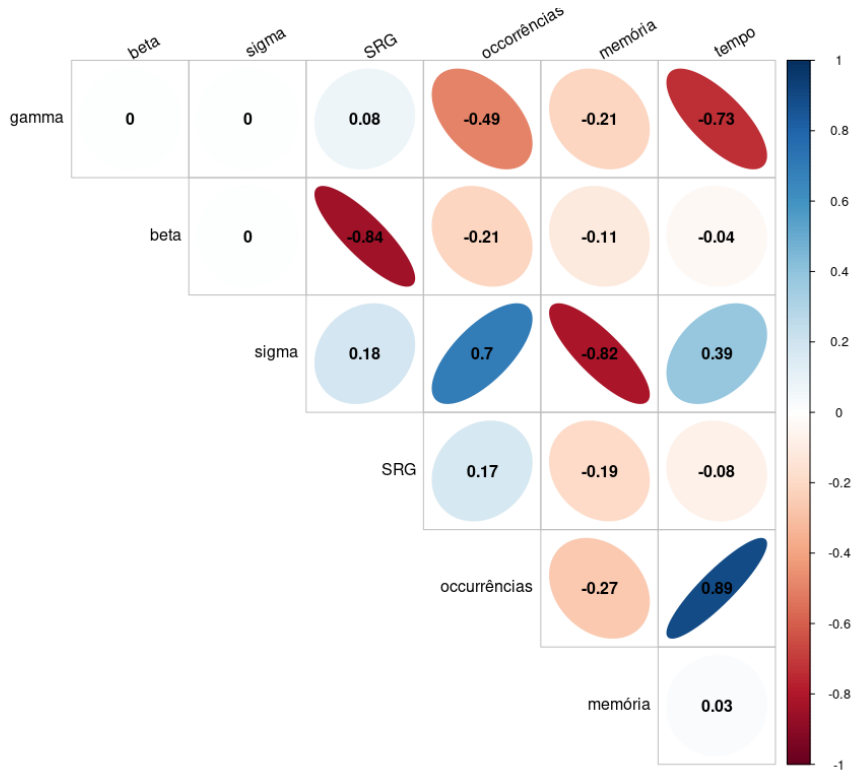


Figura 10 – Correlação entre os parâmetros de entrada (γ, β e σ), o uso de recursos, e os resultados.

A Figura 11 mostra os valores medianos de $acurácia$, $precisão$, $sensibilidade$, e f_1 dado o tamanho da sequência. Conforme o tamanho da sequência aumenta, o número de ocorrências diminui para menos da metade. Assim, os valores de TN sobem na mesma proporção enquanto TP desce um pouco (há poucos padrões previamente marcados). Desta forma, os valores de $acurácia$ aumentam.

As Figuras 12 e 13 usam o cenário C da Tabela 17. Eles mostram o tempo de execução e o uso máximo de memória, respectivamente, com o aumento do tamanho do conjunto de dados e diferentes configurações. Para ambos os gráficos, foram utilizadas três configurações diferentes, detalhadas na Tabela 19.

A configuração $HBLS$ (do inglês *High Beta Low Sigma*) é a mais restritiva, en-

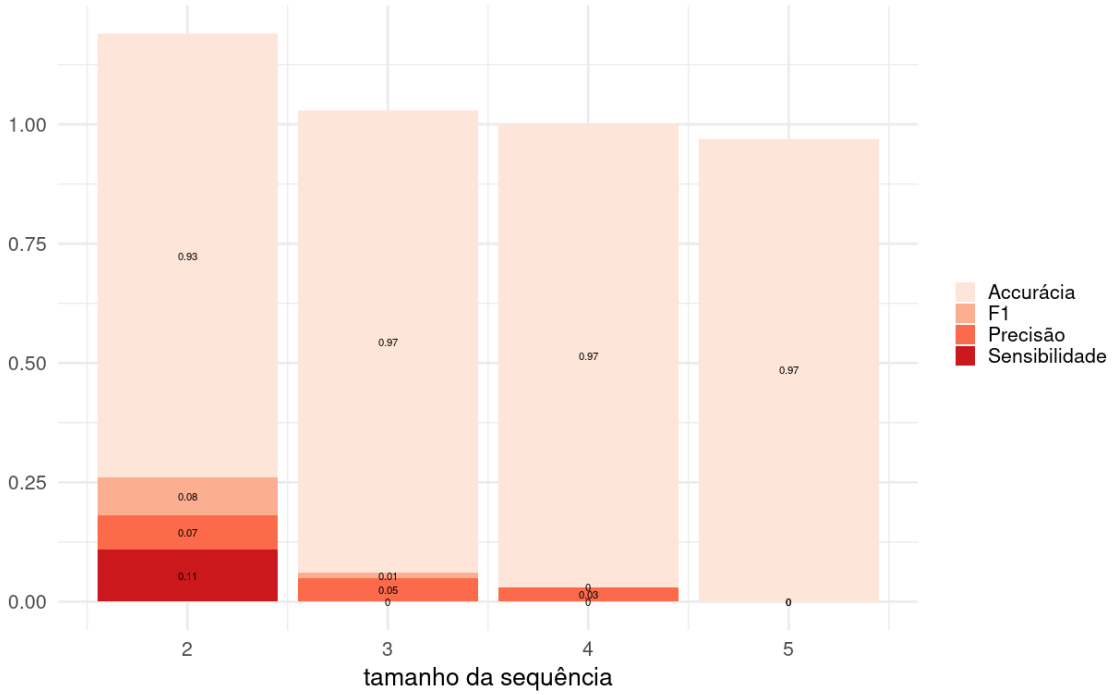


Figura 11 – Mediana de $acurácia$, $precisão$, $sensibilidade$, e f_1 dado o tamanho da sequência para o G-STSM.

contra apenas SRG mais “densos”, com um maior número de elementos por grupo e menor distância mínima dos elementos do grupo. Por outro lado, a configuração $LBHS$ (do inglês *Low Beta High Sigma*) é a menos restritiva, permitindo poucos elementos em um grupo com grande distância uns dos outros. Finalmente, a configuração $HBHS$ (do inglês *High Beta High Sigma*) tem um grande número de elementos por grupo, mas permite grande distância entre seus elementos. Para cada configuração, foram usados diferentes tamanhos de conjuntos de dados, começando de um quadrante duplicando seu tamanho até utilizar todo o conjunto de dados T401, *i.e.*, 1, 2, 4, 8 e 16 quadrantes, todas as variações da Tabela 16.

Tabela 19 – Configuração utilizada no cenário C.

configuração	γ	β	σ
HBLS	0,8	10	5
HBHS	0,8	10	10
LBHS	0,8	5	10

Conforme esperado, aumentando o tamanho do conjunto de dados, o uso de recursos aumenta. A maior diferença de tempo ocorre com todo o conjunto de dados (dezesseis quadrantes), uma diferença de 3,9 minutos, que corresponde a 9,82% de

aumento de tempo das configurações $HBHS$ para $HBLS$. Para o uso de memória, a maior diferença também ocorre ao usar todo o conjunto de dados, 340,6 MB, que corresponde a 10,55% de aumento no uso de memória. Esse comportamento indica que as configurações dos parâmetros de entrada não fazem grande diferença no desempenho ou no uso de recursos para o algoritmo G-STSM.

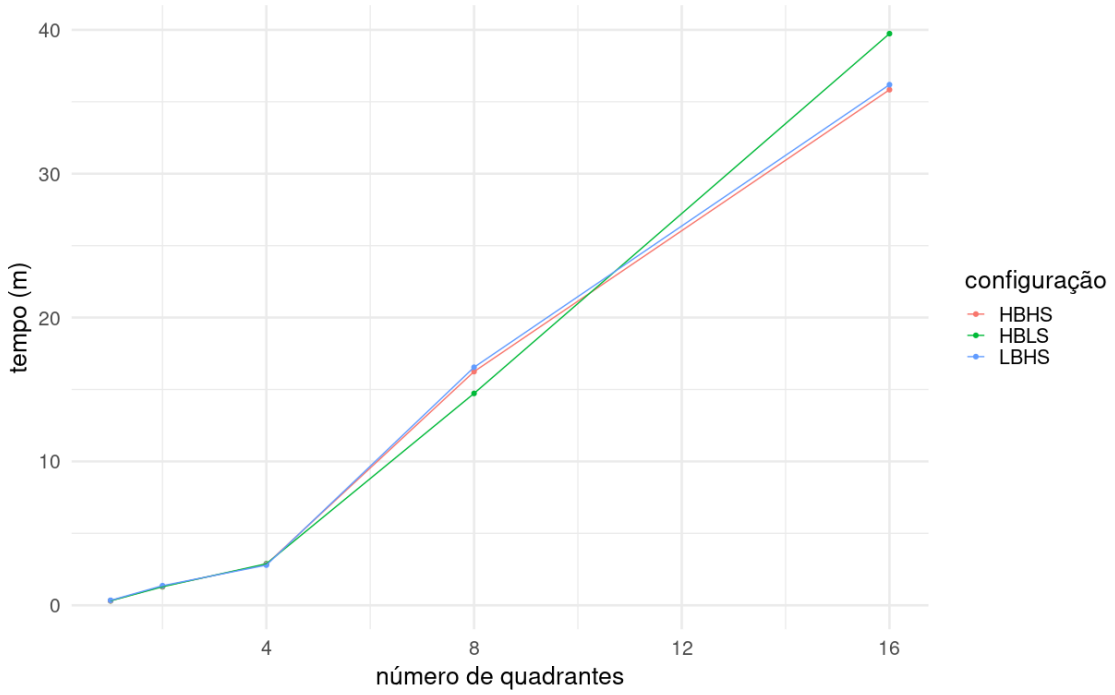


Figura 12 – Tempo de execução do G-STSM usando diferentes configurações e tamanhos de conjunto de dados.

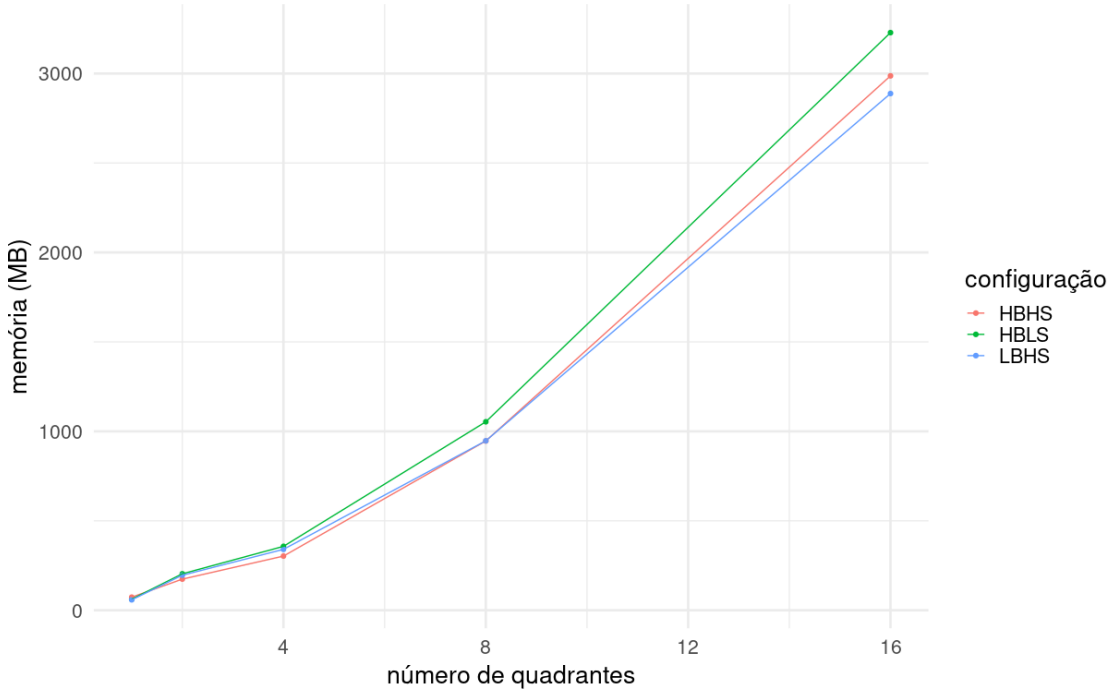


Figura 13 – Uso máximo de memória durante a execução do G-STSM para diferentes tipos de configurações e tamanhos de conjunto de dados.

6- Conclusões

Este trabalho abordou o problema de mineração de sequências restritas no espaço e no tempo. Este problema surge a partir da necessidade de extrair padrões significativos e interessantes de conjuntos de dados com marcações de espaço e tempo e, também, da observação de que nem sempre alguns padrões são frequentes por todo o conjunto de dados, mas sim restritos a um conjunto de posições espaciais e a um determinado período de tempo. Vale ressaltar que tais padrões apresentam baixo suporte se considerado todo o conjunto de dados no qual se inserem.

Foi realizada uma revisão da literatura relacionada ao tema, onde foram apresentados trabalhos que utilizam conjuntos de dados de trajetórias e de posições fixas. A partir dos trabalhos da classe de posições fixas, na qual o presente trabalho se enquadra, constatou-se que muitos trabalhos limitam previamente espaço e/ou tempo para encontrar os padrões restritos. Observou-se, também, que algoritmos de mineração de sequências e de agrupamento poderiam ser combinados para resolver o problema. Apresentou-se, assim, uma abordagem intuitiva para encontrar sequencias restritas no espaço e no tempo. Esta abordagem, nomeada SPADE+DBSCAN, utilizou o algoritmo SPADE para minerar sequências com suporte bastante baixo e o algoritmo DBSCAN para agrupar as ocorrências de cada sequência frequente.

Fundamentos importantes para o processo de mineração de sequências restritas no espaço e no tempo foram abordados e as noções de grupo, RG, KRG e SRG foram introduzidas. Baseado nestes conceitos, o algoritmo G-STSM foi proposto para o problema de descoberta de sequências restritas no espaço e no tempo. Ele encontra as sequências frequentes, a região no espaço e o período de tempo em que são frequentes, sendo capaz de encontrar diferentes tamanhos de sequências, intervalos de tempo e regiões do espaço onde uma sequência é frequente. A fim de facilitar a compreensão, um exemplo para ilustrar o funcionamento do algoritmo também foi apresentado. Até onde se sabe, este é o primeiro algoritmo capaz de encontrar sequências restritas no espaço e no tempo que funciona com uma dimensão de tempo e três dimensões de espaço.

Com o intuito de comparar o G-STSM com a abordagem SPADE+DBSCAN, foram realizados experimentos usando um conjunto de dados sísmicos real, o *inline* T401. A

existência de padrões previamente marcados por especialistas permitiu o uso de métricas de classificação: $acurácia$, $precisão$, $sensibilidade$, e f_1. Para avaliar o uso de recursos com diferentes tamanhos de conjuntos de dados e diferentes parâmetros de entrada, o conjunto de dados T401 foi dividido em quadrantes. Dessa maneira, foi possível avaliar não só a qualidade das abordagens com as métricas de recuperação da informação, mas também o uso de memória e tempo utilizado para concluir o processamento.

Ambas as abordagens obtiveram resultados qualitativos semelhantes com diferenças a partir da segunda casa decimal e com pequeno desvio padrão. Para diferentes tamanhos de conjuntos de dados, o G-STSM obteve desempenho sempre superior em relação à abordagem SPADE+DBSCAN, chegando a ser mais de três vezes mais rápido.

Além da análise comparativa, o comportamento do G-STSM foi detalhado com diferentes configurações de parâmetros e tamanhos de conjuntos de dados em uma análise de sensibilidade, a qual possibilitou observar as alterações no seu comportamento dadas mudanças nos dados de entrada. De maneira geral, a alteração de parâmetros de entrada causou poucas alterações no uso de recursos do G-STSM.

Embora o G-STSM tenha sido avaliado usando o domínio sísmico, cujas posições são ordenadas linearmente, ele foi projetado para ser uma abordagem flexível e útil para extrair conhecimento valioso de diferentes conjuntos de dados espaço-temporais. Assim, o G-STSM provou ser uma ferramenta de mineração de dados eficiente para encontrar sequências restritas no espaço e no tempo.

Dessa forma, como trabalho futuro, propõe-se o uso de conjuntos de dados de outros domínios nos quais as posições estejam distribuídas no espaço de forma não linear. Além disso, pretende-se disponibilizar o G-STSM como ferramenta em um pacote R de maneira que este esteja disponível para ser utilizado pela comunidade em pesquisas onde a mineração de sequências restritas no espaço e no tempo seja uma ferramenta útil.

Referências Bibliográficas

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.

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(P2):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.

Aydin, B. and Angryk, R. A. (2016). Spatiotemporal event sequence mining from evolving regions. volume 0, pages 4172–4177.

Aydin, B., Boubrahimi, S. F., Kucuk, A., Nezamdoust, B., and Angryk, R. A. (2020). Spatiotemporal event sequence discovery without thresholds. *Geoinformatica*, pages 1–29.

Batu, B., Temizel, T., and Duzgun, H. (2017). A non-parametric algorithm for discovering triggering patterns of spatio-temporal event types. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2629–2642.

Birant, D. and Kut, A. (2007). St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221.

Buchta, C., Hahsler, M., and with contributions from Daniel Diaz (2020). *arulesSequences: Mining Frequent Sequences*. R package version 0.2-25.

Campisano, R., Borges, H., Porto, F., Perosi, F., Pacitti, E., Masseglia, F., and Ogasawara, E. (2018). Discovering tight space-time sequences. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 247–257. Springer.

Chen, B.-H., Chuang, A.-W., and Chuang, K.-T. (2015). Discovery of spatiotemporal chaining patterns. volume 07-09-Ocobert-2015.

Chen, B.-H., Teng, S.-Y., and Chuang, K.-T. (2017). Mining spatio-temporal chaining patterns in non-identity event databases. *Intelligent Data Analysis*, 21(S1):S71–S102.

Chen, H., Yu, S., Huang, F., Zhu, B., Gao, L., and Qian, C. (2020). Spatio-temporal analysis of retail customer behavior based on clustering and sequential pattern mining. pages 284–288.

Chen, X., Pang, J., and Xue, R. (2014). Constructing and comparing user mobility profiles. *ACM Transactions on the Web*, 8(4).

Chen, Y.-L. and Hu, Y.-H. (2006). Constraint-based sequential pattern mining: The consideration of recency and compactness. *Decision Support Systems*, 42(2):1203–1215.

Cheng, L., Yang, X., Tang, L., Duan, Q., Kan, Z., Zhang, X., and Ye, X. (2020). Spatiotemporal analysis of taxi-driver shifts using big trace data. *ISPRS International Journal of Geo-Information*, 9(4).

CRAN (2019). The comprehensive r archive network - cran. `https://cran.r-project.org/`.

(DAL), D. A. L. (2020). Generalized Discovery of Tight Space-Time Sequences | Data Analytics Lab.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. (1996). Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88.

Feuerhake, U. and Sester, M. (2013). Mining group movement patterns. pages 510–513.

Flamand, C., Fabregue, M., Bringay, S., Ardillon, V., Quénel, P., Desenclos, J., and Teisseire, M. (2014). Mining local climate data to assess spatiotemporal dengue fever epidemic patterns in french guiana. *Journal of the American Medical Informatics Association : JAMIA*, 21(e2):e232–240.

Geng, L. and Hamilton, H. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3):3.

Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339.

Gurram, S. and Rama Mohan Reddy, A. (2014). A grid-based algorithm for mining spatio-temporal sequential patterns. *International Review on Computers and Software*, 9(4):659–666.

Hahsler, M., Piekenbrock, M., and Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, 15(1):55–86.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE Washington, DC, USA.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM.

Hand, D. J. (2007). Principles of data mining. *Drug safety*, 30(7):621–622.

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

He, Z., Deng, M., Cai, J., Xie, Z., Guan, Q., and Yang, C. (2020). Mining spatiotemporal association patterns from complex geographic phenomena. *International Journal of Geographical Information Science*, 34(6):1162–1187.

Huang, Q., Li, Z., Li, J., and Chang, C. (2016). Mining frequent trajectory patterns from online footprints.

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

Ibrahim, R. and Shafiq, M. (2019). Detecting taxi movements using random swap clustering and sequential pattern mining. *Journal of Big Data*, 6(1).

Julea, A., Méger, N., Bolon, P., Rigotti, C., Doin, M.-P., Lasserre, C., Trouvé, E., and Lăzărescu, 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.

Julea, A., Méger, N., Trouvé, E., and Bolon, P. (2008). On extracting evolutions from satellite image time series. volume 5, pages V228–V231.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. I. (1994). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the third international conference on Information and knowledge management*, pages 401–407. ACM.

Koseoglu, B., Kaya, E., Balcisoy, S., and Bozkaya, B. (2020). St sequence miner: visualization and mining of spatio-temporal event sequences. *Visual Computer*, 36(10-12):2369–2381.

Lee, S., Lim, J., Park, J., and Kim, K. (2016). Next place prediction based on spatiotemporal pattern mining of mobile device logs. *Sensors (Switzerland)*, 16(2).

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

Li, K. and Fu, Y. (2014). Prediction of Human Activity by Discovering Temporal Sequence Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1644–1657.

Mooney, C. H. and Roddick, J. F. (2013). Sequential pattern mining–approaches and algorithms. *ACM Computing Surveys (CSUR)*, 45(2):19.

OpendTect (2020). dgb earth sciences - innovative seismic interpretations solutions.

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

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roiger, R. J. (2017). *Data mining: a tutorial-based primer*. CRC Press.

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

Saneifar, H., Bringay, S., Laurent, A., and Teisseire, M. (2008). S2mp: Similarity measure for sequential patterns. In *AusDM'08: The Australasian Data Mining Conference*, volume 87, pages 095–104. ACS.

Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer. Book.

Sukanya, N. and Ranjit Jeba Thangaiah, P. (2019). Review on frequent patterns in data mining applications. *Journal of Advanced Research in Dynamical and Control Systems*, 11(4 Special Issue):1725–1730.

Sunitha, G. and Rama Mohan Reddy, A. (2014). Mining frequent patterns from spatiotemporal data sets: A survey. *Journal of Theoretical and Applied Information Technology*, 68(2):265–274.

Sunitha, G. and Rama Mohan Reddy, A. (2016). Wrsp-miner algorithm for mining weighted sequential patterns from spatio-temporal databases. *Advances in Intelligent Systems and Computing*, 379:309–317.

Tsai, C.-Y. and Shieh, Y.-C. (2009). A change detection method for sequential patterns. *Decision Support Systems*, 46(2):501–511.

Tsoukatos, I. and Gunopulos, D. (2001). Efficient mining of spatiotemporal patterns. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2121:425–442.

Wan, H., Guo, S., Yin, K., Liang, X., and Lin, Y. (2019). Cts-lstm: Lstm-based neural networks for correlatedtime series prediction. *Knowledge-Based Systems*.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xu, L. and Kwan, M.-P. (2020). Mining sequential activity–travel patterns for individual-level human activity prediction using bayesian networks. *Transactions in GIS*, 24(5):1341–1358.

Xue, C., Liu, J., Li, X., and Dong, Q. (2016). Normalized-mutual-information-based mining method for cascading patterns. *ISPRS International Journal of Geo-Information*, 5(10).

Yang, C. and Gidófalvi, G. (2018). Mining and visual exploration of closed contiguous sequential patterns in trajectories. *International Journal of Geographical Information Science*, 32(7):1282–1304.

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. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60.

Zaki, M. J. (2000). Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429.

Zhang, Z.-H., Zhai, W.-J., Shen, R.-P., Zeng, S.-C., and Min, F. (2018). State transition pattern with periodic wildcard gaps. pages 440–447.