

Discovering frequent parallel episodes in complex event sequences by counting distinct occurrences

Seminário 1 – Aprendizado Descritivo
2025.1

Historiador

Alexis Duarte Guimarães Mariz

Amanda Mendes Pinho

João Vítor Fernandes Dias

Kael Soares Augusto

Lucas Xavier Veneroso

Mineração de Sequências vs. Episódios

- GSP: usa sequências frequentes de tamanho $k-1$ para gerar candidatas de tamanho k e avaliar o suporte
- Spade: novas sequências podem ser geradas a partir da junção temporal de duas sequências que pertençam a uma mesma classe de equivalência (compartilham um prefixo de tamanho $k-1$)

	Sequências	Episódios
Dados	Cada usuário/cliente gera uma sequência	Sequência única de eventos temporais
Ordenação	Totalmente ordenados. Ordem total	Eventos próximos no tempo/janelas temporais. Ordem total, parcial ou nenhuma
Timestamp	Não é necessário	Sim
Anti-monotonicidade	Sim	Sim

- Exemplos de Mineração de Episódios:
 - Detecção de padrões que antecedem uma falha grave em sistemas de telecomunicações
 - Monitoramento de redes elétricas para detectar episódios que precedem quedas ou sobrecargas

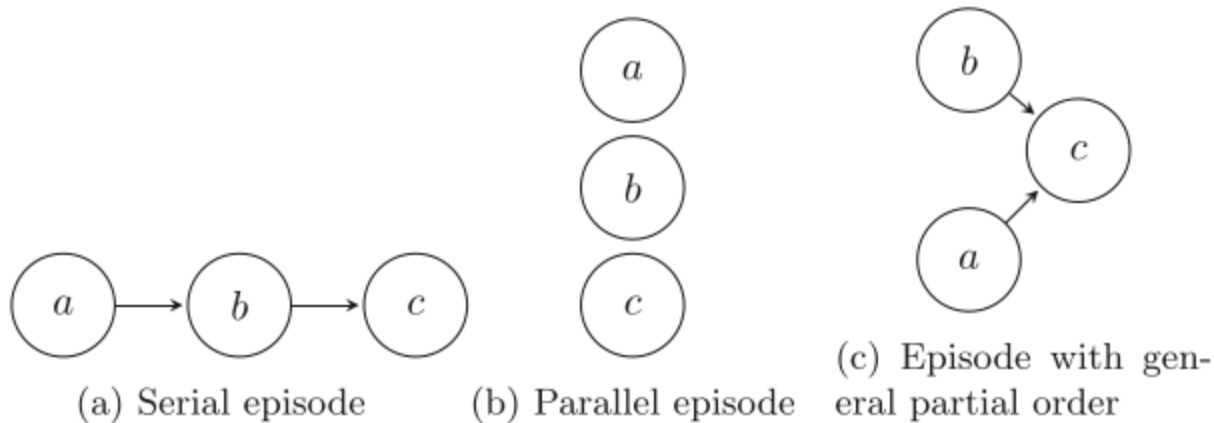
Mineração de Episódios Frequentes

- Eventos com timestamp
 - Séries temporais: sazonalidade e tendência
 - Sequências de eventos: ordem e co-ocorrência de eventos
- Episódios paralelos são padrões com ordem livre
- Sequências complexas podem ter mais de um evento por timestamp

- Soluções baseadas em: geração de episódios maximais, episódios fechados e episódios geradores
- Contribuições do artigo:
 - Nova definição de frequência
 - Algoritmos EMDO e EMDO-P

Problemas: limitações

- Algoritmos clássicos lidam com episódios em série
- Proposta: episódios paralelos



- A maioria foca em sequências simples de eventos(apenas 1 episódio por timestamp)
- Proposta: sequências complexas

Problemas: definição de frequência

- Superestimação: $(1,4), (1,5), (3,4), (3,5)$

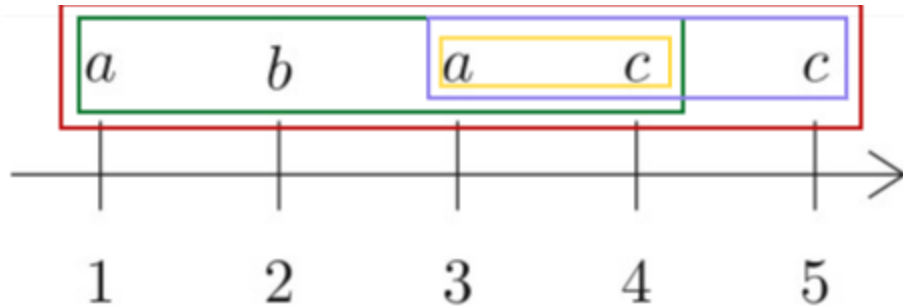


Fig. 1 A simple event sequence with five events and five timestamps

- Subestimação: $(1,4)$

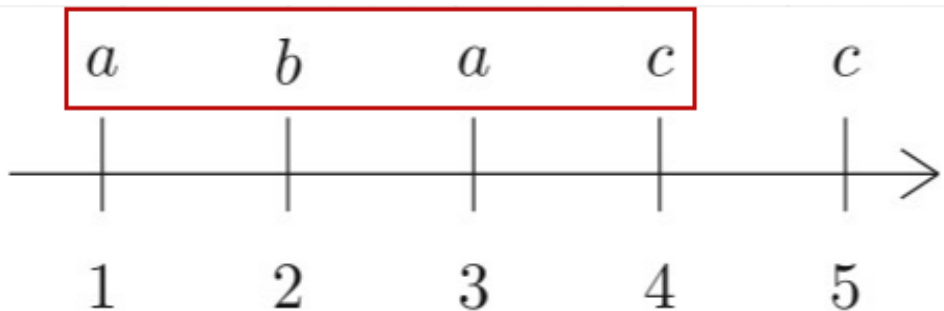


Fig. 1 A simple event sequence with five events and five timestamps

- Proposta do artigo baseada ocorrência: $(1,4), (3,5)$

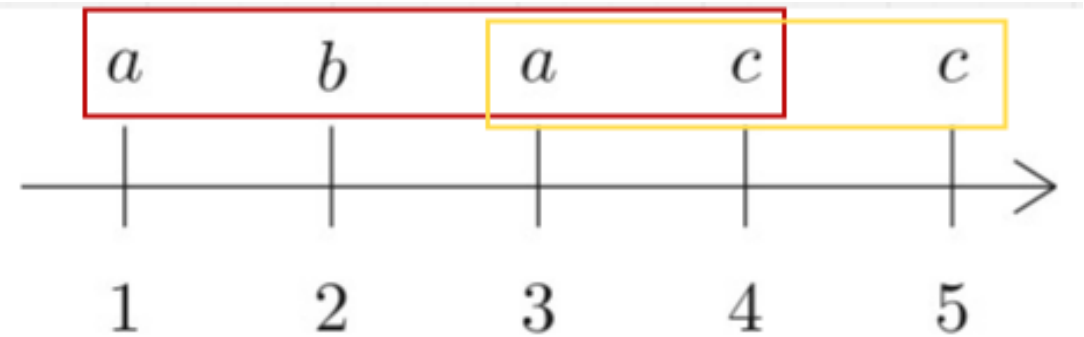


Fig. 1 A simple event sequence with five events and five timestamps

Formalizador

Henrique Rotsen

Alexandre Cassimiro Silva Araújo

Wesley Marques Daniel Chaves

Caíque Bruno Fortunato

Victor Gabriel Moura Oliveira

Arthur Yochio R. Codama

Conceitos Fundamentais

1. Evento

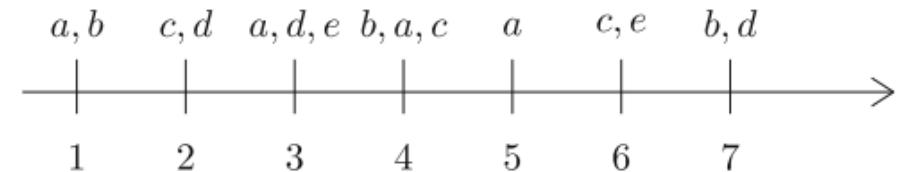
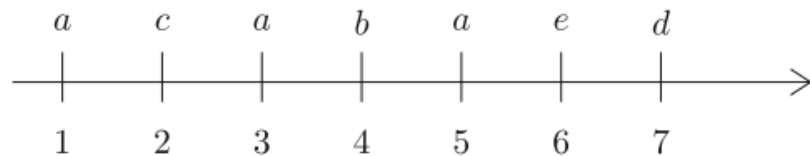
- Par (tipo de evento, timestamp)

2. Sequência de Eventos Simples

- $S = \langle (e_1, t_1), (e_2, t_2), \dots, (e_n, t_n) \rangle$ é um conjunto ordenado de eventos (e_i, t_i) ou seja, para quaisquer inteiros i, j , se $i < j$ então, $t_i < t_j$.

3. Sequência de Eventos Complexa

- $S = \langle (\epsilon_1, t_1), (\epsilon_2, t_2), \dots, (\epsilon_n, t_n) \rangle$ é um conjunto ordenado de pares (ϵ_i, t_i) , onde ϵ_i é um *conjunto* de tipos de evento.

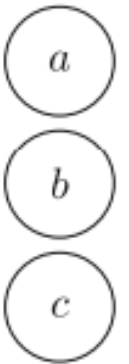


O Padrão Minerado: Episódios

Episódio: É um conjunto de nós (eventos) com uma relação de ordem parcial entre eles. O artigo foca em um tipo específico, os Episódios Paralelos Injetivos

- **Paralelo:** A ordem em que os eventos do episódio ocorrem não importa.
- **Injetivo:** Cada tipo de evento aparece no máximo uma vez dentro do episódio.

Por exemplo, $\alpha = \{A, B, C\}$ é injetivo, mas $\alpha = \{A, B, A\}$ não seria



(b) Parallel episode

Contagem por Ocorrências Distintas

Conceito chave que ataca os 2 grandes problemas: Superestimação e Subestimação

Ocorrência de um Episódio: Vetor de timestamps, ou seja, um episódio $\alpha = \{A, B\}$, uma ocorrência poderia ser $h = [t_A, t_B]$, onde t_A é o timestamp de A e t_B é o de B.

Ocorrência Distintas: Duas ocorrências são consideradas distintas se não há timestamp em comum.

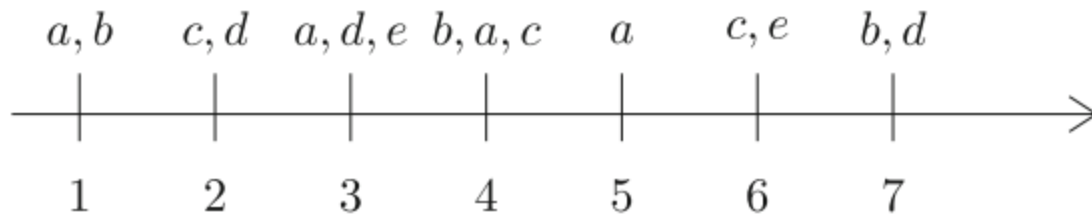
Ex: $\langle (A,1), (B,2), (A,3), (C,4) \rangle$ e o episódio $\alpha = \{A,C\}$

A ideia é encontrar o maior conjunto possível de ocorrências distintas

Suporte: É o tamanho do conjunto máximo de suas ocorrências distintas. Um episódio é frequente sse $\text{suporte}(\alpha) \geq \text{minsup}$

Exemplo

Dada a seguinte sequência complexa de eventos



Para o episódio $\alpha = abc$, temos o seguinte conjunto maximal de ocorrências distintas para α :

$$do(\alpha) = \{[1 \ 1 \ 2], [3 \ 4 \ 4], [5 \ 7 \ 6]\}$$

Logo, $suporte(\alpha) = 3$

Regras de episódio

É uma implicação da forma $\alpha \Rightarrow \beta$, onde α (antecedente) e β (consequente) são episódios frequentes.

"Quando observamos a ocorrência do padrão α , há uma forte probabilidade de que o padrão β ocorra logo em seguida."

- Ocorrência Válida de uma Regra
 - Se α começa antes de β começar e α termina antes de β terminar.
- Confiança: Medida de força da regra, é a probabilidade condicional de β ocorrer dado que α ocorreu.

$$conf(\alpha \Rightarrow \beta) = \frac{\|occER(\alpha \Rightarrow \beta)\|}{support(\alpha)}$$

- Uma regra é considerada válida se sua confiança for maior que um limiar mínimo (min_{conf}).

EMDO

1. Encontrar os Blocos de Construção

- Encontrar todos os episódios de tamanho 1 (eventos individuais) que são frequentes.
 - Varre a sequência de dados uma única vez para mapear e contar todas as ocorrências de cada tipo de evento individual
 - Calcula o suporte para cada evento
 - Filtra essa lista

2. Crescimento dos Episódios

- Combinar episódios frequentes e gerar candidatos maiores.

3. Verificação e Poda (Pruning)

- Para cada candidato gerado, o algoritmo calcula seu suporte usando a contagem de **ocorrências distintas**.
- Propriedade do "Apriori"

EMDO-P

1. Obter os Ingredientes

- Executar o EMDO para ter a lista de todos os episódios frequentes.

2. Gerar e Testar Regras

- Para cada par de episódios frequentes (α , β), o algoritmo poderia testar a regra $\alpha \Rightarrow \beta$
- **Proposição de Poda:** Se a regra $\alpha \Rightarrow \beta$ não é válida, então **qualquer regra** da forma $\alpha \Rightarrow \gamma$, onde γ é um super-episódio de β , também será inválida e não precisa ser testada.
- **Intuição:** Se a ocorrência de α já não é um bom indicador da ocorrência de β , ela certamente será um indicador pior ainda de um evento mais específico e raro como γ (que contém β e mais outros eventos)

Metodologista




Grupo: ENILDA ALVES COELHO
FÁBIO CÉSAR MARRA FILHO
GABRIEL TONIONI DUARTE
IASMIN CORREA ARAUJO

JOSE VINICIUS DE LIMA MASSARICO
LARISSA DUARTE SANTANA
MARCELO LOMMEZ RODRIGUES DE JESUS

Contextualização e Objetivo

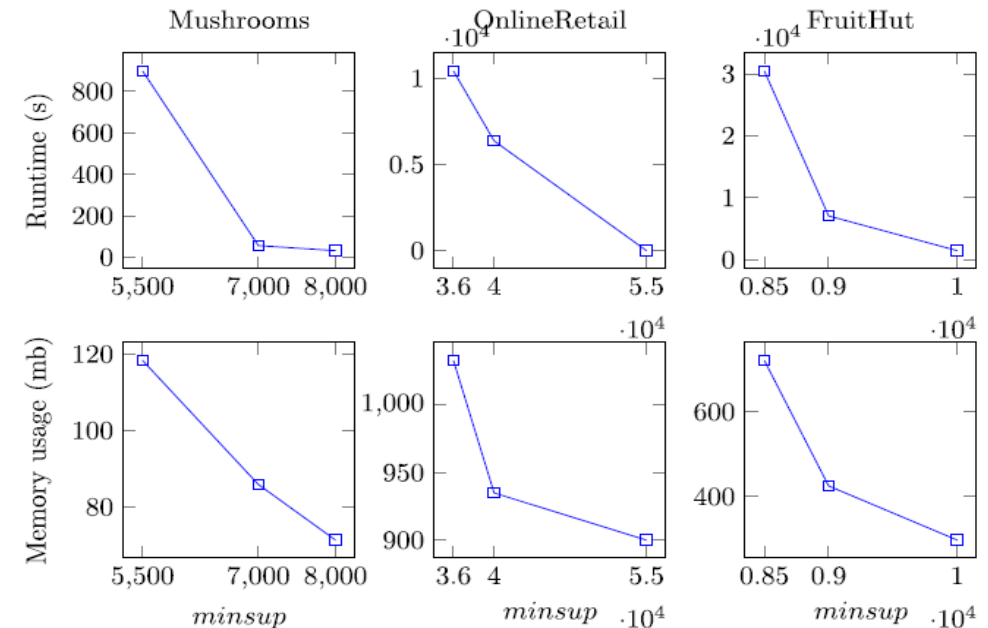
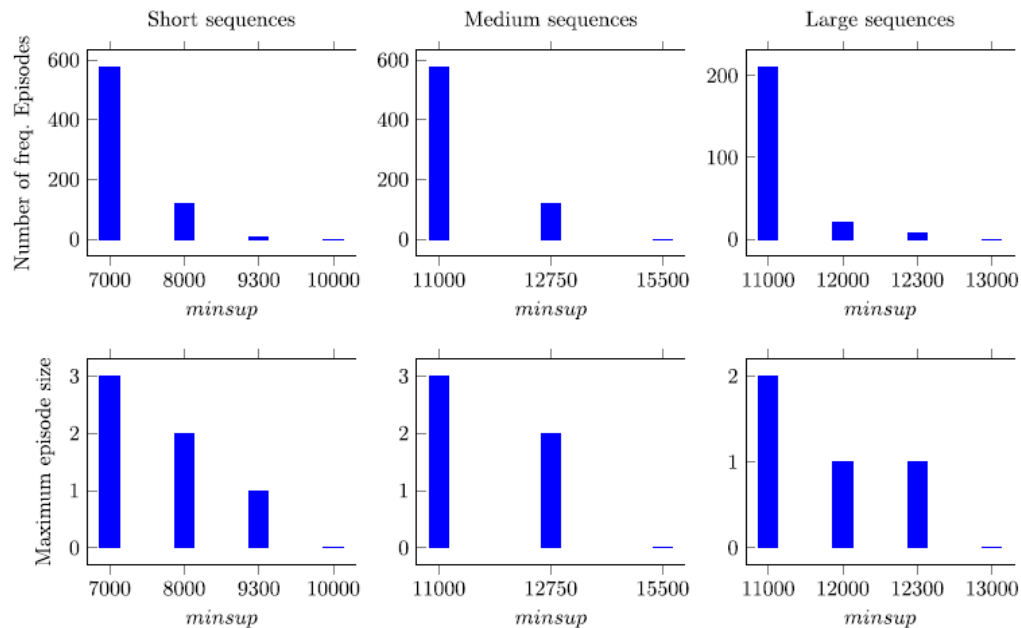
- **Problema:** Descobrir padrões frequentes em **sequências de eventos complexas**, onde múltiplos eventos podem ocorrer no mesmo timestamp.
- A maioria dos algoritmos existentes:
 - Foca em episódios **seriais**.
 - Usa definições de frequência com **sobreposição** ou **restrições excessivas**.
- Este trabalho **propõe**:
 - Uma nova abordagem baseada em **ocorrências distintas**.
 - **Dois algoritmos**:
 - **EMDO** (Algoritmo Ingênuo)
 - **EMDO_P** (Uso da propriedade de anti-monotonicidade)
- **Objetivo dos experimentos:** avaliar a eficiência e a qualidade dos padrões gerados pelos algoritmos propostos.

Datasets e Estratégia Experimental

- Todos os testes foram realizados utilizando dados sintéticos e reais para validar eficiência e qualidade das descobertas.
 - **Conjuntos Sintéticos**
 - Gerados com base em:
 - N° de eventos por sequência
 - N° de tipos de eventos
 - N° máximo de eventos por timestamp
 - Três tamanhos: **curto, médio e longo**
 - **Conjuntos Reais (SPMF)**
 -  OnlineRetail: 541.880 eventos, 2.603 tipos
 -  FruitHut: 181.970 eventos, 9.390 tipos
 -  Mushrooms: 8.416 eventos, 119 tipos

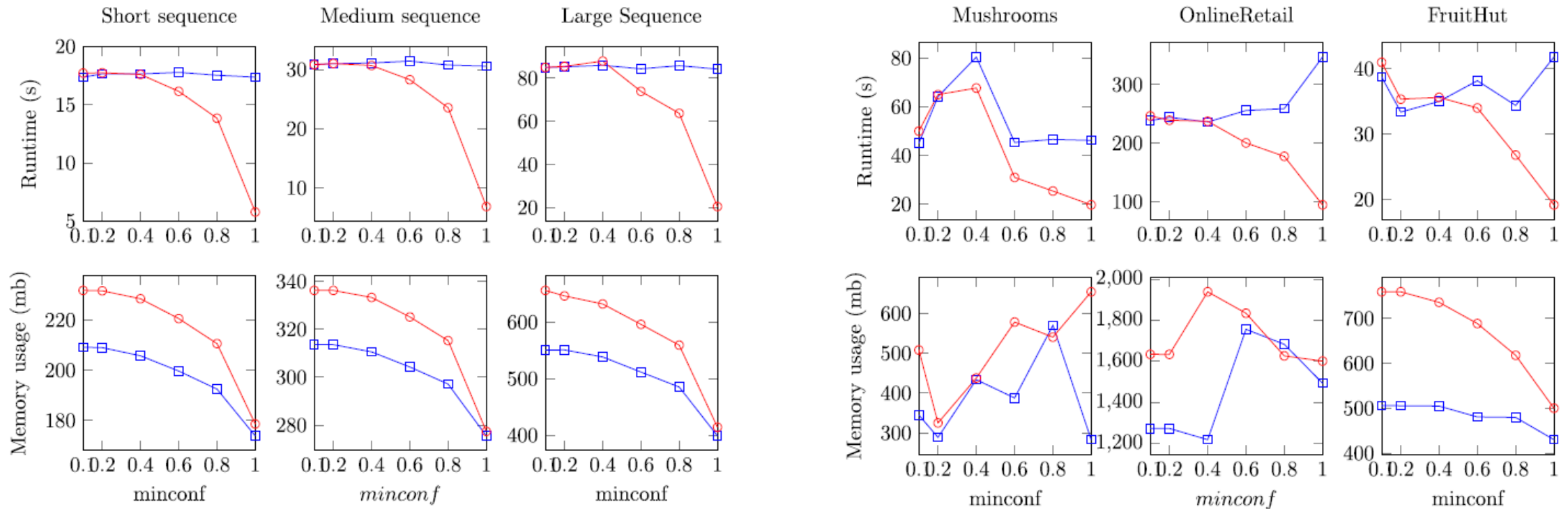
Análise de Resultados

- Descoberta de episódios frequentes (**EMDO**)
 - Avalia como o **limite de suporte** influencia o número de episódios frequentes e **tamanho** em dados reais e sintéticos, respectivamente.
 - Mostrando claramente que a **propriedade anti-monotônica é muito efetiva para a poda** no espaço de busca pela abordagem proposta.



Análise de Resultados

- Geração de regras a partir de episódios frequente (**EMDO e EMDO-P**)
 - EMDO_P gasta **mais memória**, mas o **custo é razoável**.
 - EMDO_P é **mais rápido**, pois **elimina episódios irrelevantes**.



Comparações

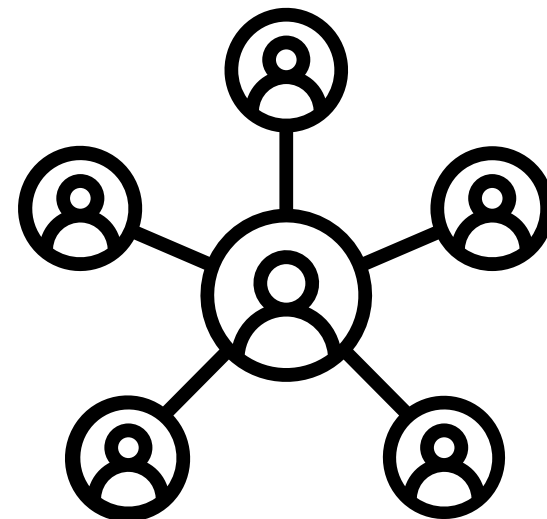
- **Comparações com MINEPI, MINEPI+, NONEPI:**
 - EMDO encontra padrões com **menos candidatos**.
 - E esse padrões/regras/relações são **mais ricas e abrangentes, não detectáveis pelas abordagens tradicionais**.
 - Garante **equilíbrio entre precisão e eficiência** ao explorar um espaço de busca reduzido com resultados mais informativos

Episode	MINEPI+	EMDO	NONEPI	MINEPI
Lettuce Iceberg, Banana Cavendish	23736	8972	7138	7582
Field Tomatoes, Banana Cavendish	35890	20220	13323	14862
Cucumber Lebanese, Banana Cavendish	27034	10489	8261	8886
Field Tomatoes, Cucumber Lebanese, Banana Cavendish	16316	10483	/	/

	MINEPI+	EMDO	NONEPI	MINEPI
Number of candidate episodes	26642	9959	1942	3993
Number of frequent episodes	992	3243	35	1244
Largest frequent episode size	8	6	2	13

Conclusões

- Primeiro algoritmo a explorar **ocorrências distintas** em **episódios paralelos** de sequências complexas.
- **EMDO e EMDO_P** mostraram:
 - Boa eficiência de tempo e memória e capacidade de **descobrir padrões úteis e ocultos**.
 - Especificamente, o EMDO_P **descobre mais padrões relevantes** por não se limitar a subepisódios. Por outro lado, **sua implementação exige estrutura de dados e lógica de controle mais complexas**, o que pode **dificultar sua aplicação direta em ambientes com restrições técnicas**.
- Abrem caminho para:
 - **Recomendações e previsões baseadas em padrões temporais**.
 - Extensões para **dados incertos** ou **eventos contínuos (streams)**.



Assessor Social

Grupo: DANIEL SCHLICKMANN BASTOS
GABRIEL CASTELO BRANCO ROCHA
GUILHERME BUXBAUM MARINHO GUERRA
JOSE EDUARDO DUARTE MASSUCATO

LEONARDO CAETANO GOMIDE
LUCAS MESQUITA ANDRADE
VINICIUS LEITE CENSI FARIA

Como EMDO impacta a sociedade?

- Como visto, o **EMDO** e o **EMDO_P** permitem inferir padrões à partir de sequências complexas de eventos.
- O mundo real é regido por eventos desse tipo: Sistemas IoT, Comércio, Telecom, Comportamentos de usuários...
- Identificação de padrões em sequências complexas pode ser ferramenta chave para diversas soluções

Impactos Positivos e Aplicações Potenciais

- **Análise de Sistemas:** Diversos sistemas modernos (Redes, IoT, etc.) Podem aplicar a descoberta de padrões para otimizar seu funcionamento
- **Análise de comportamento:** Logs de sistemas web, que são compostos de sequências de eventos complexas, se analisados para otimizar a experiência do usuário
- **Análise de Varejo e Comportamento de Compra:** Extração de regras pode ser utilizada para desenvolver estratégias de marketing baseadas em promoções ou recomendações
 - O **EMDO** é capaz de gerar regras que outros algoritmos não conseguem encontrar (considerando os mesmos dados).

Riscos e desafios éticos

- **Vigilância e Privacidade:** Com a necessidade de dados granulares e carimbados por timestamp, a possibilidade de rastreamento de atividades com más intenções é alta.
 - Vigilância de funcionários para aplicar punição
- **Reforço de viés:** Ao aprender com dados históricos, que podem refletir preconceitos e desigualdades, o algoritmo reconhece esses padrões como “regras”
 - Crédito e Finanças
- **Acesso à Oportunidade:** Criação de sistemas de pontuação para governar acesso a serviços essenciais.
 - Acesso a seguro de saúde

Em Resumo...

- Qualquer processamento de dados gera riscos diretos à Privacidade e à segurança de usuários
- Algoritmos que inferem regras ou padrões nos dados sempre vão possuir uma dicotomia:
 - A descoberta de padrões permite inferir regras que podem otimizar os serviços
 - As mesmas regras que foram inferidas e otimizam um serviço certamente irão desconsiderar variáveis que não estavam presentes, e isso pode levar a problemas sérios de segurança.

Hacker

Bernardo Seraphim

Diná Xavier

Gabriel Fadoul

Kênia Gonçalves

Oluwatoyin Joy

Samuel Kfuri

Input de variáveis

Input das variáveis em cada etapa de execução do EMDO

1. De acordo com o Capítulo 5 do artigo, os autores convertem bases de transações em sequências complexas de eventos para alimentar o algoritmo EMDO.
2. Cada linha da base é tratada como um itemset, como em bases de mercado (por exemplo, uma lista de itens comprados juntos em uma transação).
3. Esse itemset é transformado no conjunto de eventos ε_i , que representa todos os eventos que ocorreram simultaneamente em um mesmo ponto no tempo.
4. O número da transação (ou sua ordem na base) é usado como o timestamp t_i , ou seja, o tempo associado ao conjunto de eventos ε_i .

Dessa forma, a base é interpretada como uma sequência ordenada de pares (ε_i, t_i) , representando a evolução temporal dos eventos ao longo das transações.

Instalação e dependências

O artigo indica os seguintes locais como repositórios:

- <https://github.com/Oualidinx/EMDO-synthetic-datasets.git>
- <http://www.philippe-fournier-viger.com/spmf/>

1. Acesse: <https://github.com/Oualidinx/EMDO-synthetic-datasets>
2. Faça o clone ou download do repositório com: git clone <https://github.com/Oualidinx/EMDO-synthetic-datasets.git>
3. Os arquivos representam sequências complexas, com eventos que ocorrem simultaneamente em timestamps e por isto devem ser utilizados individualmente.
4. O algoritmo foi implementado em JAVA e logo não possui nenhuma dependência obrigatória*
5. Baixar JDK (Java Development Kit) na versão mais recente;
6. Há duas maneiras de executar o SPMF
 - Via arquivo.jar
 - Compilar localmente

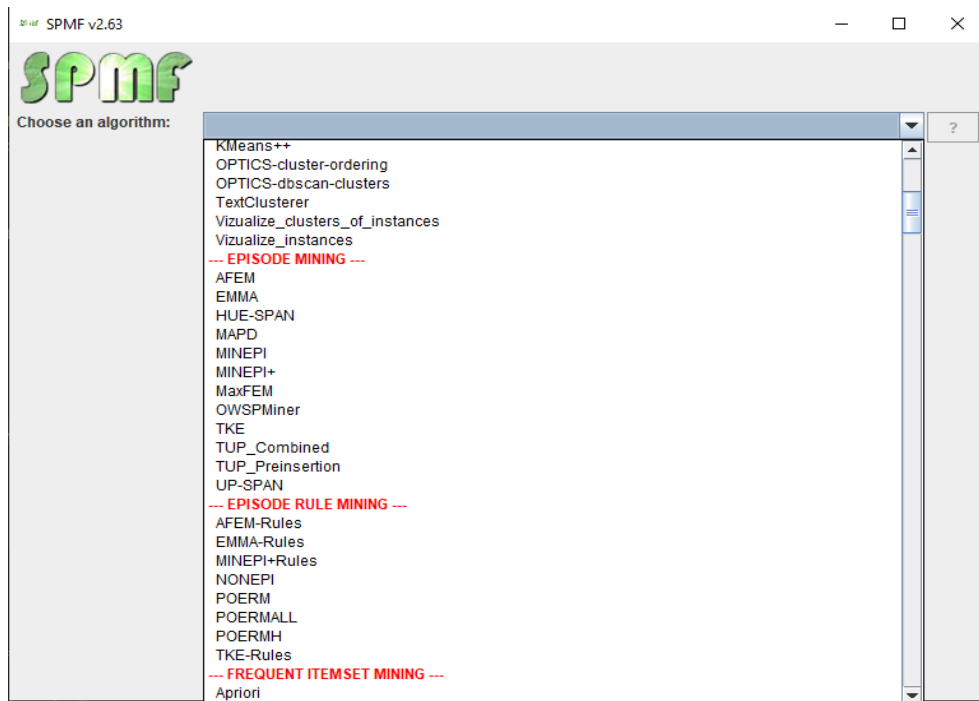
Exemplo dos dados

1	120#14 9 2 5 1 4	→ 120 → ID da transação
2	126#10 11	(14 9 2 5 1 4) → Conjunto dos eventos
3	143#12 14 7 6 2 11	
4	168#1 8	
5	194#14 1 8 13 12	
6	195#7 4 6 10 13	
7	203#11 2 0 1 3 14	
8	231#3 12 1 8 7	
9	244#12 9 13 7 3	
10	273#3 0 9	

Formato do dado: txt

Implementação dos algoritmos

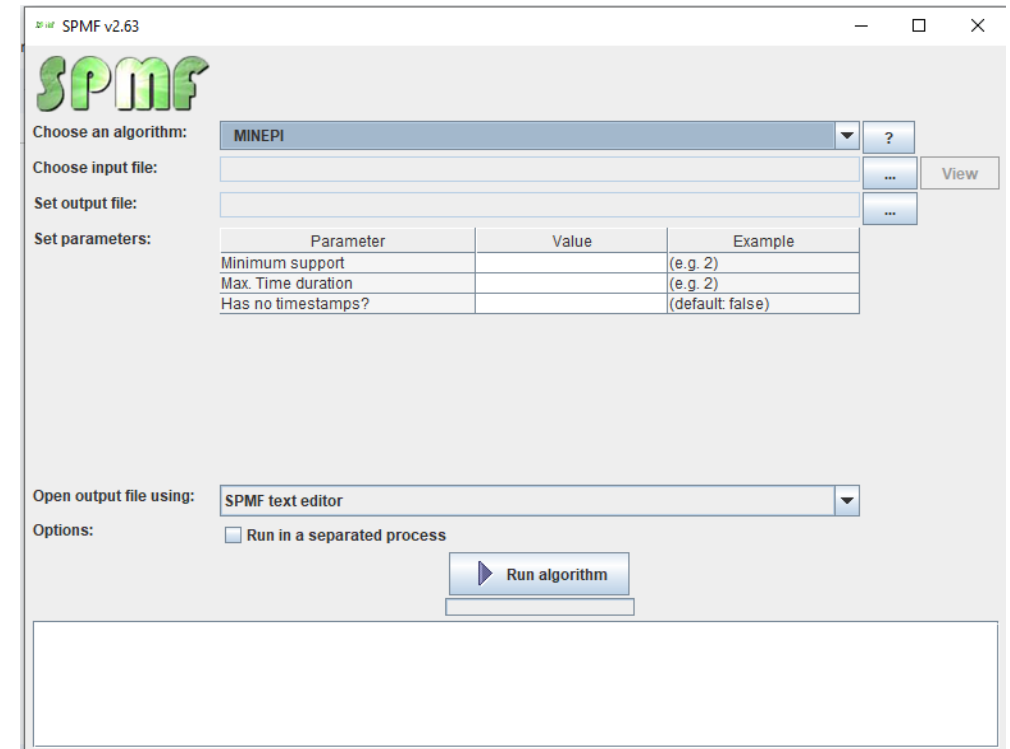
1. Via arquivo .jar: Escolha o algoritmo



2. Selecione a base de dados em “set input file”

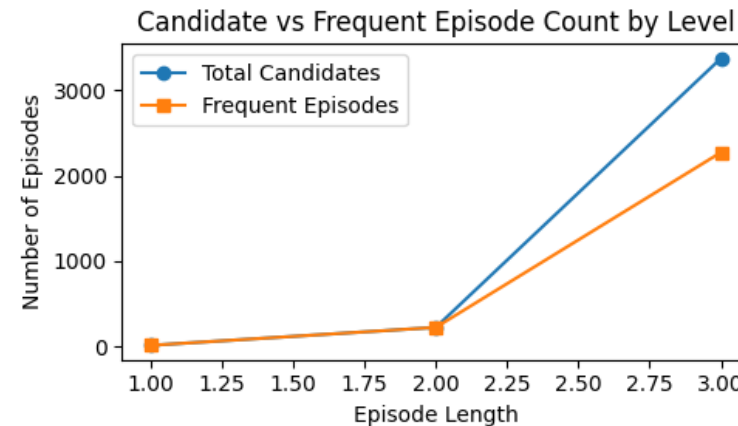
3. Coloque o nome do arquivo de saída em “set output file”

4. Configure os parâmetros e execute em “Run algorithm”



Resultados e observações

- O algoritmo não foi encontrado no link citado no artigo, como demonstrado;
- Optamos pela implementação localmente, tanto via .jar quando via compilação local;
- O github dos três autores foi verificado
- Tentamos implementar uma versão simples do edmo usando um dos conjuntos de dados do repositório (Sequence_40000_6_15) seguindo estes passos:
- Analisamos os dados em pares (conjunto de eventos, posição).
- Inicializamos o EMDO com sequence, minsup, max_len e winlen.
- Geramos candidatos de comprimento 1 a partir de eventos únicos.
- Correspondemos e contamos episódios usando posições iniciais distintas e uma janela opcional.
- Expandimos episódios frequentes por meio de junção baseada em prefixo até max_len.



Frequent episodes:

- 0 : 9265
- 1 : 9298
- 10 : 9471
- 11 : 9420
- 12 : 9273
- 13 : 9117
- 14 : 9390
- 2 : 9277
- 3 : 9246
- 4 : 9206
- 5 : 9287
- 6 : 9376
- 7 : 9257
- 8 : 9175
- 9 : 9350
- 0 → 0 : 4996
- 0 → 1 : 5106
- 0 → 10 : 5133
- 0 → 11 : 5138
- 0 → 12 : 5106
- 0 → 13 : 5082
- 0 → 14 : 5102
- 0 → 2 : 5004
- 0 → 3 : 5073
- 0 → 4 : 5044
- 0 → 5 : 5116
- 0 → 6 : 5023
- 0 → 7 : 5077
- 0 → 8 : 4912
- 0 → 9 : 5050
- 1 → 0 : 5059

Referências

Referências bibliográficas