Redução no espaço de busca para mineração de itemset-like patterns

Samuel Lipovetsky
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil
samuellpvt@gmail.com

Loic cerf
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil
lcerf@dcc.ufmg.br

***Abstract*—In tensor data mining, discovering itemset-like patterns can be highly computationally intensive. This paper introduces a method for reducing the search space by pre-processing tensors to eliminate cells that cannot contribute to valid patterns. The proposed approach involves analyzing intersections between different tensor slices and applying noise limits to refine the search process. By removing subspaces that do not meet the specified criteria, we optimize the efficiency of the mining process without losing information about potential patterns. This technique enhances the computational feasibility of pattern discovery in large-scale tensor data and provides a practical solution for the problem.**

I. Introdução

Tensores fuzzy, definidos como tensores n-dimensionais com valores no intervalo [0, 1], indicam o grau em que n-tuplas satisfazem um predicado booleano. O algoritmo Multidupehack [1] foi projetado para identificar padrões significativos nesses tensores, buscando padrões que são sub-tensores cujos valores das n-tuplas associadas estejam próximos de 1. Para tornar a mineração mais eficiente e focada, são aplicadas restrições que definem o tamanho mínimo dos padrões em cada dimensão além das cotas de ruido que definem o quão próximo de 1 as n-tuplas dos padrões devem ser. Essas restrições não apenas ajudam a filtrar padrões irrelevantes, mas também reduzem significativamente o esforço computacional necessário para encontrar os padrões relevantes. Um dos principais desafios na mineração desses padrões é a eficiência computacional. A natureza combinatória do problema exige algoritmos otimizados para que os tempos de execução sejam viáveis, uma vez que são enumerados todos os padrões que seguem as restrições definidas. Assim, este Projeto Orientado de computação propõe um novo pré-processamento para o algoritmo Multidupehack [1] e melhorias para o pré-processamento proposto em [2]. O algoritmo recebe como entrada um tensor n-dimensional, onde cada n-tupla é um valor de grau de pertencimento no intervalo $[0, 1]$, além de N tamanhos mínimos e N ruídos máximos. O conceito de ruído refere-se à tolerância permitida para desvios nos graus de pertencimento das n-tuplas dos padrões válidos. O resultado obtido são padrões representados como sub-tensores extraídos dos dados originais. Esses sub-tensores possuem n-tuplas cujos graus de pertencimento são consistentemente elevados, respeitando os limites predefinidos de ruído máximo e os critérios mínimos de tamanho estabelecidos para cada dimensão dos padrões minerados. Para o caso mais simples com os dados de entrada sendo uma matriz com duas dimensões, os tamanho mínimos controlam o número de linhas e colunas dos padrões retornados. Já os ruídos máximos controlam para as linhas e colunas dos padrões válidos o quão distante essas podem estar de serem compostas somente por n-tuplas com grau de pertencimento igual a 1.

Suponha que uma empresa está realizando uma pesquisa sobre aspectos de qualidade de diversos produtos com seus clientes. Isso gera um tensor tridimensional em que cada célula (u, p, a) representa uma nota X de 0 a 1. Nesta notação, u representa um usuário, p representa um produto e a representa um aspecto do produto avaliado. Assim, um analista desta empresa deseja obter um subconjunto desse tensor que seja próximo de ser composto somente de 1s, com um limite de ruído determinado pelo analista e que então represente um subconjunto de notas de usuários por aspecto de um produto em que a satisfação foi elevada.

	aspecto 1	aspecto 2	aspecto 3	aspecto 4
Alice	0.8	0.1	1	0.3
Bob	0.7	0.3	0.8	0.2
Carol	0.8	0.2	0.9	0.1

TABLE I: Fatia de Notas de usuários para aspectos de um produto

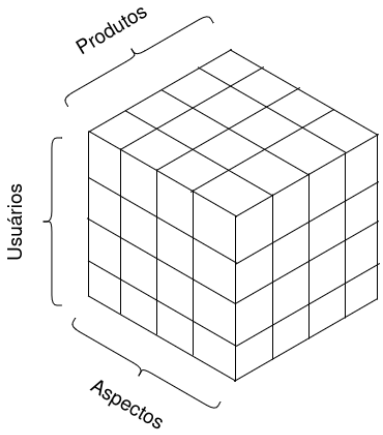
A maneira como o analista define o quão próximo de 1 o subconjunto minerado deve ser é a partir de um limite de ruido por dimensão do conjunto de dados. Por exemplo na tabela 1, temos uma fatia dos dados que fixa um produto e contem todas as notas dos aspectos por usuários daquele produto. Igualmente, existem dois outros tipo de fatias, uma que fixa um usuário e outra que fixa um aspecto. Assim, o ruido de uma fatia é soma das diferenças entre 1 e o valor de cada célula daquela fatia. Dessa forma, o analista deve definir os limites de ruido por fatia e o algoritmo multidupehack retorna subconjuntos do tensor em que o ruido das fatias desses subconjuntos sempre são menores que o limite definido. Ainda é possível definir outros limites para os padrões minerados que são descritos em [**?**].

Como minerar esses tipos de padrões é um processo extremamente caro computacionalmente, uma das maneiras de

otimizar a velocidade de execução do algoritmo é diminuir o espaço de busca removendo células que com certeza não vão pertencer a nenhum padrão válido especificado pelos limites de ruídos. Assim, o foco deste trabalho é pré-processar os dados observando as interseções entre os diferentes tipos de fatias e como o limite de ruido definido para um tipo de fatia pode ser usado em outros tipos de fatias para reduzir o espaço de busca.

II. Visualizando o problema geometricamente

Ainda Utilizando o exemplo da pesquisa com usuários sobre aspectos de vários produtos, que pode ser descrita como um tensor tridimensional que atribui um valor entre 0 e 1 para cada célula (u, p, a) que pertencem respectivamente aos conjuntos $Usuarios$,$Produtos$ e $Aspectos$



via

Fig. 1: Visualização geométrica do tensor.

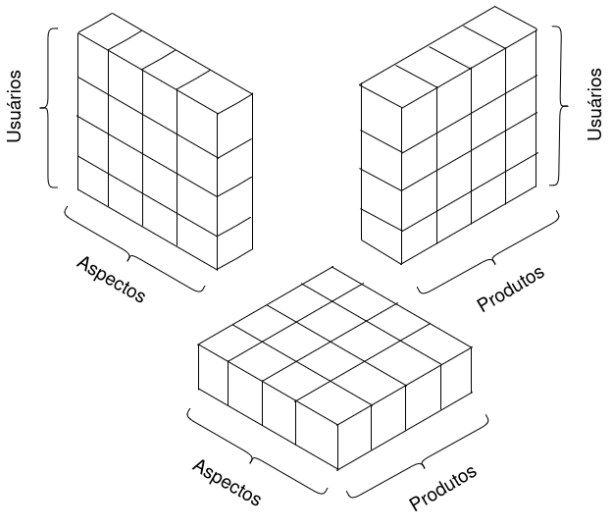
Então são definidos valores limites para os ruídos e para os tamanhos mínimos de cada tipo de fatia dos subtensores que serão minerados.

Assim, para cada fatia do subtensor minerado, a soma dos ruídos de cada célula da fatia deve ser menor que o limite especificado. Sobre o número de fatias de cada tipo, para fatias do tipo $Usuario$ que representa as notas de um usuário sobre todos os aspectos de todos os produtos, como demostrado na tabela 1, existem $|Usuario|$'s desse tipo de fatia no tensor original , sendo que essa lógica se expande para todos os outros tipos de fatia.

A. Limites de ruídos ortogonais

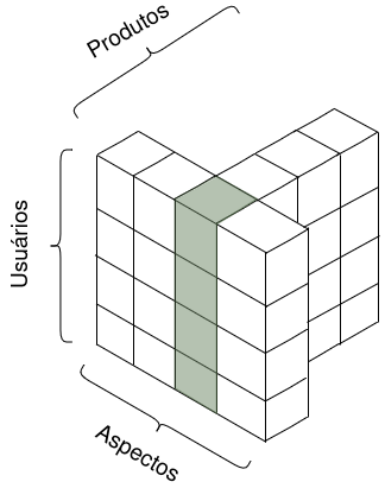
Como existem conjuntos de elementos que são comuns entre fatias ortogonais, ou seja elementos comuns entre dois tipos de fatias diferentes, esses elementos que são interseções de fatias devem obedecer os limites de ruídos de ambas as fatias que os contem.

Dessa forma, é possível diminuir ainda mais o espaço de busca a partir da remoção de conjuntos de elementos que nunca vão pertence a um padrão válido sem perda de informação para os padrões minerados apos o pré-processamento. No exemplo da Figura 3, a coluna destacada



via

Fig. 2: Tipos de fatias do subtensor.



via

Fig. 3: Interseção de fatias

deve obedecer aos limites de ruído definidos tanto para as fatias de $Usuarios \times Aspectos$ quanto para $Usuarios \times produtos$, uma vez que o subespaço destacado pertence aos dois tipos de fatia ao mesmo tempo. Assim, este trabalho busca provar que é possível remover subespaços do tensor observando esse tipo de interseção sem perda de informação nos padrões minerados.

III. Definição de padrões minerados

Sejam $D_1, .., D_n$ com $n \in \mathbb{N}$ conjuntos finitos assumidos disjuntos sem perda de generalidade. Existe um tensor T que mapeia cada N-tupla $\prod_{k=1}^{n} D_k$, em que $\prod$ representa o produto cartesiano, para um valor $T_t \in [0, 1]$ que é chamado de grau de pertencimento. Assim , podemos definir π que recebe

um conjunto de E que é uma união de subconjuntos unitários de $D_1, ..., D_n$.

$$\pi : \begin{array}{c} 2^{\cup_{i=1}^{n} D_i} \to 2^{\prod_{i=1}^{n} D_i} \\ (E,(D_1,..,D_n)) \mapsto \prod_{i=1}^{n} F_i \begin{cases} \text{se } E \cap D_i = \varnothing, F_i = D_i \\ \text{senão }, F_i = E \cap D_i \end{cases} \end{array}$$

Assim usando o exemplo da figura 2, com $E = \{Alice, produto1\}$ e $D = Usuarios, Produtos, Aspectos$.

$$\pi(\{Alice, produto_1\}, D) = \{Alice\} \times \{produto_1\} \times \{Aspectos\} \tag{2}$$

IV. Usando a função π para definir padrões válidos

O multidupehack recebe como entrada os conjuntos de dimensões com os graus de pertencimento de cada n-tupla, além de receber os limites de ruido $\{\epsilon_1, .., \epsilon_n\} \in \mathbb{R}_+$ e os tamanhos mínimos de cada dimensão $\{\gamma_1, .., \gamma_n\} \in \mathbb{N}_+$ e retorna os subconjuntos $X_1, .., X_n$ tal que $X_1 \subseteq D_1, .., X_n \subseteq D_n$ somente se:

$$\forall i \in \{1, ..., n\} \begin{cases} \forall e \in X_i, \sum_{t \in \pi(\{e\},(X_1,..,X_n))} (1 - T_t) \leq \epsilon_i \\ |X_i| \geq \gamma_i \end{cases} \tag{3}$$

V. Zerando sub-espaços $\pi(E,(D_1,..,D_n))$ de T_t

Para zerar qualquer n-tupla de T_t sem perda de informação nos padrões minerados pelo multidupehack, é necessário provar que as n-tuplas removidas não pertencem a nenhum padrão válido. Assim, dado um tensor R_t, que será chamado do tensor reduzido, que é obtido a partir de :

$$\begin{array}{c} (T, \pi(E, \{D_1, .., D_n\})) \mapsto R_t \\ R_t \mid \forall t \in R_t \begin{cases} \text{se } t \in \pi(E, \{D_1, ..., D_\}), t = 0 \\ \text{senão } t = Tt \end{cases} \end{array} \tag{4}$$

É preciso provar que os padrões válidos do tensor original T_t são os mesmos do que os do tensor reduzido R_t dado um subespaço $\pi(E,(D_1,..,D_n))$

Para auxiliar na prova será definida a função $index$ que retorna os índices das dimensões que não possuem interseção com E:

$$index : \begin{array}{c} 2^{\cup_{i=1}^{n} D_i} \to I \subseteq \{1, .., n\} \\ E \mapsto \{i \mid D_i \cap E = \varnothing\} \end{array} \tag{5}$$

Assim, seja que $B = max(m, \pi(E,(D_1, ..., D_n)))$ representa as m n-tuplas com maiores valores dentro de $\pi(E,(D_1, ..., D_n))$, podemos remover esse subespaço sem perda de padrões válidos, somente se:

$$\nexists m \in \mathbb{N} \begin{cases} \sum_{t \in B} (1 - T_t) \geq min(\{\epsilon_k | k \notin \text{index}(E)\}) \\ m \geq \prod_{i \in \text{ index}(E)} \gamma_i \end{cases} \tag{6}$$

Ou seja, é possível remover um subespaço $\pi(E,(D_1,...D_n))$ somente se não existem pelo menos m células dentro desse subespaço tal que que as soma dos ruídos dessas m células é menor que o menor limite de ruido das dimensões com elementos em E , ou se m existir mas for menor que o produto dos limites de tamanhos das dimensões sem elementos em E.

(1)

Portanto, basta provar que para um dado subespaço $\pi(E,(D_1, ..., D_n))$ se existe m que satisfaz (6) então $\pi(E,(D_1, ..., D_n))$ não pertence a um padrão válido e pode ser removido.

Seja $(E_1, .., E_n)$ as dimensões de $\pi(E,(D_1, .., D_n)$ com os limites de ruido $(\epsilon_1, .., \epsilon_n) \in \mathbb{R}_+$ e os tamanhos mínimos de cada dimensão $(\gamma_1, .., \gamma_n) \in \mathbb{N}_+$, se existe m que satisfaz (6), logo teríamos:

$$\sum_{t \in \max(m, \pi(E))} (1 - T_t) \geq min(\{\epsilon_k | k \notin \text{index}(E)\}) \tag{7}$$

e como $m \geq \prod_{i \in \text{ index}(E)} \gamma_i$ é possível concluir que:

$$\exists e_i \in E \mid \sum_{t \in (\pi(e,(E_1,..,E_n)))} (1 - T_t) \geq \epsilon_i \tag{8}$$

e assim $\pi(E,(D_1, ..., D_n))$ não pode estar em um padrão válido, uma vez que a desigualdade (9) contradiz a definição do padrão descrito em (3).

VI. Estrutura de dados utilizadas

Os dados são lidos e armazenados numa Trie, porém, a ultima camada da Trie contém ponteiros para outras estruturas de dados que realmente guardam os graus de pertencimento das tuplas. Isso ocorre pois o tubos dos tensores podem ser densos ou esparsos, e assim, para tubos densos é mais eficiente armazenar os valores em um vetor , já para tubos mais esparsos é mais eficiente utilizar uma lista de adjacência. Ainda, existem tubos específicos para dados binários ou numéricos.

VII. Gerando as restrições no código

Após a leitura dos dados, o novo pré-processamento envolve dois passos principais: gerar as restrições e impor essas restrições. O código responsável por gerar a lista de restrições cria tuplas no formato (identificador,ruído_máximo,tamanho_mínimo). O identificador é composto por uma sequência de zeros e uns, que define todas as combinações possíveis de sub-tensores com determinadas dimensões fixas e dimensões livres. Um zero na posição n do identificador indica que a dimensão n é fixa, ou seja, todo possível sub-tensor gerado conterá apenas um único elemento dessa dimensão. Por outro lado, um um na posição n identifica uma dimensão livre, o que significa que todo possível subtensor gerado incluirá todos os elementos dessa dimensão.

Portanto, existem 2^D identificadores, com D representando o número de dimensões, o que resulta em todas as possíveis combinações de zeros e uns de tamanho D. A partir desses identificadores, a próxima etapa do código, que impõe as

restrições, gera todos os sub-tensores possíveis de serem formados, obedecendo às regras de dimensões fixas e livres associadas a cada identificador. Ao gerar um sub-tensor a partir de um identificador, verifica-se se ele pode pertencer a um padrão, utilizando os parâmetros de ruído máximo e tamanho mínimo definidos para aquele identificador. O ruído máximo é calculado como o menor ruído permitido entre todas as dimensões fixas, enquanto o tamanho mínimo corresponde ao produto dos tamanhos mínimos das dimensões livres. Se a soma dos ruídos das m,igual ao tamanho_mínimo de uma restrição, menores tuplas do sub-tensor gerado for maior que o ruido_máximo da restrição, esse sub-tensor pode ser excluído dos dados sem perda de padrões, como provado na seção 5.

VIII. Paralelizando o novo pré-processamento

Como para cada dimensão fixa de um identificador deve gerar todos os sub-tensores que envolvem cada elemento dessa dimensão, é possível paralelizar o pré-processamento com um fator de speed-up relativamente alto. Isso ocorre pois é possível simplesmente dividir os elementos das dimensões fixas entre as threads e não existe nenhuma condição de corrida entre elas, umas vez que elas vão gerar sub-tensores totalmente disjuntos, que não compartilham nenhuma tupla.

IX. Experimentos

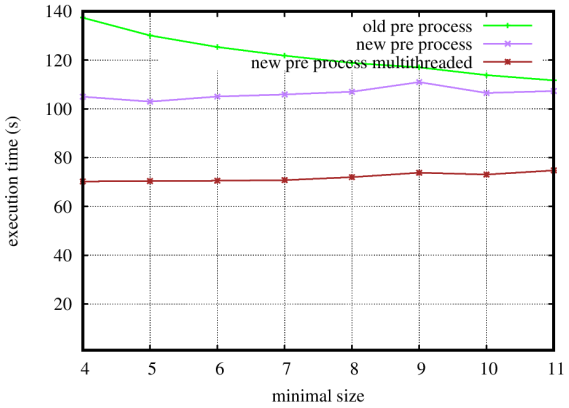
Para mostrar empiricamente os resultados obtidos com o novo pré-processamento foram feitos experimento em três bases de dados reais diferentes e para cada uma dessas foram minerados os padrões com hiper-parâmetros adequados.

A. Metodologia dos experimentos

Para cada base de dados foi executado todo o processo de mineração e o tempo de comparação sempre será o do processo completo, ou seja, ler, tratar, minerar os padrões e gerar o output. Assim, para cada versão do multidupehack em questão sendo essas: a versão com o pré-processamento antigo, a versão com a nova técnica de pré-processamento single threaded e a versão com a nova técnica multithreaded foi executado o processo de mineração para cada base de dados e os tempos de execução serão comparadas nas próximas seções. Para gerar gráficos que comparam como o desempenho das versões se comportam quando aumentamos o tamanho mínimo dos padrões sendo minerados, cada versão foi executada n vezes sempre aumentando o tamanho mínimo de uma das dimensões dos padrões de em um em um.

B. Base de dados 1: Twitch.tv

Essa base de dados, descrita em [3] gerada a partir dos logs dos chats da plataforma Twitch.tv tem informações sobre o quanto um usuário (entre os 1,198,282) da plataforma gosta de assistir um dado canal de Starcraft II (entre os 94 coletados) semana por semana entre as 19 semanas de 7 de Outubro de 2013 a 16 de Fevereiro de 2014. Os graus de pertencimento são derivados sobre quanto tempo o usuário permaneceu em um dado canal durante aquela semana.
Cada padrão minerado nessa base de dados indica um conjunto de usuários, semanas e canais em que durante essas semanas um conjunto de usuários assistiram um conjunto de canais.



via

Fig. 4: Tempo de execução twitch.tv

Observando a figura 4 é possível perceber que houve ganho de tempo significativo ao se usar o novo pré-processamento, principalmente ao se usar multiplas threads. Inicialmente quanto o tamanho mínimo do conjunto de usuários era 4, houve um ganho de cerca de duas vezes em relação ao tempo. Como quanto maior o tamanho mínimo dos padrões também se torna maior a possibilidades de prunning durante o branch and bound do algoritmo de mineração, a redução no tempo ao se aumentar o tamanho mínimo do conjunto de usuário é esperada.

C. Base de dados 2: Retweets

Esse tensor tridimensional de retweets indica o quão influente as mensagens de um usuário do Twitter (entre os 170,670) sobre os times de futebol brasileiros (entre os 29 coletados) durante uma certa semana (entre as 12 coletadas). Nesse caso, o grau de pertencimento é criado a partir de quantas vezes um tweet foi ”retweetado” e também sobre a popularidade geral do time em questão no tweet.

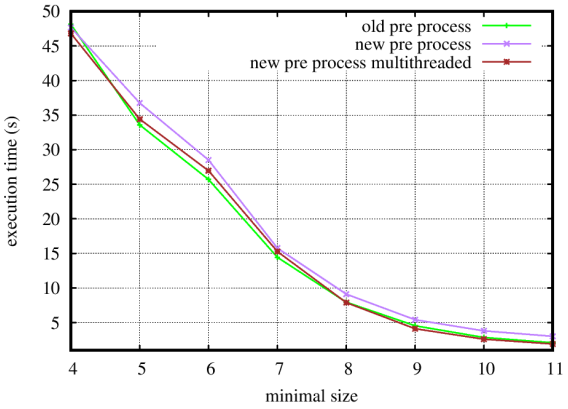


Fig. 5: Tempo de execução retweets

Pela figura 5 podemos perceber que não houve ganhos de tempo significativo para essa base de dados. Isso ocorre devido

ao formato dos dados que faz com o que pré-processamento antigo e novo executem passos muitos similares. O novo pré-processamento é uma generalização do pré-processamento antigo,e assim, existem casos em que ambos reduzem o espaço de busca de forma similar.

D. Base de dados 3: Velov

Já a terceira database, descrita em [4] aborda o uso do sistema de compartilhamento de bicicletas em Lyon, França, composto por 327 estações para aluguel e devolução. Os padrões de uso variam ao longo do dia e entre os dias da semana. Por isso, foram construídos 7×24 grafos direcionados (um para cada dia da semana e intervalo de uma hora) com base em dois anos de registros. As arestas representam o número de viagens entre pares ordenados de estações, normalizados para que cada grafo tenha o mesmo peso total. Em seguida, uma função logística transforma esses valores normalizados em graus de pertinência, resultando em um tensor difuso de dimensão $7 \times 24 \times 327 \times 327$ e densidade 0,005.

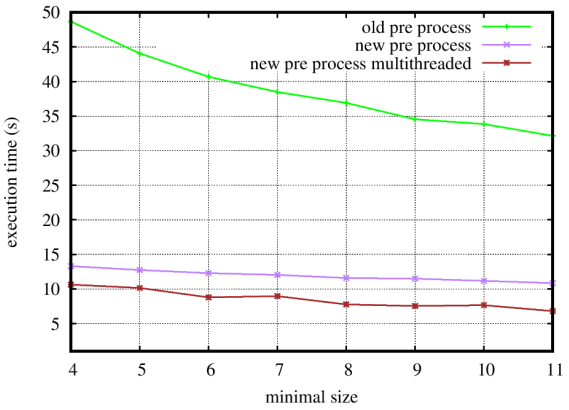


Fig. 6: Tempo de execução velov

É possível observar que para esse data set houve o melhor ganho de tempo de execução, isso ocorre pois o novo pré-processamento consegue reduzir mais o espaço de busca do que o pré-processamento antigo pois checa uma maior quantidade de sub-tensores.

X. Comparação do uso de memória

Para comparar também o uso de memória RAM para cada dataset, foram feitos experimentos que para cada dataset e versão do pré-processamento observam o maximum resident set size(RSS) de cada processo. Apesar dessa métrica não ser 100% realística, ela é uma boa indicação do uso de memória. Para todos esses experimentos os hiper parâmetros utilizados foram os menos limitantes ou seja aqueles que geram o maior número de padrões, dos utilizados nas comparações de tempo de execução das figuras 4,5 e 6. Ainda, foi utilizado o hiper-parâmetro que diz para o algoritmo não minerar nenhum padrão e somente rodar o pré-processamento, então o RSS sempre sera o do pré-processamento juntamente coma leitura dos dados que é comum para todas as versões. Como as versões multithreaded e singlethread não possuem diferenças significativas no uso de RAM, foram feitos gráficos somente com a versão antiga do pré-processamento e a versão multi-threaded.

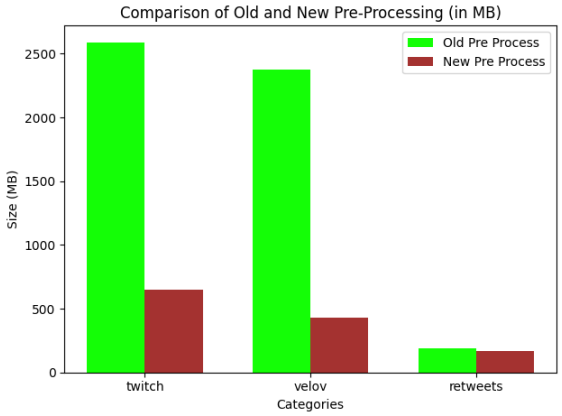


Fig. 7: Comparação do uso de RAM

Pela figura 7 é possível perceber que o novo pré-processamento utiliza quantidades significativamente menores de RAM quando comparado com o pré-processamento antigo. Isso ocorre devido as estruturas de dados utilizadas no pré-processamento antigo que armazena os dados D igual ao número de dimensão vezes.

XI. Conclusão

A partir dos gráficos da figuras 4,5 e 6 que comparam os tempos de execução é possível concluir que a nova técnica de pré-processamento é capaz de reduzir os tempos de execução do algoritmo multidupehack principalmente em casos de datasets com mais de 3 dimensões. Ainda, é possível concluir que devido a diferenças nas estruturas de dados utilizadas o novo pré-processamento utiliza menos RAM quando comparado com o pré-processamento antigo. Além disso, a técnica de paralelização do novo pré-processamento foi capaz de reduzir significativamente o tempo de execução em 2 dos 3 datasets.

References

[1] L. Cerf and W. Meira Jr., "Complete discovery of high-quality patterns in large numerical tensors," in *Proceedings of the 30th International Conference on Data Engineering (ICDE'14)*. IEEE Computer Society, April 2014, pp. 448–459.

[2] L. Cerf, "Enforcement of minimal size and area constraints before and while mining patterns in fuzzy tensors," in *Proceedings of the 38th ACM/SIGAPP Symposium On Applied Computing (SAC'23)*. ACM Press, March 2023, pp. 369–376.

[3] M. Kaytoue, A. Silva, L. Cerf, W. M. Jr., and C. Raïssi, "Watch me playing, i am a professional: A first study on video game live streaming," in *WWW'12: Proceedings of the 21st World Wide Web Conference (Companion Volume)*. ACM Press, April 2012, pp. 1180–1188, supplementary material. Acceptance rate: 55%.

[4] L. Maciel, J. Alves, V. F. dos Santos, and L. Cerf, "Climbing the hill with ilp to grow patterns in fuzzy tensors," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1036–1047, July 2020.