UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

INSTITUTO DE BIOCIÊNCIAS, LETRAS E CIÊNCIAS EXATAS
IBILCE / UNESP - CÂMPUS DE SÃO JOSÉ DO RIO PRETO

WILLIAM WALLACE COLLA CORREIA

Algoritmo eficiente de Mineração de Dados Multi-Relacional, baseado em Regras de Associação, com uso de Múltiplos Suportes Mínimos e recursos de paralelização

São José do Rio Preto
2021

WILLIAM WALLACE COLLA CORREIA

Algoritmo eficiente de Mineração de Dados Multi-Relacional, baseado em Regras de Associação, com uso de Múltiplos Suportes Mínimos e recursos de paralelização

Trabalho de Conclusão de Curso apresentado ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Carlos Roberto Valêncio

São José do Rio Preto
2021

WILLIAM WALLACE COLLA CORREIA

Algoritmo eficiente de Mineração de Dados Multi-Relacional, baseado em Regras de Associação, com uso de Múltiplos Suportes Mínimos e recursos de paralelização

Trabalho de Conclusão de Curso apresentado ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Banca examinadora:

Prof. Dr. Carlos Roberto Valêncio
UNESP – São José do Rio Preto
Orientador

Prof. Dr. Adriano Mauro Cansian
UNESP – São José do Rio Preto

Prof. Dr. Geraldo Francisco Donega Zafalon
UNESP – São José do Rio Preto

São José do Rio Preto
15 de março de 2021

Agradecimentos

A Deus pela oportunidade, por ter me guardado e abençoado durante minha caminhada.

A minha mãe, Aparecida, pelo apoio, motivação e sempre estar ao meu lado, no qual sem isso, não seria possível terminar a graduação.

A minha namorada Lara, pela companhia, carinho e motivação.

A toda a minha família, que sempre ficou na torcida apoiando.

Aos meus amigos que conheço na maior parte da minha vida, em especial ao Trick e aos meus amigos que tive a oportunidade de conhecer durante a graduação.

Aos integrantes do GBD, por todo o conhecimento compartilhado.

A todos os professores, em especial o professor Valêncio, pelos ensinamentos e auxílio durante a graduação.

“Nada como procurar quando se quer achar alguma coisa. Quando se procura geralmente se encontra alguma coisa, sem dúvida, mas nem sempre o que estávamos procurando”

J.R.R. Tolkien

Resumo

Com o avanço das tecnologias das mídias sociais, internet das coisas e a consequente geração de grandes volumes de dados, faz-se necessário o desenvolvimento de algoritmos, técnicas e ferramentas com o propósito de extrair conhecimento útil destes dados. O processo que sistematiza e reúne toda a coleção de recursos para efetuar esta tarefa é denominado de Processo de Extração de Conhecimento em Banco de Dados, ou do inglês, Knowledge Discovery in Databases – KDD. Uma das etapas do KDD é a mineração de dados, em que se tem uma série de proposições de algoritmos que executam a tarefa de encontrar padrões e comportamentos, ou seja, o conhecimento buscado Em especial a mineração de regras de associação, que visa determinar associações entre dados em uma grande base de dados. Porém tal técnica possui um problema conhecido como "problema do item raro", sendo solucionado com o uso de múltiplos suportes mínimos. Outro problema encontrado, é quanto a aplicação do algoritmo que se dá em uma única tabela e sobre a qual faz-se necessária a aplicação de preparação prévia nestes dados, sendo necessário aplicar pré-processamento. No entanto, tal procedimento provoca, em algumas ocasiões, a duplicação e ou a perda de dados, com isso, os algoritmos de mineração multi-relacionais foram propostos para contornar tais circunstâncias. Deste modo, o objetivo deste trabalho é a proposição de um algoritmo de mineração multi-relacional, baseado em regras de associação, adicionado de múltiplos suportes mínimos e com desempenho superior e menor consumo de memória do que o correspondente algoritmo da literatura.

Palavras-chave: Mineração de dados, Regras de associação, Mineração multi-relacional, Múltiplos suportes mínimos, Paralela.

Abstract

With the advancement of social media technologies, internet of things and the consequent generation of large volumes of data, it is necessary to develop algorithms, techniques and tools in order to extract useful knowledge from these data. The process that systematizes and brings together the entire collection of resources to perform this task is called the Knowledge Extraction Process in Databases, or Knowledge Discovery in Databases – KDD. One of the stages of KDD is data mining, in which there are a series of algorithmic propositions that perform the task of finding patterns and behaviors, that is, the knowledge sought. between data in a large database. However, this technique has a problem known as the "rare item problem", which is solved with the use of multiple minimal supports. Another problem encountered is regarding the application of the algorithm that takes place in a single table and on which it is necessary to apply a previous preparation in these data, being necessary to apply pre-processing. However, this procedure sometimes causes duplication and/or loss of data, so multi-relational mining algorithms have been proposed to circumvent such circumstances. Thus, the objective of this work is to propose a multi-relational mining algorithm, based on association rules, added with multiple minimal supports and with superior performance and lower memory consumption than the corresponding algorithm in the literature.

Key Words: Data Mining, Association Rules, Multi-Relational Mining, Multiple Minimal Supports, Parallel.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

FSI – *Fusion of Secondary Items*

GBD – Grupo de Banco de Dados

KDD - *Knowledge Discovery in Databases*

LMS - *Least Minimum Support*

LS - *Lowest Support*

MIS - *Minimum Itemset Support*

RDD - *Resilient Distributed Datasets*

SIVAT - Sistemas de Informação e Vigilância de Acidentes de Trabalho

TID - *Transaction Identifier*

Sumário

1 Introdução

Com a computadorização da sociedade, obteve-se um incremento importante no desenvolvimento de técnicas e ferramentas para o armazenamento dos dados gerados por todos estes recursos computacionais que vem sendo disponibilizados. Com isso, possibilitou-se um grande crescimento do volume de dados gerados por computadores de diversas áreas da sociedade, e que se fez necessário o desenvolvimento de novas estratégias de organização e como efetuar a extração de conhecimento útil destes dados (HAN; KAMBER; PEI, 2011).

O processo de extração de conhecimento em banco de dados, termos do inglês Knowledge Discovery in Databases – KDD, é composto por uma coleção de fases, entre as quais, tem-se a fase de aplicação dos algoritmos de mineração de dados, denominada Data Mining e que permite a identificação de padrões e comportamentos nestes volumes de dados. (HAN; KAMBER; PEI, 2011).

1.1 Motivação e Escopo

Um conjunto amplo de algoritmos, baseados em diferentes estratégias, podem ser utilizados para obter padrões e comportamentos válidos, desconhecidos ou úteis em grandes bases de dados, uma destas estratégias é a mineração de regras de associação (ABDEL-BASSET, MOHAMED, et al, 2018). Esta estratégia de mineração visa extrair padrões, correlações e associações de um conjunto de dados, desde que atenda a uma restrição de suporte mínimo, na qual é definida pelo usuário, descartando assim, regras que não são relevantes ou importantes. Esse suporte é dado pelo número de cruzamentos dos seus itens comparado ao conjunto total de dados (KARTHIKEYAN; RAVIKUMAR, 2014).

Inicialmente, a abordagem utilizada foi de centrar-se em uma única tabela alvo, objeto da preparação sobre todos os dados e resultante da seleção daqueles dados que se considera mais relevantes para o propósito da mineração. No entanto, este processo de preparação dos dados, cujos resultado é uma única tabela alvo, tem-se a aplicação de operações de junções e agregações, cujos resultados geram efeitos não desejados, tais como a duplicação, perda e inconsistência dos dados, efeitos que restringem o escopo da extração do conhecimento esperado. Com o propósito de contornar estas dificuldades e restrições na aplicação da mineração de dados, tem-se apresentado os algoritmos de mineração denominados de multi-

relacionais, onde a extração de conhecimento ocorre diretamente nessas tabelas sem o uso do pré-processamento de junção ou agregação dos dados (VALENCIO et al., 2018).

A computação paralela visa melhorar o tempo de resposta e a escalabilidade de grandes conjuntos de dados, ao oferecer vantagens aos métodos de mineração de dados atuais (ZAKI, 2000). Algoritmos de mineração de dados sequenciais e tradicionais utilizam-se de tempo importante para lidar com grandes bases de dados, e com a evolução de *hardware* de computadores com o passar dos anos, em particular, os recursos como memória RAM e processadores, evoluíram significativamente. Por conta disso, os algoritmos tradicionais de mineração de dados não foram concebidos de modo a aproveitar eficientemente as arquiteturas de computadores mais recentes, em que não se considera os benefícios destas plataformas de alto desempenho (YU, KUN-MING, et al, 2015).

Com isso, a combinação de arquiteturas de processadores mais atuais tornou algoritmos paralelos baseados em vários núcleos uma área de pesquisa bem atrativa para ser explorada (YU, KUN-MING, et al, 2015). Com a evolução da memória RAM, aumentando o seu tamanho, permitiu que maiores quantidade de dados sejam armazenados. Dessa maneira, a computação paralela alivia os algoritmos de mineração de dados sequenciais, na qual auxilia em reduzir conjuntos de dados e melhora o tempo de execução desses algoritmos. Portanto, os trabalhos de proposições de algoritmos paralelos de mineração de dados têm sido amplamente explorados, além das suas várias aplicabilidades (YU, KUN-MING, et al, 2015).

1.2 Objetivo

Este trabalho tem como objetivo a proposição de um agoritmo de mineração de dados multi-relacional, baseado em regras de associação, que oferece a possibilidade de definição de múltiplos suportes mínimos e implementado por meio de conceitos de computação paralela. Deste modo, a contribuição científica deste trabalho é a proposição deste algoritmo, que reúne o conjunto de características citados e desempenho computacional superior ao correlato da literatura

1.3 Metodologia

A metodologia científica adotada para o desenvolvimento deste trabalho iniciou-se por meio do levantamento do estado da arte no tema alvo desta proposição, em que se estudou os

principais conceitos e trabalhos correlatos sobre mineração de dados e suas respectivas categorias. Um estudo mais detalhado se deu no tema mineração de dados multi-relacional baseado em regras de associação e a utilização de múltiplos suportes mínimos e com foco na melhoria de desempenho destes algoritmos, em particular, sobre os aspectos de computação paralela e distribuída.

Após amplo estudo deste tema na literatura científica, foram centrados os esforços nos estudos de dois importantes algoritmos da literatura e que utilizam-se de múltiplos suportes mínimos no processo de mineração de dados, conhecido como FP-ME e proposto por Gan et al. (2017). O segundo utiliza uma abordagem tradicional para ser utilizado em ambientes relacionais e é um algoritmo multinacional de mineração de dados, conhecido como MR-Radix e proposto por Valêncio et al. (2012).

Estudou-se o algoritmo proposto por (JENSEN; SORPARKAR, 2000) baseado em regras de associação e que utiliza-se de recurso de computação paralela e o algoritmo paralelo e baseado em regras de associação denominado CBA e que também faz uso de memória distribuída (GHANSHYAM; CHANDRA JAIN, 2008).

1.4 Organização da Monografia

Foi abordado neste capítulo uma introdução do trabalho, assim como a sua motivação e a metodologia utilizada. Para os próximos capítulos, este trabalho está organizado da seguinte maneira:

- Capítulo 2 - Fundamentação teórica: Aborda conceitos introdutórios utilizados para o desenvolvimento do trabalho, como mineração de dados com regras de associação, mineração de dados multi-relacionais, mineração de dados com uso de suportes mínimos, computação paralela e distribuída com a mineração de dados e o estado da arte com algoritmos utilizados nessas áreas;
- Capítulo 3 - Aplicação da computação paralela e distribuída no algoritmo MRFP-ME: Aborda como o autor aplicou a computação paralela e distribuída que visa uma eficiência e melhoria de desempenho do algoritmo MRFP-ME;
- Capítulo 4 - Testes e Resultados: Neste capítulo é apresentado os testes realizados pelo autor e o resultados obtidos e comparações em relação ao algoritmo sem a computação paralela e distribuída;

- Capítulo 5 - Conclusão: É abordado as conclusões relacionadas ao desenvolvimento do trabalho, assim como os trabalhos futuros e contribuições que o trabalho trouxe.

2 Mineração de Dados

Neste capítulo são efetuadas as apresentações dos principais conceitos, algoritmos nos seguintes temas: mineração de dados baseada em regras de associação, algoritmos de mineração de dados com uso de múltiplos suportes mínimos, mineração de dados multi-relacional e a mineração de dados paralela. Estes conceitos são parte importante dos amplos conceitos que envolvem o tema mineração de dados, mas que são direcionados para esta categoria de algoritmos, os multi-relacionais baseados em regras de associação, uma sub-área com suas importantes contribuições.

2.1 KDD e Mineração de Dados

Com a era da informatização, são gerados e armazenados muitos dados diariamente. Com isso, veio a necessidade de se criar e utilizar métodos para extrair informações úteis desse montante de dados. Por conta disso, foi criado o KDD, um conjunto de passos com várias etapas para a extração de dados dessas grandes bases de dados (HAN; KAMBER; PEI, 2011).

Segundo (HAN; KAMBER; PEI, 2011) o KDD é dividido nas seguintes etapas: Limpeza de Dados, Integração dos Dados, Seleção dos Dados, Transformação dos Dados, Mineração dos Dados, Avaliação dos Padrões e Apresentação dos Conhecimento extraídos.

- Limpeza de Dados: Nesta etapa as inconsistências dos dados são removidas, é onde literalmente os dados são limpos;
- Integração dos Dados: É nesta etapa que se pode unir os dados de vários bancos de dados para a realização das análises;
- Seleção dos Dados: São selecionados os dados mais importantes dos bancos de dados para a realização das análises;
- Transformação dos Dados: Os dados são transformados para o padrão para a realização das análises;
- Mineração de Dados: Etapa em que os padrões são extraídos;
- Avaliação dos Padrões: Aqui é onde é feita a avaliação dos dados a partir do resultado obtido na etapa anterior, analisando se os modelos e padrões obtidos são relevantes;

- Apresentação dos Conhecimentos extraídos: Por fim, onde o conhecimento é apresentado para o usuário final.

A mineração de dados é considerada a parte mais importante dentro do KDD, pois é neste processo que existem os algoritmos para descobrir informações dos dados e a geração de padrões para serem feitas as análises (MAIMON; ROKACH, 2005).

Com a mineração de dados é possível identificar padrões e modelos de vários conjuntos de dados. Com isso, esse método pode ser dividido em dois tipos: descritivos e preditivos. Os descritivos destinam-se às propriedades relacionadas aos dados em si. Já os preditivos, como o nome já diz, busca fazer predições dos dados a partir dos dados iniciais (HAN; KAMBER; PEI, 2011).

2.2 Mineração de regras de associação

A mineração de regras de associação ajuda a extrair conhecimento útil em dados que apresentam uma associação direta ou indireta, mas que não são facilmente perceptíveis aos olhos humanos (VALENCIO et al., 2018). Ou seja, a mineração de regras de associação busca efetuar a análise de afinidades, em que contempla regras e ou associações de itens em conjuntos de dados, por meio de ocorrências de determinado item que implique na ocorrência de outro item. Por exemplo, busca-se objetos que não estão relacionados, que não são da mesma categoria e que ocorrem juntos e com a mesma frequência. Esse exemplo é conhecido como análise da cesta de compras (OLSON; LAUHOFF, 2019).

Nas regras de associação temos que X e Y são conjuntos contendo k-itens dentro de uma base de dados, a regra de associação implica em padrões no formato de $X \rightarrow Y$, onde indica que se X ocorrer, Y também ocorre (OLSON; LAUHOFF, 2019). Pode-se utilizar estatísticas que auxiliam em mostrar o quão significativo é determinada regra de associação. A saber: confiança e suporte (OLSON; LAUHOFF, 2019).

Para calcular o suporte de uma regra de associação, $X \rightarrow Y$, podemos usar a seguinte fórmula:

$$sup\ (XY) = \frac{QTDTC\ X \cup Y}{QTDTBD} \quad (1)$$

Nesta fórmula tem-se que QTDTC é igual a quantidade de transações que contenham na união dos conjuntos X e Y e QTDTBD é a quantidade de transações na base de dados, onde o resultado do suporte constitui a quantidade de transações que abrange os itens de X e Y (OLSON; LAUHOFF, 2019).

Pode-se medir a confiança de uma regra de associação por meio da seguinte fórmula:

$$(XY) = \frac{sup(XY)}{sup(X)} \quad (2)$$

Ou seja, a confiança compõe a proporção de transações que contêm na relação de X e Y relacionados ao suporte de X. Ao determinar esses valores, de suporte e confiança, é possível excluir regras que não serão úteis, ou seja, insignificantes (BOUDANE et al., 2016).

Entretanto, tem-se algumas ressalvas com o uso da mineração de regras de associação, destacando-se duas. A primeira é a dificuldade em mostrar que todas as combinações vão ter suportes maiores que o suporte mínimo desejado. O segundo está relacionado a confiança, em que a partir das combinações, é possível definir regras que detêm valores de confiança maiores que a confiança mínima (DARRAB; ERGENC, 2017).

O primeiro problema necessita de um poder computacional elevado para ser solucionado. Por conta disso, há um foco maior em criar algoritmos de padrões frequentes de itens a partir de um banco de dados (DARRAB; ERGENC, 2017). É necessário que se utilize valores diferentes de suporte mínimo para cada item da base de dados para ser feita a mineração, pois definir um único valor de suporte mínimo para todos os itens da base de dados gera problemas na utilização do suporte mínimo, uma vez que a utilização disso visa reduzir o espaço de busca e custo computacional (DARRAB; ERGENC, 2017).

2.3 Mineração com uso de múltiplos suportes mínimos

Como mencionado anteriormente, para resolver o problema de utilizar apenas um suporte único é necessário a utilização de múltiplos suportes mínimos. Com isso, é possível diferenciar os dados entre si e resolver o ‘problema do item raro’ (GAN et al., 2017). Caso seja definido um valor muito grande para o suporte mínimo, não terão regras para itens que possivelmente pouco aparecem em determinada base de dados. Por outro lado, se o valor do

suporte mínimo for um valor pequeno, pode-se gerar regras insignificantes em um número muito grande (DARRAB; ERGENC, 2017).

Com o uso da mineração de dados com múltiplos suportes mínimos, foi estipulado um valor de suporte mínimo para cada item do conjunto de dados. Esse valor é conhecido como MIS (*Minimum Itemset Support*), ou melhor, suporte mínimo do conjunto de itens. Com isto, ao utilizar regras com itens muito frequentes, é possível utilizar suporte mínimos maiores e no caso do 'problema do item raro', é possível utilizar suportes mínimos menores(GAN et al., 2017).

Para calcular esse valor, é necessário a frequência que determinado item ocorre (DARRAB; ERGENC, 2017). Darrab e Ergenc (2017) demonstraram a maneira de como calcular o valor de MIS, conforme a equação abaixo.

$$MIS(item) = max\ (\beta \times f(item_i), LS) \qquad (3)$$

Sendo $f(item_i)$, a quantidade de transações que determinado item contêm, no caso o $item_i$.

Com isso, o pesquisador poderá determinar os valores de β e LS (*Lowest Support*). LS significa o menor valor de suporte permitido para os itens. O valor de β representa a forma que o valor MIS está diretamente relacionado a frequência do seu item, sendo definido da forma que $0 \leq \beta \leq 1$. Com β sendo igual a 0, temos que o valor de LS será o mesmo para toda a base de dados, iguais aos algoritmos tradicionais. Por outro lado, β sendo maior que 0, temos MIS sendo o maior valor entre $\beta \times f(item_i) e\ LS$.

Um dos primeiros algoritmos como a utilização de múltiplos suportes mínimos foi proposto por Liu, Hsu e Ma (1999). O algoritmo é conhecido como MSApriore, o qual é um algoritmo estendido do Apriori.

2.3.1 MSApriori

O algoritmo MSApriori foi proposto para que funcione de maneira análoga ao Apriori, pois define um mesmo valor de suporte mínimo para todos os itens do conjunto de dados. Por ser um algoritmo de mineração de regras de associação com o uso de múltiplos suportes mínimos, ordena os conjuntos em ordem crescente de acordo com o MIS.

Tem-se como no primeiro passo do algoritmo que os itens são discorridos em ordem crescente do MIS, como mencionado anteriormente, e se não for adequado, é retirado. Uma vez que a contagem do suporte de cada item é realizada e é gerado o conjunto F de acordo com a seguinte equação: $F = \{item_i.sup \geq MIS(item_i), i = 1\ item_i.sup \geq MIS(item_i - 1), i > 1\}$. Após a criação do conjunto F, é criado o conjunto L_1.

O segundo passo do algoritmo é gerado um conjunto de candidatos C_2 a partir de cada item do conjunto F, se $item_i.sup \geq MIS(item_i)$ e $item_j.sup \geq MIS(item_j)$, com $MIS(item_j) > MIS(item_j)$, então $\{item_i, item_j\} \in C_2$. Após isso, é feita a análise se $c.sup \geq MIS(c)$, se for verdadeiro, o conjunto é frequente é adicionado ao conjunto L_2.

Finalmente, os passos a seguir do algoritmo são executados de formas diferentes aos anteriores. Com $k > 2$, tem-se que são gerados conjuntos de candidatos C_k junto com a junção dos conjuntos K_{k-1}e K_{k-1}. Com isso, é realizado a retaliação dos conjuntos com itens não frequentes, ou seja, $c.sup < MIS(c)$. Por fim, um conjunto L_k é gerado igualmente ao conjunto L_2, quando o conjunto L_k se torna vazio, o algoritmo é finalizado.

O algoritmo MSApriori apresenta alguns problemas relacionados ao uso de memórias e consumo do tempo, apesar de resolver o problema do item raro, ainda é necessário soluções que resolvam problemas como descrito acima (GAN et al., 2017).

2.3.2 FP-ME

Frequent patterns with Multiple Minimum Supports from the Set-Enumeration-tree (FP-ME), este algoritmo foi proposto por Gan et al (2017) sendo um algoritmo no qual utiliza múltiplos suportes mínimos para os seus itens. Este algoritmo se diferencia no MSApriori pelo falto de que o Apriore utiliza testes para poder escolher padrões frequentes. Já o FP-ME, usa uma estrutura conhecida como ME-tree, na qual poussui os elementos nos nós filhos e ordenados pelo valor do MIS. Por conta disso, o tempo de execução e consumo de memória são diminuídos.

Com o objetivo de diminuir o espaço de busca e manter a integridade dos resultados, foi introduzido o conceito conhecido como LMS (do inglês, *Least Minimum Support*), no qual se refere ao menor valor do suporte mínimo entre os padrões frequentes, ou seja, o LMS é o menor valor MIS de todos os itens. O LMS contribui para determinar o valor MIS.

Primeiramente o algoritmo realiza uma varredura na base de dados e considera os itens com o $sup\ sup\ (i) \geq LMS$, sendo realiza a busca dos padrões frequentes com esses itens. Com isso, uma nova varredura é realizada para determinar o *TidSet* para cada item da base de dados. Com isso, a ME-tree é construída, com cada nó filho sendo inserido em ordem crescente do MIS e cada nó sendo constituído do *TidSet*, MIS e sup.

A última etapa se refere em recursividade, utilizando a ME-tree da primeira etapa. Cada chamada da recursividade é utilizado um nó pai *Z* e os seus respectivos nós filhos. Com isso, um nó filho de *Z* se refere ao item *ZX*, sendo *X* os diferentes conjuntos de 1-item que são encontrados na etapa um. Portanto, para cada nó filho de *Z* é analisado se o suporte do item é maior ou igual ao MIS vezes a quantidade de transações na base de dados, caso isso realmente aconteça, o nó pai é relacionado a um padrão frequente e adicionado aos conjuntos dos padrões frequentes que são encontrados. Caso contrário, esse conjunto não precisará ser executado pelo algoritmo. Com isso, a partir dos padrões frequentes encontrados, serão geradas todas as suas possíveis extensões.

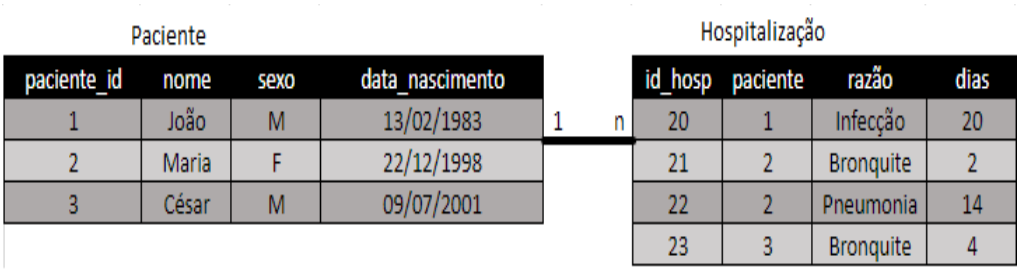
A grande vantagem entre os outros algoritmos correlatos na literatura, é que o custo de armazenamento da estrutura é reduzido, pelo fato de que os nós da árvore não estejam em memória, e somente os nós com 1 item são alocados de acordo com a necessidade e utilização. Isso se dá por conta de que a mineração é realizada a partir dos nós pais em direção aos nós filhos. Outro motivo é que a etapa 2 do algoritmo não permite a criação de nós duplicados, reduzindo assim o consumo de memória (GAN et al., 2017).

2.4 Mineração de dados multi relacional

A mineração de dados multi-relacional aborda todas as relações das estruturas de interesse, igualmente a um banco de dados relacional, diferentemente de como ocorre nas abordagens tradicionais, uma vez que apenas consideram uma única estrutura, como por exemplo, a tabela de um banco de dados relacional. (VALENCIO et al., 2018). Nos últimos anos, tem-se utilizado muitas bases de dados relacionais, por conta disso, é necessário a aplicações de algoritmos de mineração de dados nesses ambientes também. Por conta disso, foi proposto técnicas para reduzir as múltiplas estruturas e relações em apenas uma estrutura (VALENCIO et al., 2018).

Conforme ilustrado na Figura 1, é possível observar uma estrutura com duas tabelas relacionadas. Conforme é possível ver na tabela, a relação é 1 para n, ou seja, uma tupla da tabela paciente pode se relacionar com várias tuplas da tabela internação. Por outro lado, várias tuplas da tabela internação podem se relacionar com apenas uma tupla da tabela paciente.

Figura 1 - Tabelas paciente - hospitalização



Paciente

paciente_id	nome	sexo	data_nascimento
1	João	M	13/02/1983
2	Maria	F	22/12/1998
3	César	M	09/07/2001

1 n

Hospitalização

id_hosp	paciente	razão	dias
20	1	Infecção	20
21	2	Bronquite	2
22	2	Pneumonia	14
23	3	Bronquite	4

Fonte: Adaptado de VALÊNCIO et al. (2012)

Já na Figura 2, é possível observar a junção das duas tabelas vistas na Figura 1, sendo o *paciente_id* como propriedade para a junção. Contudo, a simples junção dessas tabelas pode gerar um problema, redundância dos dados. No exemplo da figura 2 os campos *nome*, *sexo*, *data_nascimento* do paciente 2 foram duplicados, pois este paciente teve mais de uma hospitalização.

Figura 2 - Junção tabelas paciente - hospitalização

paciente_id	nome	sexo	data_nascimento	id_hosp	razão	dias
1	João	M	13/02/1983	20	Infecção	20
2	Maria	F	22/12/1998	21	Bronquite	2
2	Maria	F	22/12/1998	22	Pneumonia	14
3	César	M	09/07/2001	23	Bronquite	4

Fonte: Adaptado de VALÊNCIO et al. (2012)

Ao observar a Figura 2, pode-se perceber que o paciente com o *id* igual 2, foi duplicado. Isso ocorre pelo fato de aplicar a técnica de junção na tabela, pois na tabela de Hospitalização da Figura 1, o paciente possuía duas razões de internação. Com o propósito de eliminar essa redundância dos dados, é possível utilizar funções de agregação na tabela da Figura 2. Pode-se observar o resultado da utilização dessas funções na Figura 3. Com isso, foram criadas duas novas colunas na tabela, *num_hosp* e *media_dias*. A primeira coluna refere-se a quantidade de hospitalizações que o paciente teve internado, já a segunda, a média de dias em que o paciente ficou hospitalizado. Por outro lado, ainda assim ocorre um problema ao fazer a agregação dos dados, a perda deles. Na Figura 3, com a agregação dos dados, o número exatamente de dias em que o paciente 2 ficou hospitalizado foi perdido, uma vez que foi substituído pela média dos dias. Na tabela da Figura 2, tinha a informação de que o paciente 2 ficou hospitalizado 2 dias e depois 14 dias, enquanto na tabela da Figura 3 só é mostrado 8 dias, enquanto que os outros pacientes não perderam essa informação pois só foram hospitalizados uma única vez.

Figura 3 - Agregação tabelas paciente - hospitalização

paciente_id	nome	sexo	data_nascimento	num_hosp	media_dias
1	João	M	13/02/1983	1	20
2	Maria	F	22/12/1998	2	8
3	César	M	09/07/2001	1	4

Fonte: Adaptado de VALÊNCIO et al. (2012)

Com os exemplos acima, observa-se que a utilização dessas técnicas, como a junção e agregação dos dados, traz alguns problemas, por exemplo a redundância e perda dos dados. Embora essas técnicas sejam utilizadas em ambientes com bases de dados relacionais, é necessário técnicas de mineração de dados nesses tipos de ambientes em que tais problemas não ocorram (VALENCIO et al., 2018).

Valêncio et al. (2011) propôs a criação de um modelo que armazene a tabela de origem, a coluna e o valor, com isso, pode-se representar um item relacional da seguinte forma: $TabAtrib = Valor$. Por outro lado, um único item pode ser citado várias vezes e por conta disso, isso pode ocupar muito espaço em memória, uma vez que em um banco de dados podem haver várias transações e características. Portanto há a necessidade de um método mais eficiente (VALÊNCIO et al. 2011). Por conta disso, Valêncio et al. (2011) também apresentou uma estrutura que visa diminuir a ocupação de espaço de memória, essa estrutura é chamada de

ItemMap, no qual faz o mapeamento de itens relacionais em identificadores, tornando-se possível o manuseio de vários dados de um base de dados.

Uma outra técnica para a mineração de dados multi-relacional, a qual agrupa os itens que contêm o mesmo identificador da transação, conhecida como FSI (*Fusion of Secondary Items*, ou em português, fusão de itens secundários). Essa técnica é muito importante pois mantém a semântica da base de dados. Por conta disso, ao utilizar esse tipo de técnica, ao selecionar determinado item, outros itens que possuem o mesmo identificador de transação irão estar dentro do processo (VALÊNCIO et al. 2012).

Um algoritmo muito importante que aborda esses conceitos de mineração de dados multi-relacional para dados relacionais, é o MR-Radix, proposto por Valêncio et al. (2012). Esse algoritmo trouxe uma grande vantagem, a redução do custo de armazenamento em 75%. Isso só foi possível pois é utilizado uma estrutura conhecida como Radix-tree, o qual agrupa uma quantidade de nós em um único ramo e esses nós possuem o mesmo valor de suporte (VALÊNCIO et al. 2012).

2.5 Mineração de dados paralela

Com a alta escalabilidade e o grande tamanhos dos banco de dados atuais, torna-se custoso o uso de algoritmos de mineração de dados, o que é um dos maiores problemas com a mineração de dados. Por conta disso, a computação paralela vem se tornando cada vez mais viável em algoritmos de mineração de dados, pois lida com a questão da eficiência desses algoritmos em bases de dados com grandes volumes (FREITAS, 1996).

Existem algumas formas de se usar o paralelismo, uma delas é o paralelismo de dados e o paralelismo de controle (FREITAS, 1996). O primeiro, o paralelismo de dados, está relacionado em executar a mesma instrução em cima dos subconjuntos do montante de dados, por outro lado, o paralelismo de controle, refere-se a execução de várias instruções concorrentemente (TANIAR; RAHAYU, 2002).

Apesar de ter essas duas abordagens, o paralelismo de dados se sobrepõem em relação ao paralelismo de controle. O paralelismo de dados utiliza o mesmo controle de fluxo do que a abordagem tradicional dos algoritmos. Portanto, o código serial pode ser reaproveitado, uma vez que apenas o acesso aos dados é paralelizado, simplificando drasticamente o tempo de programação (TANIAR; RAHAYU, 2002).

Outra grande vantagem do paralelismo de dados é a grande utilidade e escalabilidade em grandes conjuntos de dados. Com esse tipo de abordagem, é possível manter o tempo de respostas do processamento igual mesmo se a quantidade de dados aumenta na base de dados. Ou seja, quanto maior a quantidade de dados, mais viável se torna o uso desse tipo de abordagem (TANIAR; RAHAYU, 2002).

Existem vários algoritmos de mineração de dados que utilizam a computação paralela, especificamente baseados no algoritmo Apriori. Um Apriori generalizado, baseado na DIC (do inglês, Dynamic Itemset Counting), divide o banco de dados em pequenas partições com a finalidade de reunir os suportes individuais de cada item da base de dados alocando-os na primeira partição. Os itens frequentes encontrados são usados para gerar os candidatos para que, ao ler a segunda partição, os suportes desses candidatos são obtidos. Isso é feito até chegar na última partição que foi gerada. Todo esse processo só termina quando o suporte global de um candidato é conhecido, para isso, é preciso que todo o processamento discorre toda a base de dados e chega na última partição onde o suporte foi gerado pela primeira vez (TANIAR; RAHAYU, 2002).

Com isso, o objetivo de se usar a computação paralela em algoritmos de mineração de dados, é a melhoria de desempenho. Para que isso ocorra, tem que se levar em conta a quantidade de tarefas que podem ser realizadas em um determinado intervalo de tempo e a quantidade de tempo que leva para determinada tarefa ser concluída (TANIAR; RAHAYU, 2002).

2.6 Trabalhos Correlatos

Com o propósito de apresentar o atual estado da arte no que se refere aos temas mineração multi-relacional, uso de múltiplos suportes mínimos e paralelismo, os principais trabalhos da literatura nestes temas são descritos nesta seção.

Como já citado por este trabalho, Valêncio et al. (2012) propôs uma abordagem que utilize a mineração de dados multi-relacional, cujo recebeu o nome MR-Radix. Valêncio et al. (2018) propôs também uma extensão do algoritmo MR-Radix, nomeado de TBMR-Radix, o qual usa *templates* para definir as regras de associações geradas. Com isso, é possível diminuir

a quantidade de resultados irrelevantes e fazer uma análise mais orientada por parte do pesquisador (VALÊNCIO et al., 2018).

Uma abordagem multi-relacional proposta por Rocha et al. (2018), explora a programação lógica indutiva, na qual se refere ao método de aprendizado de máquina supervisionado que, a partir de um conhecimento prévio, é possível determinar hipóteses sobre os dados relacionados (ROCHA et al., 2018).

Nagao e Seki (2015) propuseram um algoritmo paralelo para padrões fechados em base de dados multi-relacionais, no qual mostraram que a abordagem tradicional não funcionavam bem. Chamada de subárvores de paralelização, geraram conjuntos de subárvores até determinada profundidade, chamada de *depth limit*, quando isso, o cálculo dos padrões fechados era calculado em paralelo em cada subárvore gerada. Porém, ao realizar testes, constataram que esse método ainda possuía um mal desempenho ao aplicar em dados multi-relacionais. Com isso, propuseram uma nova abordagem chamada de paralelização por nós, no qual a busca era realizada para cada nó de uma árvore, sendo aplicado a busca em paralelo.

Abdel-Basset et al. (2018) propuseram um algoritmo de mineração de dados com o uso de regras de associação para descobrir possíveis regras de associações com objetivo de minimizar a perda de regras de associação, cria-se assim, um sistema mais eficiente e eficaz para a tomada de decisão. Ao comparar o trabalho com abordagens difusas, notou-se que mais regras de associação eram descobertas e regras não utilizáveis também, sendo possível separar essas regras para serem aproveitadas da melhor maneira possível.

Em relação ao uso de múltiplos suportes mínimos, pode-se destacar os trabalhos que Liu, Hsu e Ma (1999) e Gan et al. (2017) propuseram. Vale destacar o trabalho de Gan et al. (2017), no qual propuseram um novo modelo de mineração ao projetar um algoritmo com que cada item tenha seu próprio suporte mínimo. Além disso, trouxeram como contribuições a garantia da exatidão e integridades dos resultados obtidos ao utilizar a ME-tree, no qual também reduz o tempo de execução e o consumo de memória. Outra contribuição foi o conceito de DiffSet, no qual remove padrões poucos promissores, diminuindo o tempo de execução. Por fim, o algoritmo resolve o "problema do item raro". Já Wang, Lin e Chang (2017) propuseram o uso da computação paralela para que esses tipos de algoritmos pudessem lidar com grandes volumes de dados com a utilização do *framework MapReduce*.

Publicado por Darrab e Ergenc (2017), este trabalho se refere ao problema do item raro, apresentando uma árvore chamada MIS-eclat. Para determinar os padrões frequentes, é utilizado o *bottom-up*, isso faz com que a extração se inicia nos nós filhos da árevore e percorre até os nós pais da árvore, tendo como resultado um tempo de execução e consumo de memória melhores aos correlatos na literatura.

O algoritmo MRF-ME (2018) desenvolvido por (MARTINEZ, 2018), apresentou mudanças significativas ao unir abordagens já utilizadas na literatura. Esse algoritmo, trouxe a junção e utilização de um algoritmo de mineração de regras de associação multi-relacional com o uso de múltiplos suportes mínimos

Com isso, é possível observar a Tabela 1 onde estão apresentados os trabalhos correlatos, onde é possível identificar trabalhos que abordam a mineração multi-relacional, utilização de múltiplos suportes e a computação paralela em uma das abordagens. Porém, não há nenhuma proposta que utilize ambas as abordagens em uma só junto com a computação paralela.

Tabela 1 - Comparação dos trabalhos correlatos

	Múltiplos suportes mínimos	Multi-relacional	Paralelismo
Rocha et al. (2018)	✗	✓	✗
Darrab e Ergenc (2017)	✓	✗	✗
Wang, Lin e Chang(2017)	✓	✗	✓
Nagao e Seki (2015)	✗	✓	✓
Valêncio et al. (2012)	✗	✓	✗
Valêncio et al. (2018)	✗	✓	✗
(MARTINEZ, 2018)	✓	✓	✗
Liu, Hsu e Ma (1999)	✓	✗	✗

Fonte: Elaborado pelo autor.

2.7 Considerações finais

Como visto nas seções anteriores, as abordagens tradicionais não são tão eficientes e trazem alguns problemas em bases relacionais, como por exemplo problemas ao realizar as

técnicas de junção e agregação. Por conta disso, algoritmos de mineração de dados multi-relacional resolvem tais problemas apresentados

Contudo, ao utilizar apenas um único suporte mínimo, encontra-se um problema conhecido como ‘problema do item raro’, na qual gera associações com regras irrelevantes ou perda de dados que pouco aparecem na base de dados. Ao utilizar múltiplos suportes mínimos, ou seja, um suporte mínimo para cada item da base de dados, esse problema é resolvido.

Entretanto, esses algoritmos acabam sendo custosos ao serem aplicados em grandes bases de dados, sendo necessário otimizar esses algoritmos. Uma técnica utilizada para isso, é a computação paralela, na qual melhora o tempo de execução desses algoritmos e também o consumo de memória

3 Algoritmo Proposto

Neste capítulo são apresentados os elementos que compõe a proposta do algoritmo MRFP-ME.

3.1 Algoritmo MRFP-ME

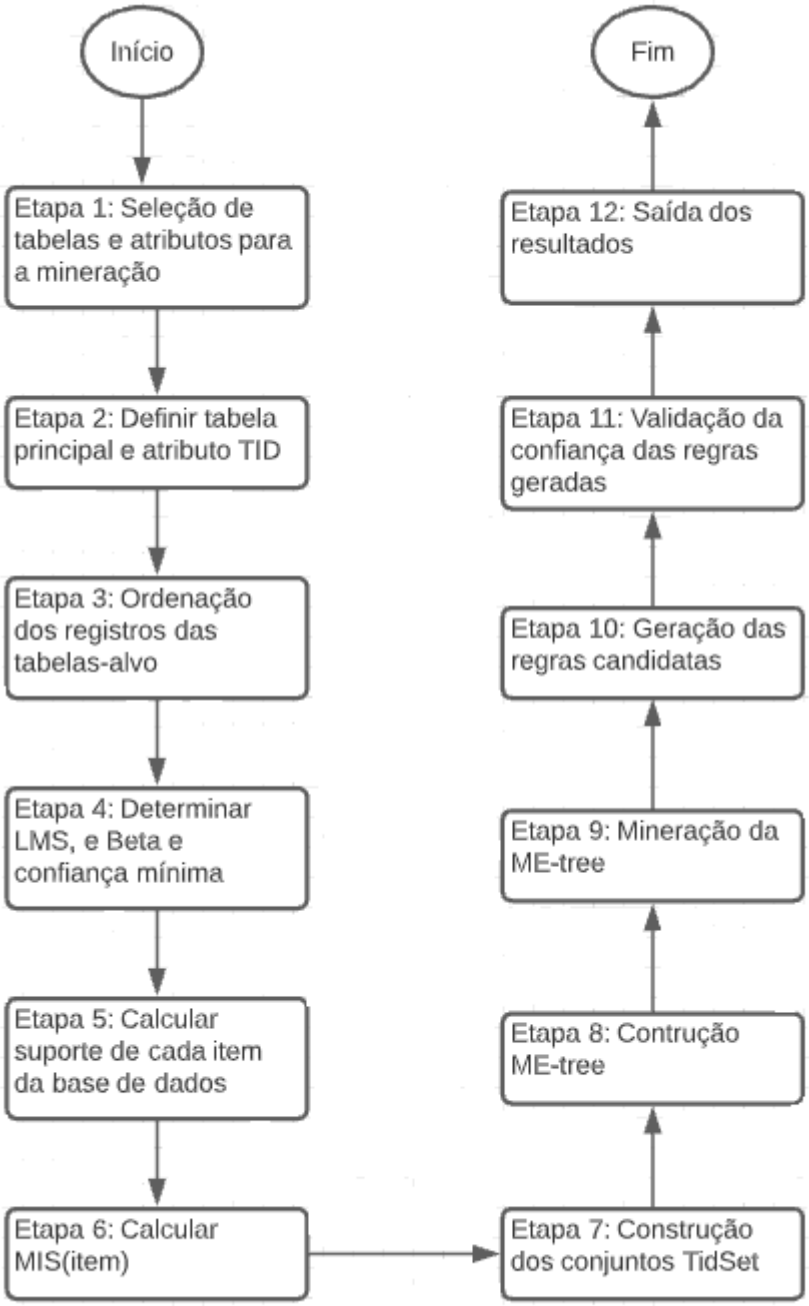
O algoritmo da literatura, no qual este trabalho se baseou para propor a versão que contempla os conceitos de computação paralela, é o algoritmo MRFP-ME. Esse algoritmo aborda conceitos de mineração de dados multi-relacional e o uso de múltiplos suportes mínimos.

Proposto por Gan et al. (2017), esse algoritmo utiliza a estrutura ME-tree para o uso de múltiplos suportes mínimos. Para que não haja uma maneira errônea de se representar os itens relacionais, é utilizado a notação *Tab.Atributo = Valor* (VALÊNCIO et al., 2011). Para reduzir o consumo de memória, é utilizado o conceito de *ItemMap*, como mencionado nas seções anteriores (VALÊNCIO et al., 2011). E para que haja semântica entre os itens, é utilizado o conceito de FSI (VALÊNCIO et al. 2012).

O algoritmo ME-tree, utiliza apenas o método de mineração de itens frequentes. Uma vez que um algoritmo de regras de associação é dividido com conjuntos com itens frequentes e a geração de regras de associação, o ME-tree não gera regras de associação, por conta disso, foi utilizado um método semelhante ao proposto por Agrawal et al. (1993) para a geração de regras de associação no MRFP-ME.

Na Figura 4 é mostrado um fluxograma das etapas que compõem o algoritmo MRFP-ME.

Figura 4 - Fluxograma do algoritmo MRFP-ME



Fonte: Elaborado pelo autor.

Como descrito no fluxograma, a Etapa 1 consiste em selecionar a tabela e os atributos para a realização da mineração. A Etapa 2 consiste em definir a tabela principal que será utilizada e o atributo com o identificador de transação, sendo nesta etapa a aplicação do FSI, apresentado por Valêncio et al. (2012). A Etapa 3 consiste em ordenar os dados em ordem

crescente utilizando o identificador de transação como parâmetro. A última etapa antes do algoritmo começar, é a etapa 4. As Etapas 5 a 9, correspondem a mineração de dados de padrões frequentes. A Etapa 5 calcula o suporte mínimo de cada item e é nesta etapa em que o FSI é utilizado Já na Etapa 6 o valor de MIS de cada item é calculado. Na Etapa 7 é construído o conjunto de identificadores de transação. As Etapas 8 e 9 onde acontece a construção e mineração da ME-tree de acordo com Gan et al. (2017). As etapas 10 e 11 constituem na geração das regras de associação de acordo com os padrões encontrados nas etapas anteriores. E por fim, a Etapa 12 refere-se à apresentação dos resultados obtidos com o processo de mineração.

3.1.1 Mineração de padrões frequentes

Esta parte refere-se em determinar os padrões de ocorrências entre os itens da base de dados (DARRAB; ERGENC, 2017). Neste algoritmo, esta etapa é dividida em construir a ME-tree e a mineração da mesma.

3.1.1.1 Construção da ME-tree

Nesta parte, já se tem as tabelas e seus atributos selecionados e a base de dados é entendida como entrada do algoritmo. Vale ressaltar que β e *LMS* já são determinados pelo pesquisador e os dados da base de dados são ordenados em ordem crescente pelo valor do *TID*. Com isso, a primeira leitura do algoritmo é realizada. O processamento dos itens na tabela primária ocorre de modo que os itens singulares achados, são adicionados ao *ItemMap*, assim, todos os itens repetidos encontrados, o seu suporte no *ItemMap* é apenas incrementado. Diferentemente das tabelas primárias, o tratamento das tabelas secundárias é feito verificando o *TID* do item processado, uma vez que o suporte tem que ser incrementado com *TID* diferentes. O algoritmo prevê que instâncias repetidas de um item não seja processador, para isso, uma variável auxilia na construção da estrutura do *ItemMap*. Com isso, a técnica FSI é aplicada para a fusão entre os itens das tabelas utilizadas, uma vez que os itens são ordenados em ordem crescente pelo TID, os próximos TID deste item serão maiores ou iguais a variável auxiliar.

Após isso, os valores dos suportes são os valores definitivos, determinando o número de ocorrências de cada item final. Por outro lado, o valor do LMS ainda não é definitivo, sendo então necessário determinar o valor de LMS. Para isso, basta multiplicar o valor do LMS ainda não definitivo pela quantidade de registros na tabela principal.

O cálculo do MIS é efetuado verificando se o suporte mínimo é maior ou igual ao valor do LMS de acordo com cada item inserido no *ItemMap*. Caso seja verdadeiro, o valor de MIS é calculado. Caso contrário, esse item não é frequente e conjuntos que possuem esse item também não é frequento, portanto, removido do *ItemMap*, sendo necessário reordenar os itens pelo valor de MIS.

A segunda leitura determina o TidSet para cada item que está contido no ItemMap. Isso é feito de maneira que, para os itens da tabela primária é verificado se o mesmo está contido no ItemMap, se estiver, o TID é inserido no TidSet. Para os itens da tabela secundária, é veiricado se o TID já está no TidSet, para isso é utilizado a variável auxiliar utilizado na primeira etapa, se o TID não estiver no TidSet, é inserido no TidSet.

Após esta etapa, todos os itens adicionados no ItemMap, são inseridos como nós na árvore, tendo cada nó o conjunto de itens, MIS, o sup e o TidSet. Pelo fato da mineração ocorrer de forma que vai no sentido do nó pai para os nós filhos, não é necessário gerar todos os nós, pois eles serão gerados de acordo com a mineração da árvore é realizada. Por fim, para que o processo de mineração seja iniciado, o ItemMap e a ME-tree são retornados.

Figura 5 – Pseudocódigo da Mineração da ME-tree

Algoritmo: Construção da ME-tree
Entradas: Base de dados D, LMS, β;
Saída: ItemMap IM, ME-tree $MEtree$;

```
1)  para cada tabela T de D faça
2)      se T é tabela principal então
3)          para cada item I em T faça
4)              se I ⊂ IM então
5)              IM[posicao(I, IM)].suporte++
6)          senão
7)              adicionar I ao IM
8)              IM[posicao(I, IM)].suporte = 1
9)      senão
10)         para cada item I em T faça
11)             se I ⊂ IM então
12)                 se I.tid ≠ IM[posicao(I, IM)].tid então
13)                     IM[posicao(I, IM)].TID = I.tid
14)                     IM[posicao(I, IM)].suporte++
15)             senão
16)                 adicionar I ao IM
17)                 IM[posicao(I, IM)].suporte = 1
18)                 IM[posicao(I, IM)].TID = I.tid
19) LMS = (número de registros da tabela primária) * LMS
20) para j = 1 até tamanho(IM) faça
21)     se IM[j].suporte ≥ LMS então
22)         IM[j].MIS = max (β * IM[j].suporte, LMS )
23)         IM[j].TID = 0
24)     senão
25)         remover IM[j] do IM
26) ordenar IM de forma crescente de valor MIS
27) para cada tabela T de D faça
28)     se T é tabela principal então
29)         para cada item I em T faça
30)             se I ⊂ IM então
31)                 adicionar I.tid a IM[posicao(I, IM)].TidSet
32)     senão
33)         para cada item I em T faça
34)             se pertence IM então
35)                 se I.tid ≠ IM[posicao(I, IM)].tid então
36)                     adicionar I.tid a IM[posicao(I, IM)].TidSet
37)                     IM[posicao(I, IM)].TID = I.tid
38) para j = 1 até tamanho(IM) faça
39)     N.ItemSet = j
40)     N.TidSet = IM[j].TidSet
41)     N.suporte = IM[j].suporte
42)     adicionar N em MEtree.nós
43) retorne MEtree, IM
```

Fonte: Adaptado de (MARTINEZ, 2018).

3.1.1.2 Mineração da ME-tree

Esta parte do algoritmo é utilizado recursividade em suas chamadas. Esta parte, recebe para o processo o ItemMap e Me-tree que foram gerados na etapa da construção da ME-tree. Para cada chamada, uma iteração sob cada nó dentro da Me-tree é realizada, sendo analisado se o conjunto de itens é frequente, para isso, o suporte e o MIS são comparados. O MIS é determinado a partir do MIS da primeira posição do conjunto, como dito anteriormente, o conjunto é ordenado de acordo com o valor de MIS, sendo assim, o primeiro elemento conterá o menos MIS em relação a outros itens (GAN et al., 2017).

Quando o valor do MIS é maior que o suporte, indicando que o nó não tenha itens frequentes, não será necessário realizar o processamento do mesmo (GAN et al., 2017). Com isso, uma outra técnica é aplicada que não haja criação de nós não utilizáveis. Por outro lado, o conjunto de itens é adicionado ao conjunto de saída. Após isso, o índice do ItemMap será obtido a partir do último item do conjunto. Com isso, realizando a iteração do ItemMap, para cada item de determinada posição, é gerado um nó na ME-tree, conforme dito anteriormente, tendo assim um novo conjunto de itens gerados. Por fim, como a função é recursiva, uma nova chamada é realizada com passagens dos parâmetros gerados e enviados para uma próxima execução. Pode-se checar esses passos na Figura 6 apresentado a seguir.

Figura 6 – Pseudocódigo da Mineração da ME-tree

Algoritmo: Mineração da ME-tree
Entradas: ItemMap IM, ME-tree $MEtree$;
Saída: Conjunto dos padrões frequentes PF;
1) declarar função $mineracaoMEtree(CN: conjunto\ de\ nós\ da\ MEtree)$:
2) para cada nó N em CN faça
3) se $N.suporte \geq IM[N.itemset[1]].MIS$ então
4) adicionar $N.ItemSet, N.sup, IM[N.itemset[1]].MIS$ a PF
5) $n = tam(N.ItemSet)$
6) $i = N.ItemSet[n]$
7) para $j = i + 1$ até $tamanho(IM)$ faça
8) $extensaoN.ItemSet = N.ItemSet \cup j$
9) $extensaoN.TidSet = N.TidSet \cap IM[j].TidSet$
10) $extensaoN.sup = tamanho(extensaoN.TidSet)$
11) adicionar $extensaoN$ a $N.extensoesN$
12) $mineracaoMEtree(N.extensoesN)$
13)
14) $PF = \emptyset$
15) $mineracaoMEtree(MEtree.nós)$

Fonte: Adaptado de (MARTINEZ, 2018).

3.2 Regras de Associação

Com as etapas acima descritas, é necessário detalhar quais regras serão uteis para o processo e que de fato, quais regras serão utilizadas, para isso é utilizado o valor de confiança, conforme citado anteriormente. Esta parte do algoritmo, possui como entrada os padrões frequentes encontrados na etapa da ME-tree e seus suportes.

Para cada padrão frequente, é adicionado um valor de suporte utilizando uma tabela hash para armazenar tais valores, pois dessa forma, é possível utilizar o conjunto de itens como chave, permitindo uma busca mais facilitada aos valores do suporte mínimo, otimizando a busca.

Como na etapa de mineração existem padrões que são gerados com apenas um item, é verificado o tamanho do conjunto de itens, uma vez que as regras de associações são geradas com dois itens ou mais.

É realizado uma iteração para poder determinar regras de associações possíveis, para isso, uma marcação em binário é utilizada. Como isso, para cada posição da iteração possuir o valor 1, esse item será adicionado no subconjunto A, e assim, o conjunto será inteiramente composto por padrões frequentes. Após isso, o cálculo da confiança é gerado, e por fim, se caso a confiança gerada seja maior ou igual a confiança mínima, ela será considerada e adicionado ao conjunto de regras de associação utilizada.

3.3 Spark

Com o crescimento dos dados nos últimos anos, foi necessário aprimorar técnicas para processar e analisar esses dados de forma eficiente. Por conta disso, foi pensado algoritmos para realizar tais tarefas utilizando processamento paralelo e em memória (SINGH; REDDY, 2015). Pensando nisso, um dos *frameworks* mais utilizados atualmente para realizar processamento com grandes volumes de dados é o Spark, com ele é possível realizar processamento paralelo e em memória, e ainda disponibiliza computação em *clusters* e diversas outras bibliotecas para processamento de dados (MAVRIDIS; KARATZA, 2017).

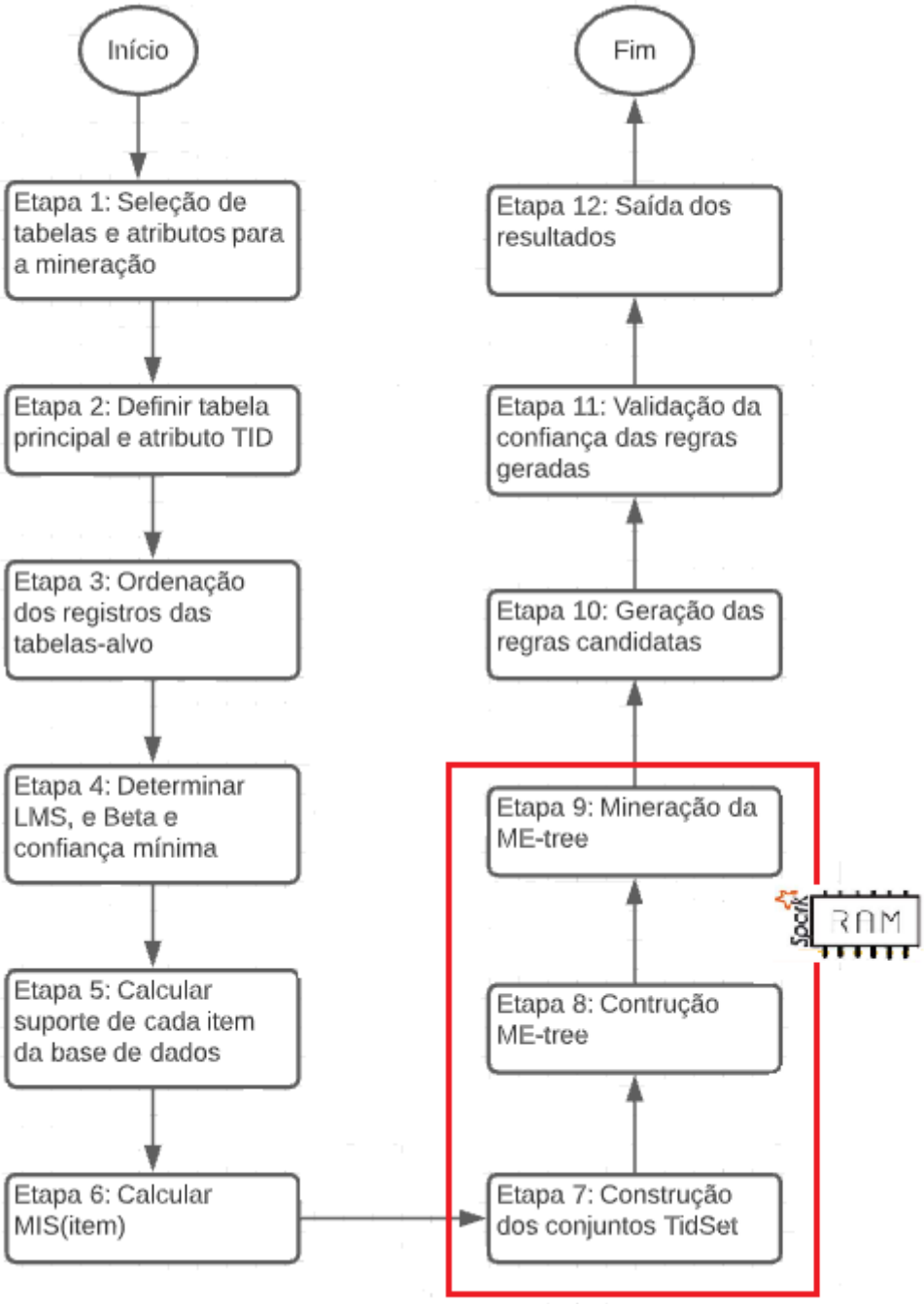
O Spark utiliza abstrações conhecidas como RDD (do inglês, *Resilient Distributed Datasets*), Datasets ou Dataframes, no qual permite o processamento em memória. Com essas abstrações, é possível armazenar os dados nelas e realizar operações do Spark para a confecção

do processamento. Um detalhe importante sobre essas abstrações é que elas são tolerantes a falhas e realizam o armazenamento paralelo dos dados (SHMEIS; JABER, 2019). Essas operações são divididas em dois grupos: transformações e ações. As transformações são responsáveis por transformar um RDD em outro RDD por meio de uma função escolhida, e isso é feito de forma paralela. Já as ações, são operações que manipulam diretamente os dados, permitindo obter e visualizar o resultado obtido ao realizar uma transformação. Uma transformação só é concretizada quando uma ação é realizada, tal motivo explica a importância dessas duas operações serem tão importantes ao utilizar o Spark.

Por utilizar a memória para armazenar resultados, que normalmente seriam gravados em disco, o Spark se torna muito vantajoso, uma vez que diminui o acesso ao disco (SHMEIS; JABER, 2019).

Diante destes conceitos, torna-se viável a utilização do Spark no algoritmo, mais especificamente em partes do algoritmo que possui um alto custo. Adaptado da Figura 4, é mostrado na Figura 6 como ficaria o fluxograma no algoritmo ao utilizar o framework Spark.

Figura 7 – Fluxograma do algoritmo ao utilizar o Spark.



Fonte: Elaborado pelo autor.

3.4 Considerações finais

Neste capítulo foi apresentado a metodologia utilizada para a construção do algoritmo MRFP-ME abordando a computação paralela, ou seja, foi utilizado o *framework* Spark para escalonar o algoritmo e paralisá-lo.

4 Avaliação Experimental

Neste capítulo são apresentadas as abordagens utilizadas para efetuar os testes do algoritmo, com o propósito de avaliar a eficiência e o desempenho desta proposição de algoritmo, com os recursos de computação paralela e as características para a mineração de dados que buscam ampliar a semântica das regras de associação em relação ao respectivo algoritmo da literatura.

4.1 Metodologia de experimentação

Os experimentos foram realizados na base de dados do Sistema de Informação e Vigilância de Acidentes de Trabalho (SIVAT), fornecido e mantido pelo GBD - Grupo de Banco de Dados – Ibilce - UNESP para fins acadêmicos. O motivo de utilizar essa base de dados para os testes, é que possui um bom número de registros, as relações apresentam consideráveis ocorrências de relacionamentos implícitos e, consequentemente, um número importante de dados oriundos de outras relações, além da considerada a principal.

Neste experimento, o valor de *LMS* foi definido sendo LMS = 10% e β foi definido em um intervalo entre [0, 1] e um valor de 40% para confiança mínima. O LMS se refere ao menor valor de suporte mínimo, já o β, a frequência dos itens. Com isso, as métricas coletadas e utilizadas para a análise foram o consumo de memória e tempo de execução, sendo comparado o MRFP-ME com o MRF-ME que utiliza a abordagem paralela.

A mineração de padrões frequentes requer um maior custo computacional por parte do algoritmo, por conta disso, implementou-se os recursos de computação paralela por meio do framework Spark e com o propósito de contar com uma concepção que minimize o tempo gasto pela distribuição em tarefas que possam ser executadas ao mesmo tempo.

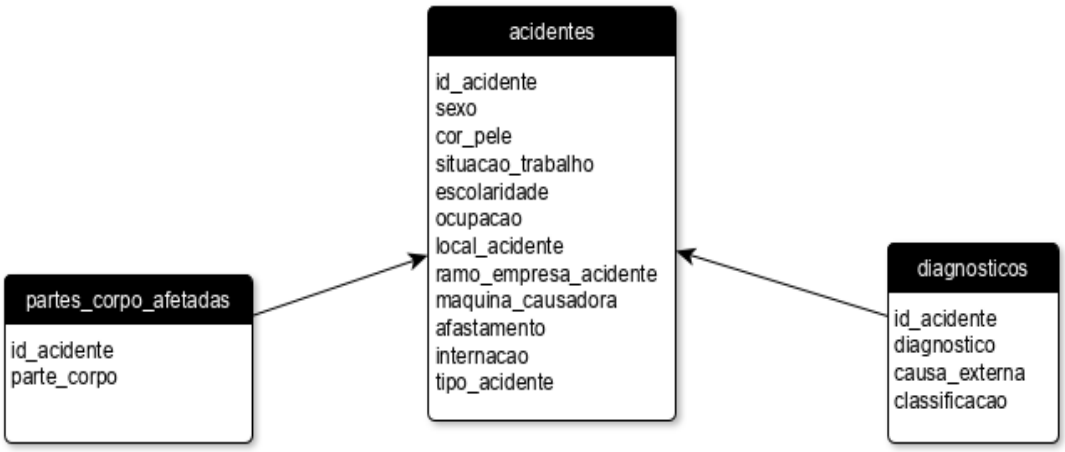
Para a realização dos testes, foi utilizado as seguintes configurações: Sistema Operacional Windows 10 Home, processador AMD Ryzen 5 3600 6-Core Processor 3.59 GHz, 16GB de memória RAM, 120GB de SSD, 1TB de HD e placa de vídeo RX 560 4GB.

Vale ressaltar que foram realizados 10 testes sendo utilizado a mesma base dados para chegar nos resultados obtidos.

4.2 Análise de acidentes de trabalho

O SIVAT é uma base de dados composta com registros de acidentes de trabalhos que aconteceram na região de São José do Rio Preto - SP. Essa base de dados possui três tabelas com muitos registros, sendo elas: *partes_corpo_afetadas*, *acidentes* e *diagnosticos*. Na Figura 8 é possível visualizar as tabelas e seus campos.

Figura 8 - Tabelas do SIVAT



Fonte: Elaborado pelo autor.

A tabela *partes_corpo_afetadas* possui 79043 tuplas, e descreve qual parte do corpo foi afetada em determinado acidente com seu respectivo identificador que se relaciona com a tabela *acidentes*. Já a tabela *diagnosticos* possui 126347 tuplas, e descreve o diagnóstico, a causa externa do acidente, a sua classificação e seu respectivo identificador que também se relaciona com a tabela *acidentes*. Por fim, temos a tabela *acidentes*, com 72017 tuplas, nas quais representam todas as informações gerais do acidente.

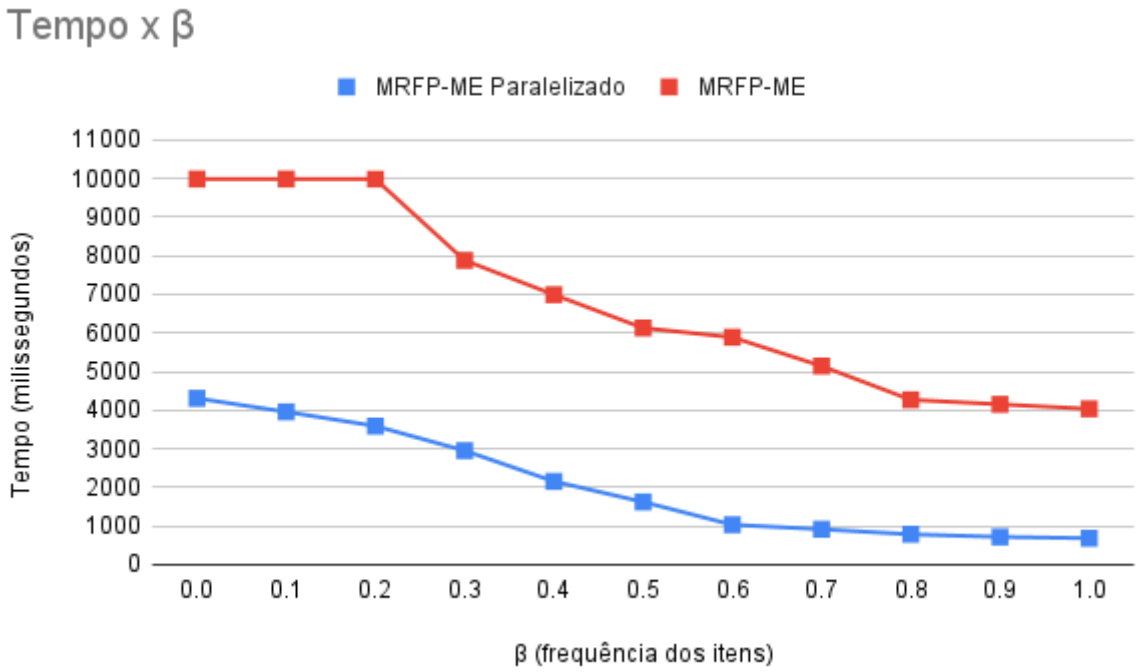
As tabelas *partes_corpo_afetadas* e *diagnosticos* possuim um relacionamento 1:N com a tabela *acidentes*, ou seja, um registro da tabela acidentes pode se relacionar com vários registros das outras duas tabelas, porém, cada uma das outras duas tabelas só podem se relacionar com apenas um acidente.

4.3 Resultados com o consumo de memória e o tempo de execução

Como dito anteriormente, a parte de mineração de padrões frequentes é a etapa que requer o maior custo computacional, por conta disso, os esforços para paralelizar o algoritmo foram focados para essa etapa.

No gráfico 1, é apresentado um comparativo entre os tempos de execução do algoritmo MRFP-ME e o MRFP-ME utilizando o Spark.

Gráfico 1 - Comparação do tempo de execução entre o MRFP-ME e o MRFP-ME utilizando o Spark



Fonte: Elaborado pelo autor

Por meio do gráfico 1 é possível observar que o tempo de execução do algoritmo com recursos de computação paralela, diminuiu drasticamente. Contudo, seguiu o decaimento semelhante ao do MRFP-ME original, isso ocorre pelo fato do MRFP-ME não utilizar a junção física entre as tabelas envolvidas na mineração. Mas é perceptível que, de acordo com o aumento dos valores de β, como esperado, o tempo de execução do algoritmo foi diminuindo. Uma diferença entre o MRFP-ME tradicional e o MRFP-ME ao utilizar-se de recursos da computação paralela, foi que o primeiro, no começo, o tempo de execução para os valores de $0.0 \geq \beta \geq 0.2$ permaneceram quase

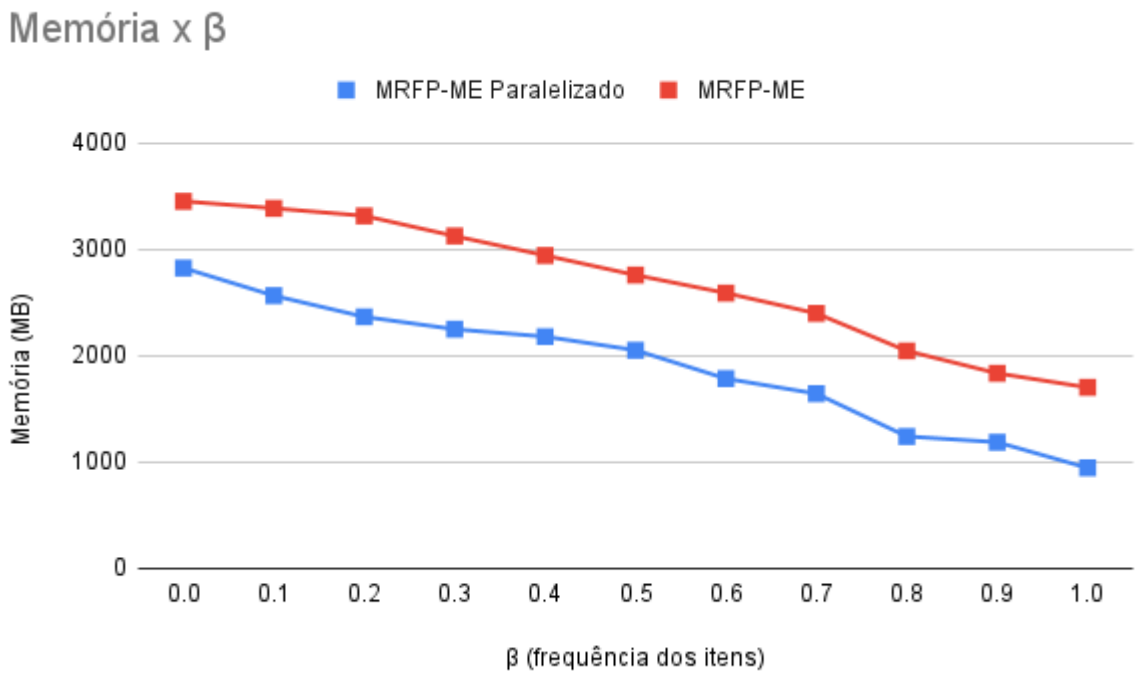
semelhantes, enquanto ao utilizar conceitos da computação paralela, o tempo de execução foi diminuindo gradativamente para esse mesmo intervalo dos valores de β.

Com isso, percebe-se que realmente a abordagem paralela do algoritmo MRFP-ME melhorou o tempo de execução, ao chegar no resultado esperado, com o aumento do desempenho do algoritmo.

Por conta da computação paralela lidar com o processamento em memória, foi perceptível que isso influenciou muito a reduzir o consumo de memória. Vale lembrar que, por conta do MRFP-ME lidar com itens relacionais e múltiplos suportes mínimos, possui um alto consumo de memória.

No gráfico 2 é possível visualizar tal análise.

Gráfico 2 - Comparação do consumo de memória entre o MRFP-ME e o MRFP-ME utilizando o Spark



Fonte: Elaborado pelo autor

Apesar do MRFP-ME ser um algoritmo que consome bastante memória, conforme visto no gráfico 2, o consumo de memória diminui com o algoritmo paralelizado, o que era esperado ao utilizar os conceitos de computação paralela. Essa diferença não foi tão impactante pelo fato do MRFP-ME considerar itens de várias tabelas e suas relações. Apesar

disso, foi possível perceber que o consumo de memória foi mais eficiente do que a abordagem tradicional do MRFP-ME.

O desvio padrão é uma medida de dispersão em torna da média, no qual o baixo valor do desvio padrão, indica que os valores estão perto da média ou do resultado esperado. Por outro lado, um alto valor do desvio padrão, indica que os valores da amostragem estão espalhados. Com isso, pode-se utilizar o desvio padrão como uma forma de validar os resultados esperados teoricamente.

Portanto, foi utilizado essa medida estatística para poder validar os resultados apresentados neste trabalho. Foi calculado um desvio padrão de 1.4 segundos para o tempo de execução do algoritmo. Com esse valor, pode-se concluir que os valores da amostragem estão próximos da média, portanto, próximos do resultado esperado. O mesmo vale para o consumo de memória, com um desvio padrão de 605,5 MB.

4.4 Considerações finais

Neste capítulo foram apresentados os experimentos e os resultados realizados neste trabalho, com o objetivo de avaliar e comparar o algoritmo MRFP-ME que utiliza conceitos de computação paralela e sua abordagem tradicional. Tais experimentos foram realizados diversas vezes na base de dados do SIVAT, pelo fato dela possuir um grande número de registros e as tabelas possuírem um maior número de relações entre si.

Com os testes realizados, percebe-se o benefício em utilizar a computação paralela para escalonar um algoritmo e melhorar o seu desempenho, o que foi obtido como resultado ao utilizar no algoritmo MRFP-ME.

Portanto, obteve-se um ganho em relação ao tempo de execução do algoritmo em relação ao algoritmo tradicional. Por sua natureza, o algoritmo MRFP-ME tradicional já possui um alto consumo de memória, porém, ao utilizar a computação paralela no algoritmo, obteve-se uma redução do consumo de memória, beneficiando ainda mais essa abordagem do algoritmo paralelo.

5 Conclusões

Este trabalho trouxe uma abordagem paralela do algoritmo MRFP-ME. Com essa nova abordagem, foi possível trazer um desempenho melhor para o MRFP-ME, no qual diminuiu o tempo de execução e o consumo de memória.

Com isso, é possível afirmar que o objetivo deste trabalho foi alcançado, pois manteve o objetivo do MRFP-ME, que é a mineração de regras de associação em bases de dados relacionais com uso de múltiplos suportes mínimos, e trouxe um ganho de desempenho e eficiência para o algoritmo.

5.3 Contribuições científicas

A contribuição científica deste trabalho foi apresentar uma versão paralela do algoritmo MRFP-ME tradicional. O algoritmo proposto mostrou desempenho superior de até 79.9% em relação ao consumo de memória e em até 4.8x mais rápido em relação ao correlato da literatura. Com isso, trouxe ganhos em relação ao tempo de execução e consumo de memória.

Tais comparações podem ser visualizadas na tabela 2 a seguir.

Tabela 2 - Comparação dos trabalhos correlatos com este trabalho

	Múltiplos suportes mínimos	Multi-relacional	Paralelismo
Rocha et al. (2018)	✗	✓	✗
Darrab e Ergenc (2017)	✓	✗	✗
Wang, Lin e Chang(2017)	✓	✗	✓
Nagao e Seki (2015)	✗	✓	✓
Valêncio et al. (2012)	✗	✓	✗
Valêncio et al. (2018)	✗	✓	✗
MRFP-ME (2018)	✓	✓	✗
Liu, Hsu e Ma (1999)	✓	✗	✗
Este trabalho	✓	✓	✓

Fonte: Elaborado pelo autor

5.4 Trabalhos Futuros

Como trabalhos futuros propõe-se ampliar o uso dos conceitos de computação paralela em outras partes do algoritmo com o propósito de melhorar ainda mais o seu desempenho. Outra possibilidade, é a utilização de *templates* no algoritmo MRFP-ME, conforme Valêncio et al. (2018) fez, na qual permite ter uma análise mais específica sobre os dados e reduz os números de regras irrelevantes geradas pelo algoritmo.

Como parte importante para ainda corroborar as contribuições deste trabalho e que deva contemplar uma nova versão com mais recursos de computação paralela, deve-se validar os recursos dos múltiplos suportes mínimos e muti-relacional. Estes dois recursos devem ser comprovados por meio de testes que demonstrem a amplitude alcançada por meio das regras de associação obtidas em relação ao respectivo algoritmo da literatura e que não contempla estes dois tipos de recursos.

Referências

Abdel-Basset, M., Mohamed, M., Smarandache, F., & Chang, V. (2018). Neutrosophic association rule mining algorithm for big data analysis. Symmetry, 10(4), 106.

DARRAB, S.; ERGENC, B. Vertical pattern mining algorithm for multiple support thresholds. **Procedia computer science**, v. 112, p. 417-426, 2017.

FREITAS, Alex A. A survey of parallel data mining. 1996.

GAN, W.; LIN, J.C.; FOURNIER-VIGER, P.; CHAO, H.; ZHAN, J. Mining of frequent patterns with multiple minimum supports. Engineering **Applications of Artificial Intelligence**, v. 60, p. 83-96, 2017.

GHANSHYAM, Thakur; CHANDRA JAIN, Ramesh. A Framework for Fast Classification Algorithms. 2008.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. Elsevier, 2011.

HUCK, Kevin A.; MALONY, Allen D. **Perfexplorer: A performance data mining framework for large-scale parallel computing**. In: SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing. IEEE, 2005. p. 41-41.

JENSEN, Viviane Crestana; SOPARKAR, Nandit. **Frequent itemset counting across multiple tables**. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Berlin, Heidelberg, 2000. p. 49-61.

KARTHIKEYAN, T.; RAVIKUMAR, N. **A survey on association rule mining**. International Journal of Advanced Research in Computer and Communication Engineering, v. 3, n. 1, p. 2278-1021, 2014.

MAIMON, Oded; ROKACH, Lior. **Introduction to knowledge discovery in databases**. In: Data mining and knowledge discovery handbook. Springer, Boston, MA, 2005. p. 1-17.

MAVRIDIS, I.; KARATZA, H.; Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. Journal of Systems and Software. Elsevier, v. 125, p. 133-151, mar. 2017.

NAGAO, Masahiro; SEKI, Hirohisa. Towards parallel mining of closed patterns from multi-relational data. In: 2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA). IEEE, 2015. p. 103-108.

OLSON, D.L.; LAUHOFF, G. Association Rules. In: **Descriptive Data Mining**. Springer, Singapore, 2019. p. 67-76.

SHI, R.; GAN, Y.; WANG, Y. Evaluating scalability bottlenecks by workload extrapolation. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, Milwaukee, p. 333-347, 25 set. 2018.

SHMEIS, Z.; JABER, M. A rewrite-based optimizer for spark. Future Generation Computer Systems. Elsevier, v. 98, p. 586-599, 06 abr. 2019.

SINGH, D.; REDDY, C. K. A survey on platforms for big data analytics. Journal of big data. Springer, Detroit, v. 2, n. 1, p. 8, 9 out. 2014.

TANIAR, David; RAHAYU, J. Wenny. Parallel data mining. In: Data mining: A heuristic approach. IGI Global, 2002. p. 261-289.

VALÊNCIO, C. R.; OYAMA, F. T.; NETO, P. S.; COLOMBINI, A. C.; CANSIAN, A. M.; DE SOUZA, R. C. G.; CORRÊA, P. L. P. MR-Radix: a multi-relational data-mining algorithm. **Human-centric Computing and Information Sciences**, v. 2, n. 1, p. 4, 2012.

VALÊNCIO, C. R.; OYAMA, F. T.; ICHIBA, F. T.; DE SOUZA, R. C. G. Multi-relational Algorithm for Mining Association Rules in Large Databases. In: **Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on**. IEEE, 2011. p. 269-274.

VALÊNCIO, Carlos Roberto et al. **A user-driven association rule mining based on templates for multi-relational data**. Journal of Computer Science, p. 1475-1487, 2018.

ZAKI, Mohammed J. **Parallel and distributed data mining: An introduction**. In: Large-scale parallel data mining. Springer, Berlin, Heidelberg, 2000. p. 1-23.

ZHANG, Feng et al. A distributed frequent itemset mining algorithm using Spark for Big Data analytics. Cluster Computing, v. 18, n. 4, p. 1493-1501, 2015.

ZHAO, Qiankun; BHOWMICK, Sourav S. **Association rule mining: A survey**. Nanyang Technological University, Singapore, v. 135, 2003.

WANG, C.; LIN, S.; CHANG, J. MapReduce-based frequent pattern mining framework with multiple item support. In: Asian Conference on Intelligent Information and Database Systems. Springer, Cham, 2017. p. 65-74.

Yu, Kun-Ming, et al. "Apriori-based high efficiency load balancing parallel data mining algorithms on multi-core architectures." International Journal of Grid and High Performance Computing (IJGHPC) 7.2 (2015): 77-99.