MiRABIT: Um Novo Algoritmo para Mineração de Regras de Associação

Sandro da Silva Camargo
Universidade da Região da Campanha (URCAMP)
96.400-101 – Bagé – RS – Brasil
scamargo@urcamp.tche.br

Paulo Martins Engel
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
engel@inf.ufrgs.br

Resumo. A mineração de regras de associação em um banco de dados de transações de vendas é uma tarefa extremamente cara em termos de tempo de processamento. Dependendo do ramo de atividade da empresa, algumas características do banco de dados são diferentes. Os algoritmos utilizados atualmente otimizam a mineração para bancos de dados com alto tamanho médio de transação, tais como supermercados. O algoritmo MiRABIT otimiza a mineração de regras de associação para bancos de dados com baixo tamanho médio de transação, tais como comércio de confecção. Testes comparativos demonstram a eficiência do algoritmo MiRABIT em relação ao algoritmo padrão Apriori em um banco de dados de comércio de confecção.

Palavras-chave: Mineração de dados, regras de associação, otimização de algoritmos.

1. Introdução

Na atualidade, a maioria das empresas interage com seus clientes através de computadores, ficando estas interações armazenadas em um banco de dados na forma de transações. Uma transação de venda consiste em um conjunto de itens vendidos a um cliente. Dado um banco de dados de transações de vendas, a mineração de regras de associação é uma forma de encontrar grupos de itens que tendem a ocorrer juntos em uma transação, ou seja, as associações significativas, do ponto de vista do algoritmo, entre itens tal que a presença de alguns itens em uma transação implicará a presença de outros itens na mesma transação.

Uma afirmação na forma "c% dos clientes que compram o item A também compram o item B" consiste em uma regra de associação.

1.1. Formalização do problema

Um modelo matemático foi proposto em Agrawal, Imielinski e Swami (1993) para descrever o problema de mineração de regras de associação. Tem-se que $I = \{i_1, i_2, i_3, ..., i_m\}$ é um conjunto de atributos binários, chamados itens. Tem-se que $D = \{T_1, T_2, T_3, ..., T_m\}$ é um banco de dados de transações, onde cada transação T é um conjunto de itens tal que $T \subseteq I$. Cada transação T é representada como um vetor binário, com $t[k]=1$ se T contém o item i_k, e $t[k]=0$ caso contrário. Há uma tupla no banco de dados para cada transação. Considerando que X é um conjunto de alguns itens em I, uma transação T contém X se para todos itens i_k

em X, $t[k]$=1. Considerando que Y é um conjunto de alguns itens em I, uma transação T contém Y se para todos itens i_k em Y, $t[k]$=1.

Uma regra de associação consiste em uma implicação da forma $X => Y$, onde $X \subset I$, $Y \subset I$, $X \cap Y = \varnothing$ e $(X \cup Y) \subset T$, ou seja, X é um conjunto de alguns itens de I, e Y é um conjunto de itens em I que não está presente em X. A regra $X => Y$ é satisfeita no conjunto de transações D com o fator de confiança $0 \leq c \leq 1$, se no mínimo c% das transações em D que satisfizerem X também satisfaçam Y. Será utilizada a notação $X => Y \mid c$ para especificar que a regra $X => Y$ tem um fator de confiança de c. A regra $X => Y$ é satisfeita no conjunto de transações D com o fator de suporte s se s% das transações em D contiverem $X \cup Y$.

Dado um conjunto de transações D, é necessário gerar todas as regras que satisfaçam certas restrições, sob dois aspectos diferentes:

1. Limites sintáticos: envolvem restrições em itens que podem aparecer em uma regra. Por exemplo, pode-se estar interessado apenas em regras que tenham um item específico i_x aparecendo no conseqüente, ou regras que têm um item específico i_y aparecendo no antecedente. Combinações sobre limites também são possíveis – podem ser necessárias todas as regras que têm itens de algum conjunto predefinido X aparecendo no conseqüente, e itens de algum outro conjunto Y aparecendo no antecedente.
2. Limites de suporte: referem-se ao número de transações em D que obedecem a regra. O suporte para uma regra é definido como sendo uma fração de transações em D que satisfaçam a união dos itens em conseqüente e antecedente de uma regra.

Conceitualmente, enfatiza-se a diferença entre suporte e confiança: enquanto confiança é uma medida da força da regra, suporte corresponde a uma significância estatística. Além da significância estatística, outra motivação para o limite de suporte vem do fato de geralmente haver interesse apenas nas regras com suporte maior que algum limiar mínimo por razões do negócio. Se o suporte não é grande o suficiente, isto significa que a regra não vale a pena ser considerada ou que ela é simplesmente menos importante, sendo estas regras eliminadas para otimização de tempo de execução do processo.

1.2. Decomposição do problema

Nesta formulação, o problema de mineração de regras pode ser decomposto em dois subproblemas Agrawal, Imielinski e Swami (1993):

1. Geração de todas combinações de itens que têm uma fração de suporte transacional maior que um certo limiar, chamado suporte mínimo. Estas combinações são chamadas de itens freqüentes, e todas outras combinações com suporte menor que o limiar são chamadas de itens não freqüentes. Limites sintáticos adicionalmente limitam as combinações admissíveis contribuindo para a diminuição do tempo de execução. Por exemplo, se apenas regras envolvendo um item i_x em um antecedente são interessantes, então é suficiente gerar apenas aquelas combinações que contenham i_x no antecedente.
2. Para um dado conjunto de itens freqüentes $Y = \{i_1, i_2, ..., i_k\} \mid k >= 2$, gerar todas as regras que contenham itens do conjunto $i_1, i_2, ..., i_k$. O antecedente de cada uma destas regras será um subconjunto X de Y tal que X tem k-1 itens, e o conseqüente será o item $Y - X$. Para gerar uma regra $X => Y \mid c$, onde $X = i_1, i_2, ..., i_{j-1}, i_{j+1}, i_k$, é necessário dividir o suporte de Y pelo suporte de X. Se a taxa é maior que c e que o fator de confiança mínimo definido então a regra é satisfeita com o fator de confiança c, caso contrário não.

Se o conjunto de itens *Y* é freqüente, então todo subconjunto de *Y* também será freqüente, e, também, todas regras derivadas de *Y* satisfazem o limite de suporte se *Y* satisfaz o limite de suporte.

O primeiro passo faz sucessivas passagens sobre o banco de dados e é responsável pela maior parte do tempo de processamento. Deste passo resulta o conjunto de itens freqüentes e seus respectivos contadores de suporte. O segundo passo é pouco oneroso em tempo de processamento. Deste passo, a partir do resultado da primeira passagem é gerado o conjunto de regras.

1.3 Trabalhos Relacionados

O algoritmo *Apriori* proposto por Agrawal e Srikant (1994) é o algoritmo mais utilizado para mineração de regras de associação. Este algoritmo faz várias passagens sobre o banco de dados e utiliza um procedimento de geração de itens candidatos no qual apenas os itens frequentes em uma passagem são combinados para gerar os conjuntos de itens candidatos na próxima passagem. Desde o *Apriori* diversos outros algoritmos têm sido propostos na literatura com a finalidade de melhorar o desempenho da tarefa da mineração utilizando outras técnicas para tratar o problema.

Uma das técnicas utilizadas é diminuir o I/O é diminuindo o número de passagens sobre o banco de dados. Como exemplo desta técnica pode ser citado o algoritmo *Partition* de Savarese e Zaki (1997) que minimiza o I/O passando apenas duas vezes sobre o banco de dados. Este algoritmo particiona o banco de dados em pequenas partições que podem ser manipuladas em memória, na primeira passagem os conjuntos de itens potencialmente frequentes são gerados e segunda passagem são contados. Outros algoritmos como o *Eclat*, *MaxEclat*, *Clique* e *MaxClique* propostos por Zaki, Parthasarathy, Ogihara e Li (1997) utilizam uma única passagem sobre o banco de dados fazendo um processo de clusterização antes de gerarem e contarem os conjuntos de itens candidatos. Outra maneira de diminuir o I/O é utilizando a técnica de *sampling* apresentado por Zaki, Parthasarathy, Li e Ogihara (1997) , onde não é analisado todo o banco de dados mas apenas um conjunto de transações é selecionado por amostragem para ser analisado.

2. Análise dos bancos de dados

De acordo com Camargo (2002) dependendo do ramo de atividade da empresa algumas características dos bancos de dados podem ser diferentes, tais como o tamanho médio das transações. O tamanho médio das transações é uma característica peculiar ao ramo de atividade, refletindo as características do negócio, que influi decisivamente no desempenho do algoritmo de mineração de regras de associação.

A seguir são analisados bancos de dados reais de três empresas de ramos de atividades diferentes e demonstradas suas diferentes características.

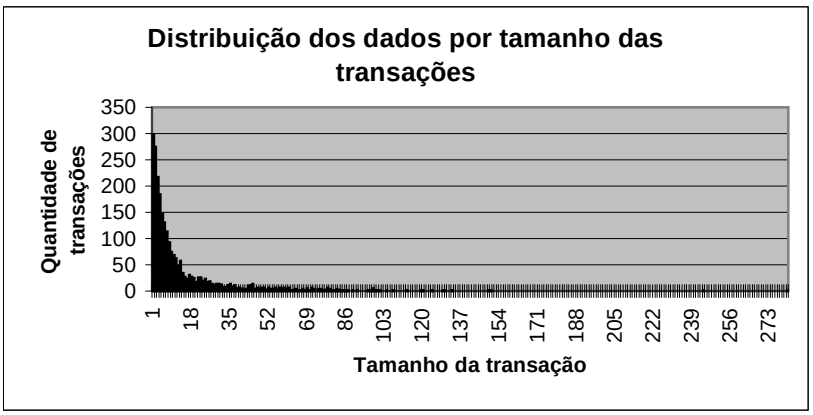

Figura 1. Distribuição do tamanho das transações em um banco de dados de supermercado

A figura 1 mostra o gráfico de distribuição dos dados por tamanho das transações em um banco de dados real de um ambiente de supermercado. Neste banco de dados foram identificadas, no período de um mês de vendas, 2449 transações com 34375 itens o que resulta em uma média de tamanho das transações de aproximadamente 14,03 itens. O tamanho da maior transação encontrada era de 281 itens.

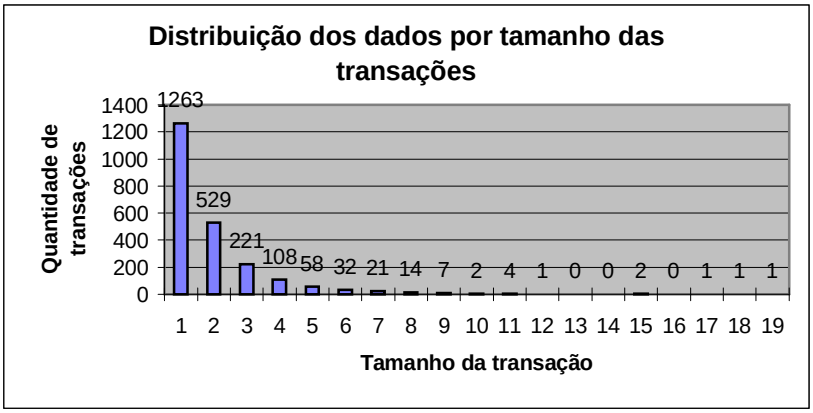

Figura 2. Distribuição do tamanho das transações em um banco de dados de uma empresa de comércio varejista de confecção.

A figura 2 apresenta o gráfico de distribuição dos dados por tamanho das transações em um banco de dados real de uma empresa de comércio varejista de confecção. Neste banco de dados foram identificadas, no período de um mês de vendas, 2266 transações e 4380 itens o que resulta em uma média de aproximadamente 1,93 itens por transação. A maior transação encontrada possuía 19 itens.

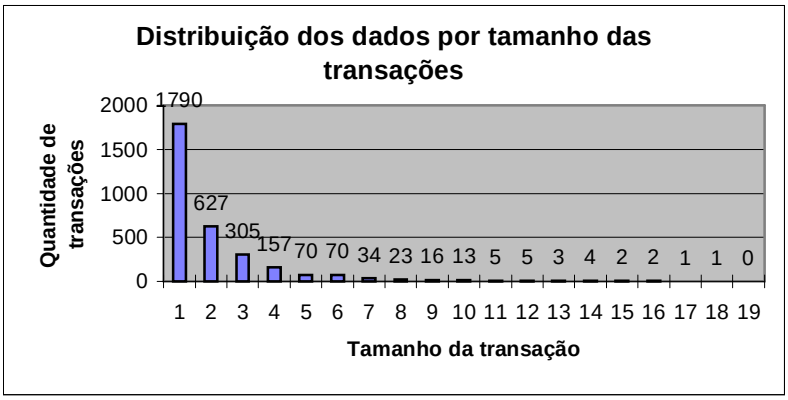

Figura 3. Gráfico do tamanho das transações em um banco de dados de uma empresa de comércio varejista de ferragens.

A figura 3 apresenta o gráfico de distribuição dos dados por tamanho das transações em um banco de dados real de uma empresa de comércio varejista de ferragens. Neste banco de dados foram identificadasanalisadas, no período de um mês de vendas, 3132 transações e 6454 itens o que resulta em uma média de aproximadamente 2,06 itens por transação. A maior transação encontrada possuía 31 itens.

Nos trabalhos encontrados na literatura verificou-se que os testes de desempenho dos algoritmos utilizam bancos de dados com tamanho médio de transação similar ou maior ao tamanho médio de transação do setor de supermercados. Em Agrawal e Srikant (1994) os testes de desempenho são executados em bancos de dados com um tamanho médio de transação entre 5 e 20. Em Bayardo e Agrawal (1999), os testes de desempenho são executados em bancos de dados com tamanho médio de 43 e 49 itens por transação. Entretanto, tanto os setores de comércio varejista de confecção, como o setor de ferragens, apresentam um tamanho médio de transação abaixo do intervalo utilizado nos testes de desempenho.

3. Proposta

Conforme demonstrado através da análise de diferentes bancos de dados, percebe-se que há bancos de dados com outras características diferentes das de supermercados. Estas características podem ser exploradas a fim de aumentar o desempenho da tarefa de descoberta de regras de associação.

Neste sentido, foram executados vários experimentos com os algoritmos estudados, analisando-se seus pontos críticos de desempenho. Após a análise do algoritmo *Apriori* proposto por Agrawal, Imielinski e Swami (1993), identificou-se que um dos pontos críticos de desempenho do algoritmo é a geração de conjuntos de itens candidatos para a passagem sobre o banco de dados, que é executada no início de cada passagem. Conforme demonstrado posteriormente através da análise do banco de dados típico de um supermercado, verificou-se que, na segunda passagem do algoritmo, 34% dos conjuntos de itens candidatos tornaram-se freqüentes. Por outro lado verificou-se que, em um banco de dados típico de uma empresa do comércio varejista de confecção, apenas 0,03% dos conjuntos de itens candidatos tornaram-se freqüentes. Constatou-se então que a geração dos conjuntos de itens candidatos no início de cada passagem melhora o desempenho do algoritmo apenas quando grande parte dos conjuntos de itens candidatos torna-se freqüente. Para que isto ocorra é necessário que a média de itens por transação seja alta. No banco de dados do supermercado esta média era em torno de 14 itens por transação, já no comércio varejista era em torno de 1,93 itens por transação. A geração dos conjuntos de itens candidatos torna-se mais crítica em um ambiente

de comércio varejista de confecção, e outros bancos de dados com tamanho médio de transação similar, onde a razão entre itens freqüentes e itens candidatos é bem menor que em um ambiente de supermercado. Constatou-se então que, nestes bancos de dados, seria mais eficiente gerar o conjunto de itens candidatos a cada transação do que gerar o conjunto de itens candidatos no início de cada passagem passagem através da combinação dos conjuntos de itens freqüentes da passagem anterior.

A partir desta análise é proposto que, em bancos de dados com uma pequena quantidade de itens por transação, a geração de conjuntos de itens candidatos seja executada a cada transação, e não no início da passagem, como é feito pelo algoritmo *Apriori*, resultando em uma melhora de desempenho da tarefa de descoberta de regras de associação.

Para provar esta hipótese foram executados experimentos para verificar o desempenho de um novo algoritmo, baseado no algoritmo *Apriori* de Agrawal e Srikant (1994) com um processo de geração de conjuntos de itens candidatos mais eficiente em bancos de dados com baixo tamanho médio de transações. Os resultados apresentados no item 4 demonstram uma melhora de desempenho do algoritmo proposto em relação ao algoritmo *Apriori* nas áreas de comércio varejista de confecção e comércio varejista de ferragens.

3.1 Descoberta de conjuntos de itens frequentes

A partir da análise desenvolvida no item anterior, é proposto um novo algoritmo, chamado MiRABIT, apresentado em Camargo e Engel (2001), cuja principal característica é que a geração dos conjuntos de itens candidatos a cada transação e não no início da passagem *k* tal qual o algoritmo *Apriori* e similares. O principal objetivo desta característica é melhorar o desempenho da tarefa de mineração de regras de associação em bancos de dados com um baixo tamanho médio de transações. Como conseqüência desta alteração, o algoritmo MiRABIT gera um conjunto de itens candidatos geralmente entre 0,5 e 2 % do conjunto de itens gerados pelo *Apriori* no caso em estudo, o que resulta em um relevante ganho de desempenho.

A figura 4 apresenta o pseudocódigo do algoritmo MiRABIT.

```
F1 = {conjuntos de itens freqüentes com 1- elemento};
  for (k = 2; (( |Fk-1| ≥ 2) or (k ≤ tamanho_maximo_regra)); k++) do begin
    forall transações t ∈ D do begin
      Ct = subset (Fk-1,t,k); // Gera possíveis candidatos na transação
      forall candidatos c ∈ Ct do
        c.count ++;
      Ck=Ck ∪ Ct;
    end
    Fk = {c ∈ Ck | c.count ≥ suporte_minimo}
    Answer=Answer ∪ Fk;
```

Figura 4. Pseudocódigo do algoritmo MiRABIT.

A primeira linha do pseudocódigo apresentado na figura 4 demonstra que a execução do algoritmo parte do conjunto de itens freqüentes com 1 elemento. Para gerar este conjunto,

o algoritmo necessita passar uma vez sobre o banco de dados contando em quantas transações cada item está presente.

A segunda linha do pseudocódigo define uma estrutura de controle de iteração. Esta iteração será controlada pela variável *k* que identifica o número da passagem do algoritmo sobre o banco de dados. A variável *k* tem seu valor inicial definido como 2 e a iteração será executada enquanto o conjunto de itens freqüentes gerado na passagem anterior tiver 2 ou mais elementos. A iteração deixará de ocorrer se o conjunto tiver menos de dois elementos pois será feita uma combinação de *n* itens em conjuntos com *p* itens, logo *p* não poderá ser menor que *n*. Outra condição para a iteração ser executada é que *k* seja menor ou igual ao valor definido pelo usuário como tamanho máximo da regra. Esta restrição limita a quantidade de passagens do algoritmo sobre o banco de dados, diminuindo também o tempo para execução do algoritmo. A cada nova iteração, o valor da variável *k* é incrementado de uma unidade.

A cada nova passagem sobre o banco de dados diminui a probabilidade de ser encontrado um conjunto de itens freqüente; por este fato, a utilização de uma restrição de tamanho máximo da regra pode ser muito importante. Tendo-se um banco de dados, a probabilidade de um item qualquer ser freqüente é determinada pela razão entre o número de itens freqüentes na primeira passagem e o número total de itens; este resultado é elevado à ordem da passagem. Por exemplo, em um banco de dados com a distribuição de itens por transação mostrada na figura 2, temos 1263 / 26330 = 0,048 ≅ 5% de probabilidade para a primeira passagem. A probabilidade de um conjunto de 2 itens ser freqüente é $0,048^2 = 0,002$ ≅ 0,2%. A probabilidade de um conjunto de 3 itens ser freqüente é $0,048^3 = 0,00011$ ≅ 0,011%.

A terceira linha do pseudocódigo define uma nova estrutura de controle de iteração que obriga a execução das linhas seguintes para todas as transações no banco de dados. Neste ponto é importante salientar-se que a seleção, que é um dos passos do pré-processamento, já foi executada, sendo portanto consideradas somente as transações que atendem as restrições pré-estabelecidas.

A quarta linha do pseudocódigo executa a função *subset* que terá seu funcionamento detalhado posteriormente. A função *subset* gera todos os conjuntos de itens candidatos para cada transação.

Na quinta linha do pseudocódigo é definida uma estrutura de controle de iteração que obriga a execução das linhas posteriores para todos os conjuntos de itens gerados anteriormente pela função *subset*.

Na sexta linha é definida a instrução para que o contador de cada conjunto de itens encontrado na transação seja incrementado em uma unidade.

Na sétima linha o conjunto de itens candidatos encontrado na transação (C_t) é inserido no conjunto de itens candidatos da passagem.

Na oitava linha é finalizada a estrutura de controle iniciada na terceira linha.

Na nona linha do pseudocódigo é definida a instrução que irá incluir os conjuntos de itens candidatos com suporte maior ou igual ao suporte mínimo no conjunto de itens freqüentes da passagem. Esta linha será executada após a análise de todas as transações do banco de dados selecionadas para serem mineradas.

Na décima linha do pseudocódigo é definida a instrução que irá inserir os conjuntos de itens freqüentes na passagem *k* ao conjunto resposta. Ao final da execução do algoritmo o

conjunto resposta conterá todos os conjuntos de itens freqüentes encontrados em cada uma das passagens do algoritmo sobre o banco de dados juntamente com seus respectivos contadores de suporte.

Na décima primeira linha do algoritmo é definido o fim da estrutura de controle iniciada na segunda linha e finalizada a execução do algoritmo.

A partir dos conjuntos de itens freqüentes são geradas as regras de associação que satisfaçam a restrição de confiança mínima definida pelo usuário.

3.2.1 Função *Subset*

No algoritmo MiRABIT a função *subset* tem como finalidade principal identificar que conjuntos de itens em uma transação são candidatos para posterior contagem destes conjuntos de itens. Esta função recebe como argumentos o conjunto de itens freqüentes na passagem anterior F_{k-1} e a transação atual *t*. Cada conjunto de itens *c* contido na transação que seja derivado de um conjunto de itens freqüentes na passagem anterior é incluído no conjunto de itens candidatos da transação C_t para sua posterior contagem.

forall c ∈ t **do**

forall (*k*-1)-subsets of $c \in F_{k-1}$ then

$c \in C_t$;

Figura 5. Pseudocódigo da função *subset* do algoritmo MiRABIT.

A função *subset* apresentada na figura 5 gera as possíveis combinações de itens em conjuntos de *k* itens e também implementa uma função de poda, onde para cada conjunto de itens gerado em uma transação, é verificado se algum de seus subconjuntos não é freqüente; em caso positivo, este conjunto de itens não é considerado candidato e por conseqüência, seu suporte não é contado.

Visto que o conjunto de itens conjuntos gerados é menor que no algoritmo *Apriori*, conforme demonstrado em experimentos no próximo capítulo, a quantidade de memória utilizada durante o processo também é menor.

4. Análise comparativa de desempenho

Segundo Toscani e Veloso (2001), a complexidade de um algoritmo pode ser medida pela complexidade de tempo e pela complexidade de espaço. A complexidade de tempo refere-se à velocidade do algoritmo. A complexidade de espaço refere-se à quantidade de memória necessária para a execução deste algoritmo. Os algoritmos Apriori e MiRABIT foram implementados e aplicados sobre duas bases de dados reais de diferentes ramos de negócio para serem executados os testes de desempenho. Por ambos bancos de dados apresentarem características de sazonalidade o período selecionado para ser executada a mineração foi de um mês. Conforme Camargo (2002), quando é executada a mineração em bancos de dados com características de sazonalidade, a execução do processo sobre um período maior a qualidade das regras geradas fica comprometida.

A seguir é apresentada uma tabela com o resumo dos resultados obtidos nos nove experimentos executados.

TABELA 1 – Análise comparativa de desempenho

Ramo de Negócio do Banco de Dados	Suporte Mínimo	*Apriori*		MiRABIT		Ganho	
		Tempo (s)	Conjuntos de itens candidatos	Tempo (s)	Conjuntos de itens candidatos	Tempo %	Espaço %
	2	130	648707	50	2509	61,53	99,61
confecção	3	49	107594	34	1419	30,61	98,68
T_m=1,93	4	30	39344	24	830	20	97,89
	5	28	17582	21	545	25	96,90
	3	557	1516711	372	4235	33,21	99,72
ferragem	4	210	278911	134	3185	36,19	98,85
T_m=2,06	5	166	82051	127	2440	23,49	97,02
	6	89	39579	57	1494	35,95	96,22
	7	77	22261	51	1487	33,76	93,32
Ganho médio em complexidade						33,30	97,57

Tem-se que T_m significa o tamanho médio das transações em cada um dos bancos de dados. As duas últimas colunas demonstram o ganho de desempenho obtido nas duas dimensões de complexidade analisadas: tempo e espaço. O ganho relativo à complexidade de tempo denota a quantidade de tempo ganha com a utilização do algoritmo MiRABIT em relação ao algoritmo *Apriori*. O ganho relativo à complexidade de espaço denota o quanto menor foi a quantidade de conjuntos de itens candidatos gerada pelo algoritmo MiRABIT em relação ao algoritmo *Apriori*.

Conforme resultados apresentados na tabela 1, o algoritmo MiRABIT obteve um ganho médio de tempo de 33,30% em relação ao algoritmo *Apriori* para executar a tarefa de mineração de regras de associação. Em relação ao espaço, o conjunto de itens candidatos gerado pelo algoritmo MiRABIT foi, em média, 97,57% menor do que o conjunto de itens candidatos gerado pelo algoritmo *Apriori*. O algoritmo Mirabit, por gerar um conjunto de itens candidatos menor que o algoritmo *Apriori*, necessita de requisitos de hardware mais modestos para executar a tarefa de mineração de regras de associação. Também é importante salientar que, conforme demonstrado nos experimentos, o resultado final foi exatamente o mesmo, ou seja, ambos algoritmos geraram o mesmo conjunto de regras de associação por obterem o mesmo conjunto de itens freqüentes.

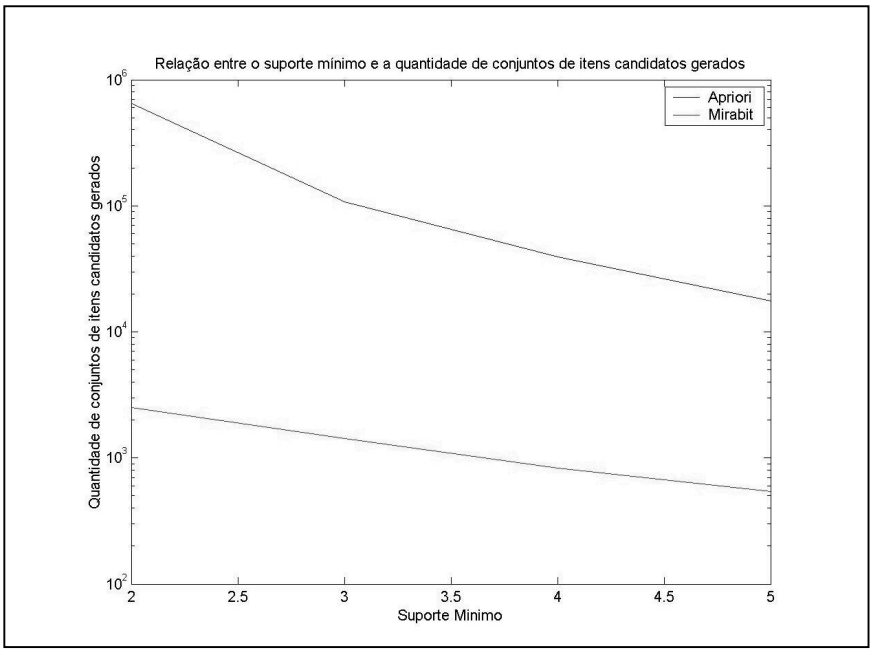

Figura 5. Pseudocódigo da função *subset* do algoritmo MiRABIT.

A figura 5 demonstra um gráfico em notação logarítmica da relação entre o suporte mínimo e a quantidade de conjuntos de itens candidatos gerados pelos algoritmos MiRABIT e Apriori.

A diminuição do número de conjunto de itens candidatos do algoritmo *Apriori* se dá de forma exponencial porque a função *Apriori-gen* gera todas as combinações possíveis entre conjuntos de itens através de análise combinatorial. Como o algoritmo Mirabit gera mais eficientemente os conjuntos de itens candidatos que o algoritmo *Apriori*, a quantidade destes conjuntos de itens tende a diminuir de forma quase linear. Por este motivo, à medida que o suporte mínimo aumenta o ganho de espaço obtido pelo algoritmo Mirabit tende a desaparecer. À medida que o suporte mínimo diminui o número de passagens sobre o banco de dados tende para 1. Como a primeira passagem sobre o banco de dados de ambos os algoritmos é idêntica, a medida que o suporte mínimo aumenta o tempo de execução de ambos os algoritmos tenderá a ser o mesmo.

Como pode ser visto na tabela 1 o ganho de espaço diminui gradativamente à medida que o suporte mínimo aumenta. Isto já não ocorre com o ganho de tempo pois esta medida depende da quantidade de conjuntos de itens freqüentes encontrados na passagem.

Também pode ser verificado pela tabela 1 que o ganho de tempo não é diretamente proporcional ao ganho de espaço. Este fato ocorre por haver um *overhead* para que seja feita a geração mais eficiente de conjuntos de itens candidatos. Como pode ser verificado na discussão individual do resultado de cada experimento, este *overhead* é menor nas passagens iniciais do algoritmo sobre o banco de dados e tende a aumentar à medida que aumenta o número da passagem sobre o banco de dados.

5. Conclusões

Dentre os algoritmos estudados, constatou-se uma preocupação dirigida à primeira fase da mineração de regras de associação, que é a parte mais onerosa processamento. Nesta fase, o algoritmo executa múltiplas passagens sobre o banco de dados e uma das principais

características dos algoritmos estudados é que no início cada passagem são gerados todos os seus possíveis conjuntos de itens candidatos. Foram realizados experimentos para avaliação de desempenho onde constatou-se que a geração de conjuntos de itens candidatos pode ser otimizada, se executada a cada transação ao invés de ser executada no início da passagem. Esta alteração aumenta o desempenho do processo de mineração de regras de associação em bancos de dados com uma baixa média de itens por transação. Também foi observado, dentre os trabalhos pesquisados, uma preocupação dirigida a bancos de dados com alto tamanho médio de transação, não sendo encontrada na literatura pesquisas dirigidas a bancos de dados com baixo tamanho médio de transação. Esta constatação nos motivou a propor um algoritmo dirigido para a descoberta de regras de associação aplicado ao comércio varejista de confecção, que tem como principal característica a não geração de conjuntos de itens candidatos, diminuindo assim o tempo de processamento. Foram realizados testes de desempenho onde o algoritmo proposto demonstrou sua maior eficiência em relação ao algoritmo *Apriori* quando aplicados ao tipo de problema em questão.

Referências

Agrawal, Rakesh; Imielinski, Tomasz; Swami, Arun. Mining Associations between Sets of Items in Massive Databases. In: ACM SIGMOD INT. CONFERENCE ON MANAGEMENT OF DATA, 1993. Proceedings... Washington D.C.: ACM Press, 1993, p. 207-216.

Agrawal, Rakesh; Srikant, Ramakrishnan. Fast Algorithms for Mining Association Rules. In. INT. CONFERENCE ON VERY LARGE DATABASES, 20, 1994. Proceedings... Hove: Morgan Kaufmann, 1994, p.487-499.

Bayardo Jr., Roberto J.; Agrawal, Rakesh. Mining the Most Interesting Rules. In: ACM SIGKDD INT. CONFERENCE ON KNOWLEDGE DISCOVERY, 5, 1999. Proceedings... San Diego: ACM Press, 1999, p. 145-154.

Berry, Michael J.A. and Linoff, Gordon. Data Mining Techniques, John Wiley & Sons ltd., 1997.

Camargo, Sandro da Silva, Engel, Paulo Martins. "MIRABIT: Mineração de regras de associação em bancos de dados de comércio varejista de confecção." In: Revista do Ccei, 2001. Bagé: Ediurcamp, 2001, v.5, n.8, p.12-18, 2001.

Camargo, Sandro da Silva, Engel, Paulo Martins. "MiRABIT: Um novo algoritmo de mineração para regras de associação." In: OFICINA DE INTELIGÊNCIA ARTIFICIAL, 5, 2001. Anais... Pelotas: Educat, 2001.

Camargo, Sandro da Silva. Mineração de regras de associação no problema da cesta de compras aplicada ao comércio varejista de confecção. Dissertação de mestrado (Mestrado em Ciência da Computação), Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

Savarese, A.; Omiecinski, E.; Navaux, S. An efficient algorithm for mining association rules in large databases. In: INT. CONFERENCE ON VERY LARGE DATABASES, 21, 1994.

Toscani, Laira Vieira; Veloso, Paulo A. S.. Complexidade de algoritmos: análise, projeto e métodos. Porto Alegre: Sagra Luzzatto, 2001. 202p.

Zaki, Mohammed Javeed; Parthasarathy, Srinivasan; Li, Wei, Ogihara, Mitsunori. Evaluation of sampling for data mining of association rules. In: INT. CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3, 1997.

Zaki, Mohammed J.; Parthasarathy, Srinivasan; Li, Wei. New Algorithms for Fast Discovery of Association Rules. In: ACM KDD INT. CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 3, 1997. Proceedings... Newport: AAAI Press, 1997, p. 283-286.